Algorithms for Conditional Search Space Optimal Layout Problems

A dissertation submitted by

Juliette Gamot

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Informatics)

ÉCOLE DOCTORALE SCIENCES POUR L'INGÉNIEUR
UNIVERSITÉ DE LILLE
Lille, France

&

ONERA - THE FRENCH AEROSPACE LAB
Palaiseau, France

Defended publicly on the 18th of December 2023 in front of a jury composed of :

Pr. Pascal LAFON	Professor, Université de Technologie de Troyes, Troyes	Reviewer
Pr. Frédéric SAUBION	Professor, Université d'Angers, Angers	Reviewer
Pr. Patrick SIARRY	Professor, Université Paris-Est-Créteil, Créteil	Examiner
Dr. Delphine SINOQUET	Research engineer, IFPEN, Rueil-Malmaison	Examiner
Dr. Mathieu BALESDENT	Director of Research, ONERA, Palaiseau	Thesis advisor
Pr. Nouredine MELAB	Professor, Université de Lille, Lille	Thesis director
Pr. El-Ghazali TALBI	Professor, Université de Lille, Lille	Invited
Dr. Romain WUILBERCQ	Research engineer, ONERA, Palaiseau	Invited

Développement d'algorithmes pour les problèmes d'agencement interne à espace de recherche conditionnel

Thèse de doctorat présentée et soutenue par :

Juliette Gamot

en vue d'obtenir le grade de

Docteur d'Université

en Informatique

ÉCOLE DOCTORALE SCIENCES POUR L'INGÉNIEUR
UNIVERSITÉ DE LILLE
Lille, France

&

ONERA - THE FRENCH AEROSPACE LAB
Palaiseau, France

Soutenue publiquement le 18 décembre 2023 devant un jury composé de :

Pr. Pascal LAFON	Professeur, Université de Technologie de Troyes, Troyes	Rapporteur
Pr. Frédéric SAUBION	Professeur, Université d'Angers, Angers	Rapporteur
Pr. Patrick SIARRY	Professeur, Université Paris-Est-Créteil, Créteil	Examinateur
Dr. Delphine SINOQUET	Ingénieure de recherche, IFPEN, Rueil-Malmaison	Examinatrice
Dr. Mathieu BALESDENT	Directeur de recheche, ONERA, Palaiseau	Encadrant
Pr. Nouredine MELAB	Professeur, Université de Lille, Lille	Directeur
Pr. El-Ghazali TALBI	Professeur, Université de Lille, Lille	Invité
Dr. Romain WUILBERCQ	Ingénieur de recherche, ONERA, Palaiseau	Invité

Remerciements/Acknowledgments

Me voici enfin face à ces dernières pages blanches qui me permettent de conclure cette thèse en remerciant un certain nombre de personnes qui ont toutes contribué, par leur présence et leur soutien, à faire de ces trois années de thèse une aventure tant scientifique qu'humaine.

Je souhaite, en premier lieu, remercier Pascal Lafon et Frédéric Saubion qui ont accepté avec beaucoup d'entrain la tâche de relire et d'évaluer ce manuscrit. Merci pour vos relectures minutieuses dans un temps imparti relativement court, pour vos précieuses remarques et pour l'intérêt que vous avez porté à mon travail. J'exprime aussi ma profonde gratitude envers Patrick Siarry et Delphine Sinoquet pour avoir accepté d'examiner mes travaux de thèse et pour la discussion au cours de la soutenance, qui ouvre de vastes perspectives au-delà de ce manuscrit.

Je remercie mon directeur de thèse, Nouredine Melab, pour m'avoir permis de travailler sur ce sujet en acceptant de diriger mes travaux. Merci pour tes conseils sur les conférences et journaux scientifiques, merci pour ta relecture du manuscrit et ton aide lors de l'organisation de la soutenance, tout ça dans un timing effréné !

Mes prochains remerciements vont indéniablement à mon trio d'encadrants : Mathieu, Romain et Arnault. Je vous remercie grandement pour votre encadrement quotidien, les réunions hebdomadaires où vous étiez presque systématiquement présents tous les trois, ainsi que pour les relectures assidues de toutes mes productions écrites et orales qui ont contribué à ma découverte du monde de la recherche et à la valorisation de mes travaux. Ces trois années passées à l'ONERA, guidées par votre expertise, m'ont offert un éventail complet de compétences aussi bien dans la démarche scientifique qu'en algorithmie. Plus particulièrement, merci Mathieu pour ton impressionnante réactivité, ton assiduité et ta rigueur scientifique. Merci Romain pour tes idées créatives qui ont amené un point de vue toujours innovant à cette thèse. Enfin, merci Arnault pour ta bienveillance car, au-delà même de tes qualités scientifiques, tu as été un réel soutien moral dans les moments difficiles.

Ces trois ans n'auraient pas autant participé à mon épanouissement personnel sans la présence de la formidable équipe de doctorants du DTIS ! Merci particulièrement aux doctorants (trop vite envolés) de 2 et 3ème années Camille, Ali, Enzo, Esteban, Clément... pour votre accueil chaleureux à l'ONERA, entre deux confinements. Je souhaite ensuite remercier les doctorants arrivés en même temps que moi dans cette glorieuse aventure dont nous sommes tous sortis vivants (ouf !) : Hanae, on a pu partager 3 ans et demi (et même un bureau !) à l'ONERA ainsi que d'innombrables débats. Mathieu et Clara, on se connaît depuis 7 ans maintenant et je suis ravie d'avoir pu apprendre à beaucoup mieux vous connaître durant cette période ! Merci aussi à tous les doctorants qui ont rejoint le navire au long de ces 3 dernières années : Amina, Antoine, Bastien, Enzo, Jules, Lucas, Matthieu, Pelin, Périclès, Renato, Romain, Shon, Thibaud, Thomas et Wilfried. Merci à tous pour les joyeux moments partagés lors des Cafés de l'Espace, des cafés tout court et des traditionnels Sham ou Deer & Beer ! Je souhaite aussi remercier Thomas et Julie, doctorants à l'INRIA. On s'est finalement assez peu croisés mais je vous remercie pour votre gentillesse et votre soutien lors de nos échanges.

Mes prochains remerciements vont à mes amis qui ont tous, de près ou de plus loin, chacun à leur manière, contribué à égayer mon quotidien et participé à mon équilibre. Merci Emma et Paula pour les innombrables sorties vélos, sillonner la France avec mes deux cyclo-c******es préférées m'a permis de rester "content d'être heureeeeux" !! Je souhaite particulièrement te remercier Emma pour ton écoute et ton soutien infaillible dans les bons comme dans les moins bons moments. Merci aussi à Camille, Rémi, Morgane, qui complètent bien cette équipe, pour tous les gais moments partagés autour d'une (ou généralement plusieurs) bière(s) ! Merci également à Honorine, pour ton soutien et tes conseils lors de mes virées Brestoises ou Roscovites.

Merci aux membres de la Sekonde : Bibiche, Lebas ainsi que Marie et Doui. 7 ans que cette formidable association existe et s'exporte aux 4 coins de la France (que dis-je, du monde !). Merci pour votre soutien, votre bonne humeur, votre humour (toujours au top).

Je remercier également la LMTeam : Aurélia, Daby, Hélo, Claire et Mahmoud à qui j'associerai Francis et Jason, pour les sorties vélo ainsi que les nombreuses "Recyclerie sportive". Merci pour les nombreux et joyeux moments partagés depuis 7 ans.

Durant ces trois dernières années, j'ai aussi pu partager des bouts de quotidien avec mes 3 colocs Soukayna, Joséphine et Samsi ! Merci beaucoup pour vos multiples conseils, votre écoute dans les moments difficiles ainsi que pour le partage de votre expérience. Merci pour les très bons instants que j'ai pu passer durant ces années de cohabitation en votre compagnie.

Merci aussi à mes ami.e.s du Grand Est : Vali, pour ta gentillesse et ton empathie hors du commun, ainsi que Nat et Baptiste. Je garde d'excellents souvenirs de notre cohabitation strasbourgeoise durant la "période covid" entre Mysterium et Flamenkuchens. Au-delà de ce qui nous sépare, merci Diane car tu as beaucoup contribué à mon bien-être durant le début de cette thèse.

Je souhaite désormais remercier ma belle-famille : Véronique, Richard, Dorine, Hermione, Omar, Mathis, Mamie Madeleine... ainsi que toute la ribambelle de petits habitants à plumes, à poils ou à écailles de la Mardelle, qui m'ont bien aidée à écrire ce manuscrit ! Un grand merci pour votre accueil chaleureux, votre générosité et votre bienveillance.

Mes prochains remerciements vont bien sûr à ma famille (toute aussi belle !). Vous avez tous fortement participé à mon épanouissement tant personnel que pro-

fessionnel. Merci à mes grands-parents Mamileine, Papily, Manille et Alain pour avoir contribué à faire de moi l'adulte que je suis. Merci à mes tantes, oncles et cousins Béatrice, Guillaume, Jules, Théo, Loïc, Daniela, Edgar et Thomas, ainsi qu'à ma marraine Corinne et à mes presque-cousins Florian, Loann et Naël. Merci pour tous les moments partagés, très souvent festifs, joyeux, riches d'échanges et de vie !

Je souhaite remercier plus particulièrement mes parents, pour m'avoir toujours poussée à donner le meilleur de moi-même et m'avoir donné les clés pour intégrer la vie professionnelle. Merci immensément pour l'éducation que vous m'avez donnée et tout ce que vous avez fait pour moi. Je remercie aussi chaleureusement ma sœur Pauline et à mon frère Victor pour ces années passées à grandir ensemble. Vous apportez beaucoup de légèreté et de fun dans ma vie, j'espère que cela ne changera pas, il va bien falloir remplir des années de Petit Potin !! Merci aussi à Hugo et Victoria, c'est un plaisir de partager des moments en famille avec vous. Enfin, j'ajouterai que même si ce n'est pas une compétition j'espère que l'on se souviendra que je suis docteure avant chouchou et chouchette !!

Enfin, mes tous derniers remerciements vont à Marion. Au-delà même de cette thèse, depuis nos pérégrinations Pyrénéennes et jusqu'à ces derniers mois pas tous les jours faciles, ta présence, ton affection et ton soutien inconditionnel m'ont donné énormément de force, d'équilibre et de bonheur. Pour ça, pour tout, je te remercie du fond de mon cœur.

Abstract

This thesis falls within the scope of layout optimization, which is an important stage in the design of complex multidisciplinary engineering systems such as aerospace vehicles. Optimal layout problems (OLPs) involve finding the best arrangement of a set of components within a single- or multi-container system or space to meet specific objectives (cost reduction, performance enhancement, *etc.*) while satisfying various constraints (geometrical, functional, *etc.*). Dealing with OLPs is challenging both in terms of their formulation and their efficient and effective resolution. Actually, OLPs are often highly constrained and involve many mixed decision variables (continuous, discrete/categorial) which may be fixed or conditional. Conditional variables are highly useful to define different design choices when the set of components to be arranged is variable and dynamic. Consequently, their resolution requires the use of advanced optimization algorithms combining different classes of (mixed-variable) methods including metaheuristics and Bayesian optimization.

The overall objective of the thesis is to investigate OLPs, their formulation in different contexts, their resolution using various optimization methods and their hybridization, and their validation within the framework of aerospace vehicle design. The contributions of the thesis are organized in two parts corresponding to two types of OLPs. In the first (resp. second) part, the set of components to be arranged is fixed (variable or conditional) involving fixed search space OLPs or FSS-OLPs (resp. conditional search space OLPs or CSS-OLPs). In both cases, the system/space in which the components are arranged is considered single- or multi-container.

In the first part, a survey of constrained mixed-variable FSS-OLPs is proposed including their generic formulations, applications and resolution methods with a particular focus on quasi-physical methods and population-based metaheuristics. Based on a virtual force system (VF) quasi-physical algorithms emulate the principle of physical laws in system dynamics and deal efficiently with highly constrained problems. A variant (namely CSO-VF) of these algorithms is devised for solving single-container FSS-OLPs. In CSO-VF, the positions and orientations of the components are evolved using VF. To deal with multi-container systems, CSO-VF is combined with a Genetic Algorithm (GA) in a two-stage algorithm that assigns the components to the containers and optimizes their layout. These single- and multi-container algorithms are assessed considering satellite module FSS-OLPs that are representative benchmarks.

In the second part, a survey of constrained mixed-variable CSS-OLPs is pro-

posed in the same way than in the first part. Conditional variables involve more complex OLPs. Actually, for instance, in the context of aerospace concept design, a given amount of fuel could be included in a container in either one large tank or two smaller ones. Therefore, as the number of components to position is not the same in both cases the number of design variables as well as constraint functions vary during the optimization process. To deal with single-container CSS-OLPs, two approaches have been investigated: the first one is a GA revisited considering hidden variables, leading to variable-geometry OLPs (in objective and constraint functions). The second approach is a two-stage surrogate guided-CSO-VF algorithm combining Bayesian Optimization with CSO-VF. Bayesian Optimization selects the components with are considered by CSO-VF for layout optimization. This latter approach has been extended with a GA in a three-stage algorithm to tackle multi-container CSS-OLPs. Finally, all the algorithms are evaluated and compared based on their application to CSS variants of satellite module OLPs.

Keywords: Optimal Layout, Conditional Search Space, Quasi-Physical Approach, Bayesian Optimization, Hybridization.

Résumé

Cette thèse s'inscrit dans le cadre de l'optimisation d'agencement, une étape importante dans la conception de systèmes multidisciplinaires complexes tels que les véhicules aérospatiaux. Les problèmes d'agencement optimal (OLPs) consistent à trouver la meilleure disposition d'un ensemble de composants dans un système ou un espace, afin d'atteindre certains objectifs (réduction des coûts, amélioration des performances, etc.) tout en satisfaisant diverses contraintes (géométriques, fonctionnelles, etc.). Le traitement des OLPs est encore un défi aujourd'hui, tant en termes de formulation que de résolution. En effet, les OLPs sont souvent très contraints et impliquent de nombreuses variables d'optimisation (continues, discrètes, catégorielles), qui peuvent être fixes ou conditionnelles. Les variables conditionnelles sont utiles pour définir différents choix de conception qui doivent être faits en même temps que l'optimisation de l'agencement des composants. Ainsi, la résolution des OLPs nécessite l'utilisation d'algorithmes d'optimisation avancés combinant différentes catégories de méthodes, comme par exemple les métaheuristiques et l'optimisation Bayésienne.

L'objectif global de la thèse est d'étudier les OLPs, leur formulation dans différents contextes, leur résolution à l'aide de diverses méthodes d'optimisation et hybridations, ainsi que la validation de ces méthodes dans le cadre de la conception de véhicules aérospatiaux. Les contributions de la thèse sont organisées en deux parties correspondant à deux types d'OLPs. Dans la première (respectivement deuxième) partie, la liste de composants à agencer est fixe (resp. variable), impliquant des OLPs à espace de recherche fixe ou FSS-OLPs, (resp. des OLPs à espace de recherche conditionnel ou CSS-OLPs). Dans les deux cas, le système/l'espace dans lequel les composants sont agencés est considéré comme mono ou multi-contenant. Dans la première partie, une étude des FSS-OLPs est proposée, incluant leurs formulations génériques, leurs applications et méthodes de résolution, avec un focus particulier sur les méthodes quasi-physiques et les métaheuristiques. Basés sur un système de force virtuelle (VF), les algorithmes quasi-physiques simulent les lois de la dynamique et traitent efficacement les problèmes fortement contraints. Une variante (nommée CSO-VF) de de ces algorithmes est developpée afin de résoudre les FSS-OLPs à un seul contenant. Dans le CSO-VF, la position et l'orientation des composants évoluent grâce au VF. Pour traiter les systèmes multi-contenants, le CSO-VF est hybridé à un algorithme génétique (GA) dans un algorithme à deux étages qui assigne les composants aux contenants puis optimise leur disposition dans chacun des contenants. Ces deux algorithmes sont évalués grâce à des problèmes d'agencement de satellites.

Dans la deuxième partie, une étude des CSS-OLPs est proposée de la même manière que dans la première partie. Les variables conditionnelles engendrent des OLPs plus complexes. Par exemple, dans le contexte de la conception aérospatiale, une quantité donnée de carburant peut être incluse dans le système, soit dans un grand réservoir, soit dans deux plus petits. Par conséquent, le nombre de composants à positionner n'est pas le même dans les deux cas et le nombre de variables de conception et de contraintes varient donc au cours du processus d'optimisation. Deux approches ont été développées pour traiter les CSS-OLPs à un seul contenant : la première est un GA modifié pour introduire des variables cachées dans les chromosomes. La seconde est une approche bi-niveaux combinant optimisation bayésienne et l'algorithme CSO-VF. L'optimisation bayésienne sélectionne les composants et le CSO-VF optimise leur agencement. Cette dernière approche a été hybridée avec un GA dans un algorithme tri-niveaux afin de traiter les CSS-OLPs multi-contenants. Enfin, tous les algorithmes sont évalués et comparés grâce à des problèmes d'agencement de satellites.

Mots clés : Optimisation d'Agencement, Espace de Recherche Conditionnel, Méthodes Quasi-Physiques, Optimisation Bayésienne, Hybridation.

Contents

Chapter 1

Introduction

1.1 Optimal Layout Problems

General context and issues. The creation of complex engineering systems encompasses multiple stages, from initial concept development to the realization of that concept. Each subsequent phase consists in numerous analyses and optimizations. This enables designers to determine the most suitable design(s) with respect to the system's objective(s) in the limits of the accuracy of the used models. Moreover, in realistic industrial settings, complex systems are often made up of numerous couplings between the several disciplines that are required for their design [Der+08]. For instance, in the field of aerospace engineering, the preliminary sizing frequently integrates the coupling between vast fields such as aerodynamics, propulsion, structure and/or flight mechanics. While a number of those engineering fields are often tightly integrated at the early design stage, the design of the internal layout of future systems is, however, often set aside and is rarely part of a fully integrated design process.

The internal layout refers to the task of finding the most efficient and effective arrangement of internal components, structures, or devices within the given designed space or system [CSY02]. This problem seeks to optimize the spatial organization of these elements so as to achieve specific objectives, such as maximizing capacity, minimizing costs, improving accessibility, or enhancing the overall system performance. In fact, internal layout problems are often solved by hand or by a set of simple heuristic rules (*e.g.*, expert system) able to mimic the cognitive process of experts. By the fact that this process is performed in the last steps of the design procedure, it does not guarantee that an overall optimal solution has been identified at preliminary design stages. Indeed, the arrangement of components often has a first-order impact on the performance of the system (*e.g.*, for flying qualities of an aircraft, power efficiency of an electronic chip, *etc.*) and can thus be critical to the feasibility of a concept. The main concern is thus the automated optimal design of the internal layout of complex systems that makes it compatible with the inclusion in multidisciplinary design processes.

Examples of optimal layout problems. Optimal Layout Problems cover a large spectrum of applications: packing, facility layout, wind farms, power plants, coverage, complex systems layout problems, *etc.* An overview of these various application cases is given in the following paragraphs.

Packing problems [CH06; LRP22; He+13] are a class of operational research and combinatorial optimization problems which involve arranging a set of items or objects within one or several container(s), such as bins, boxes, freight holds, *etc.* The objective is to find a packing arrangement that optimizes a number of user-defined criteria, which may include maximizing space utilization, minimizing wasted space, and/or satisfying specific constraints like weight limits or item orientations. These problems have diverse applications like for instance logistics, manufacturing, transportation in which packing problems have practical implications in optimizing resource allocation, minimizing costs, and enhancing the utilization of available space. In an operational research context, they come in various forms, including the bin packing problem, the knapsack problem, and the cutting stock problem. Figure 1.1 illustrates some existing packing problems.

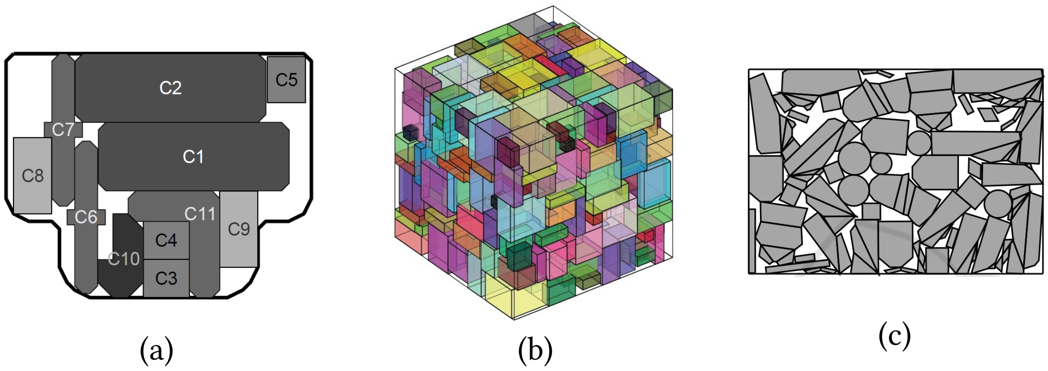
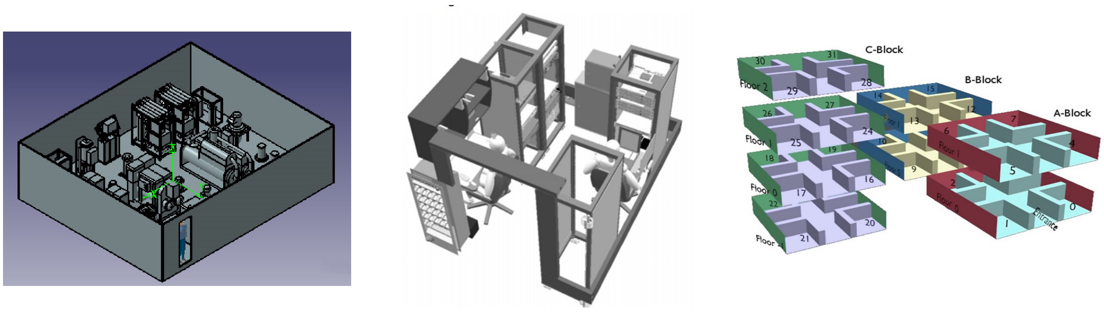


Figure 1.1: Examples of packing problems: (a) car trunk loading [Jac10], (b) container loading [Jac10], (c) strip packing for cutting stock problems [PAO18].

Facility layout problems [ASE12; Has+17; Pen+18] involve optimizing the layout of facilities or resources within a given space to enhance efficiency and reduce operational costs. These problems arise in a wide range of contexts, including manufacturing plants, warehouses, healthcare facilities, offices, and service centers. The aim is to design layouts that streamline operations, improve work processes and, ultimately, increase plant performance, making them indispensable to the various fields seeking operational efficiency. Figure 1.2 illustrates existing facility layout problems.

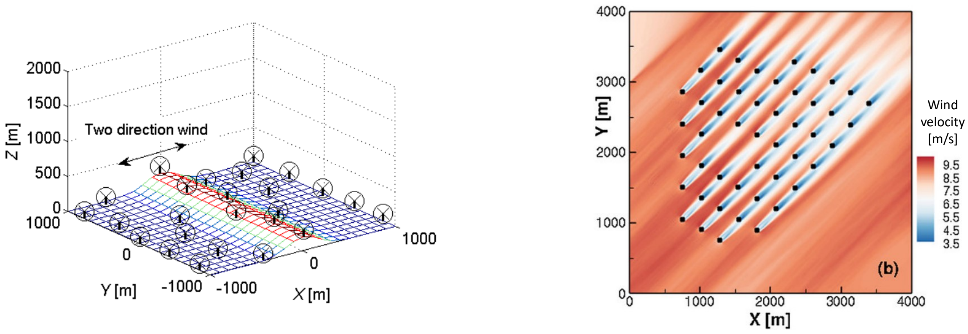
Wind farms layout problems [FS14; GA16; Wan+17] refer to optimization problems in the field of renewable energy. These problems involve determining the optimal arrangement and placement of wind turbines within a designated area, often subject to geographical and environmental constraints. The aim is to maximize energy production and minimize costs. Key objectives include maximizing energy capture by positioning turbines in areas with favorable wind conditions, minimizing wake effects that can reduce efficiency, and considering factors like land use, environmental impact, and infrastructure costs. Figure 1.3 illustrates wind farm layout



(a) (b) (c)

Figure 1.2: Examples of facility layout problems: (a) boat cabin equipment layout [MSK21], (b) shelter layout [Ben+11], (c) hospital layout [THU20].

problems.

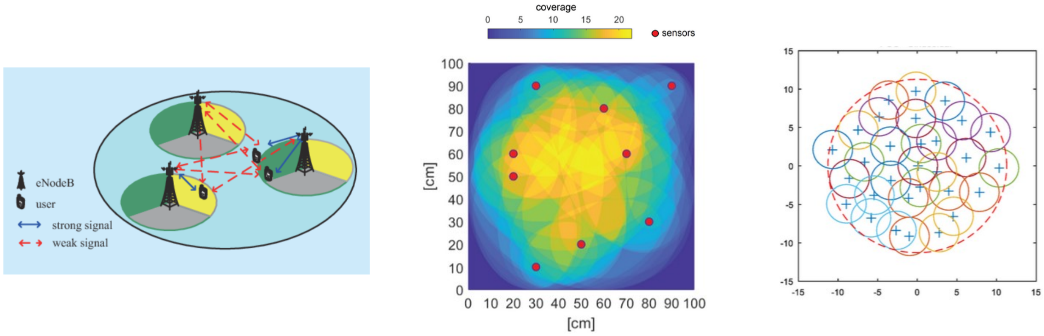


(a) (b)

Figure 1.3: Examples of wind farms layout problems: (a) [FS14], (b) [GA16].

Coverage problems [Pha+17; Som+19; Zha+22b] involve determining how to efficiently cover or reach certain specified locations, regions, or points of interest within a given area. The primary objective is to optimally deploy a limited set of resources or sensors to ensure coverage while minimizing redundancy and cost. These problems have applications in various fields, including telecommunications, sensor networks, environmental monitoring, surveillance, *etc.* Coverage problems can take on different forms, such as area coverage, target coverage, or barrier coverage, depending on the specific context and objectives. Figure 1.4 illustrates some existing coverage problems.

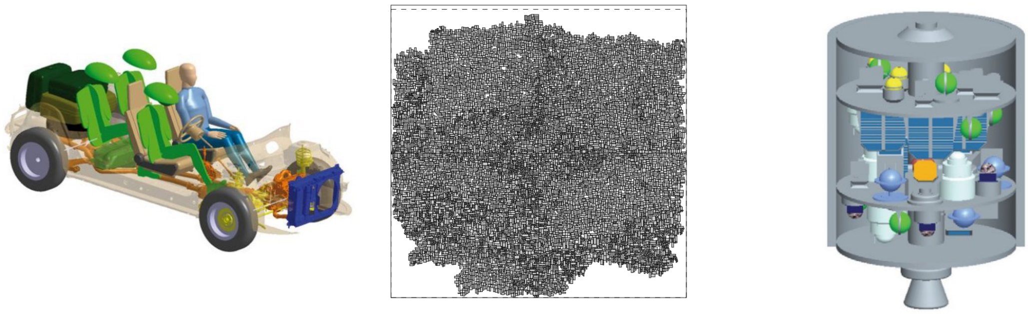
Finally, in the field of complex systems design, internal optimal layout problems [MTK96; ST03; XXA07c] consist in optimizing the arrangement of components (*e.g.*, various equipment, structure elements, electronics) within the internal structure or physical space of a complex system. These systems can encompass a wide range of domains, including but not limited to manufacturing plants, automotive vehicles, aircraft, buildings, and electronic devices. The primary goal of addressing internal



(a) (b) (c)

Figure 1.4: Examples of coverage problems: (a) telecommunication antenna coverage [Pha+17], (b) sensor coverage for damage detection [Som+19], (c) wireless sensor networks coverage [Zha+22b].

layout problems is to achieve an arrangement that optimizes various criteria, such as space utilization, operational efficiency, accessibility, and workflow. Figure 1.5 illustrates existing complex systems layout problems.



(a) (b) (c)

Figure 1.5: Examples of complex systems layout problems: (a) vehicle layout problem [Sta+13], (b) chip layout problem [Ege03], (c) satellite layout problem [YH09].

Definitions of optimal layout problems. Optimal layout problems are defined in terms of objective and constraint functions and the aim of the optimization process is to find the values of the design variables that lead to the best value of the objective function(s) while satisfying all of the constraint(s) [CSY02]. The layout may be solved in 2-dimensions or in 3-dimensions and the components can either be identical (homogenous shapes) or of dissimilar geometries (heterogenous shapes). Single-container or multi-container configurations can be addressed, and the container(s) might not have predetermined dimensions.

The design variables of a generic optimal layout problem are the positions and orientations of the components to be incorporated into the system and and thus they could either be continuous or discrete. For example, the positions of the components can be continuous and the orientation may be restricted to a set of predetermined values (0°,90°) [CZa19; ST03]. In case of the multi-container configuration, other design variables must be added in order to assign the components to a container.

optimal layout problems are often characterized by one or several objective function(s) depending on the system at hand. For example, most packing problems involve minimizing the occupied volume of the container [He+13], whereas facility layout problems often minimize total handling costs [ASE12], and the design of complex systems mainly involves optimizing certain performance metrics or mechanical characteristics, such as maximizing power production or minimizing overall inertia [WTS09b].

Finally, some specifications regarding the design of the system may be converted to constraints [SJ00]. For example, the most obvious specification might be that a feasible optimal layout must not contain any overlapping components [CSY02]. Another constraint may be based on the radiation level within an aerospace systems [CZa19]. Depending on the complexity of the considered system, several dozens or even hundreds of constraints can be added to the problem definition in order to account for all of the design requirements, making the optimal layout problem at hand highly-constrained. The layout resolution can be considered as dynamic or static depending whether or not the time variable is taken into account in the objective and constraint functions. The problem at hand is referred to as **constrained mixed-variable optimal layout problem**.

Short overview of existing methods. The first range of techniques are exact methods which guarantee the optimality of the solution. Among them, the CPLEX solver [BBL14], the Branch and Bound algorithm [ChL13; KYS08; XS08] and Dynamic Programming [Cin+08; CH06; FLS17] have been used to solve Quadratic Assignment Problem [KB57] and Mixed Integer Linear Programming [Mon90] formulations.

Heuristics have then been developed in order to provide good quality results in a more reasonable computational time in comparison to exact methods. They mainly consist in construction [BHS20; CE91; Des+16] and improvement-based algorithms [BME94; LKM17; PRR14].

Over the past 20 years, the most used methods for optimal layout problems are the metaheuristic algorithms. They are problem-independent techniques which aim at exploring more thoroughly the search space than previously used heuristic methods and thus often provide better solutions. Among them, Simulated Annealing [BG01; BR84; MB96; Tam92a], Tabu Search [CCC14; KM97; LC08], Genetic Algorithm [Bor06; Pen+18; ST03; YK13], Particle Swarm Optimization [AWT16; Liu+18; XXA07b], Covariance Matrix Adaptation-Evolution Strategy [AWT16; Wag+11], *etc*, have been used to solve various forms of optimal layout problems. Those techniques have also been hybridized between themselves [LL02; LC12b; LPK17] or with heuristic methods [Che+18; XXA07c] in order to take advantage of the strengths of different algorithms and improve their overall computational performance.

Another range of techniques are quasi-physical methods which are techniques having roots in both physics and mathematics and which aim at mimicking the laws of physics. Most of the time, they correspond to energy [He+18; HY11a] or force-based algorithms [He+13; HY11b; Wan+02].

Finally, recently, machine learning methods have been applied to layout optimization problems even if they remain less investigated as of now [BWH21]. For instance, Bayesian Optimization [SL22], Neural Networks [TBT96] or Reinforcement Learning [Vas+20] have been used to solve optimal layout problems.

Figure 1.6 sums up the different characteristics of optimal layout problems as well as the different application fields and the existing resolution approaches.

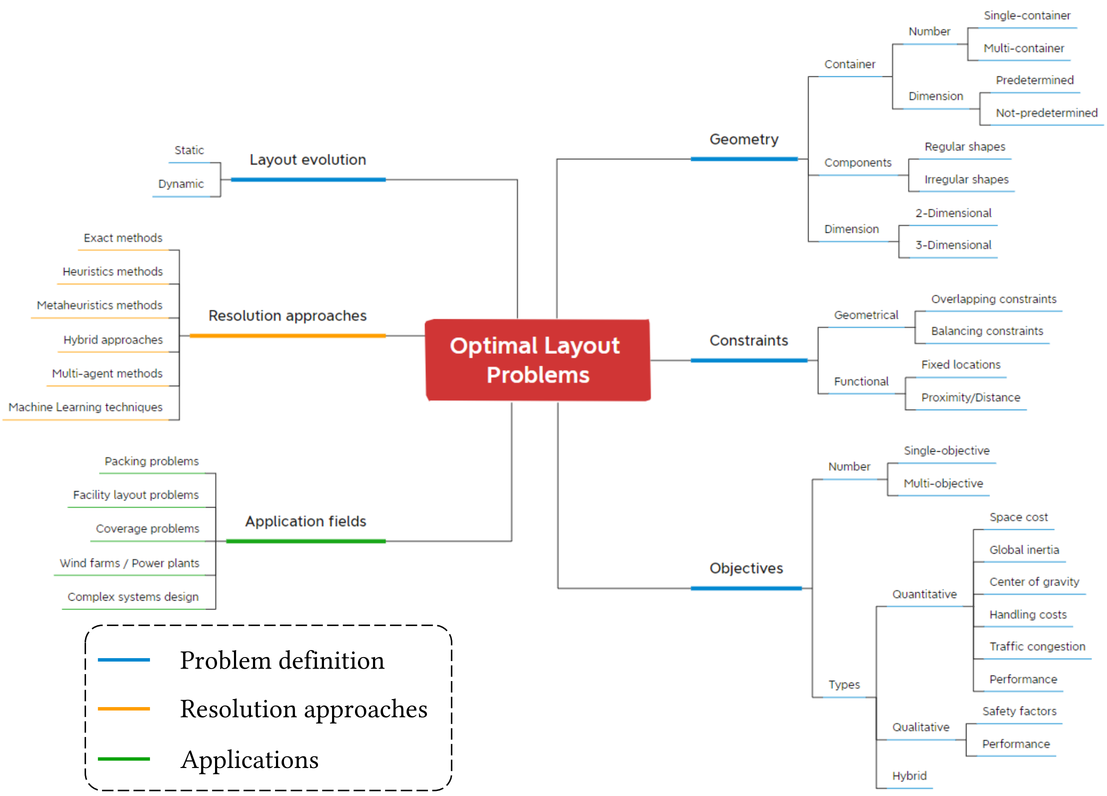


Figure 1.6: Taxonomy of optimal layout problems.

1.2 Conditional Search Space Problems

General context. As introduced in the previous section, the set of components to position within a container is usually defined during the design process stages that precede the optimal layout process. Consequently, the number of design variables related to each of the components (and associated constraints) remain fixed during the whole optimization process. The typology of the optimization problem (*i.e.*, design variables, constraints, objective function) is completely defined before the resolution and does not evolve. This type of problems is referred to as **fixed search space optimal layout problems** or fixed-size design space optimal layout problems [Abd13].

In the early phases of the design process of most real-world engineering systems (*e.g.*, complex systems like aerospace vehicles but also electronic devices, facility dispersion or even settings of complex algorithms), some architecture choices can not be made beforehand and must be defined together with the optimal layout process. For example, in the context of the decarboniation of aeronautical systems, new configurations such as hydrogen powered aircraft are under investigation. For these aircrafts, due to the lower density of hydrogen, the layout of tanks can be deemed very challenging and subject to a large number of options: in the fuselage, in the wings, etc. In addition to their layout within the system itself, designers must take decisions regarding the number and sizes of these aircraft's tanks [Sil+19]. Another example is the optimal layout of wind farms which often consists in maximizing the power produced by wind turbines positioned in the farm [Sam13]. The number of wind turbines has thus a strong influence on the power produced and can be considered as a variable in the optimization process [Gon+11].

Considering a variable number of items to position will make the number of design variables vary from one subproblem of the optimal layout problem to another, as well as the number and type of constraints. Therefore, the most naive approach could be to optimize the layout of the system for each of the possible architecture definitions [Fra+18]. Considering the wind farm optimal layout problem, this approach would consist in optimizing the layout of the wind farms for all possible numbers of wind turbines and choose the best one in terms of their power production. However, depending on the complexity of the system at hand, several thousands of architecture definitions can be considered and this approach may be very time consuming and in some cases unfeasible. Consequently, the layout must be optimized simultaneously with some of the design variables which define it, and the number of components as well as constraints may vary during the optimization process. Allowing the number of components to vary during the optimization may provide a better solution at the end. The choices related to the architecture definition (*i.e.*, the choice of the components to be integrated to the system) can be represented with categorical or qualitative variables in the optimization problem formulation. The values of these additional variables influence the number and types of continuous and discrete variables defining the layout as well as the number and type of the constraint functions. These variables are named conditional variables [Swe+14]. Therefore, the resulting optimal layout problem is referred to as a **conditional search space problem** [LPS05; NAO15]. It is also sometimes referred to as variable-size design space problem [GA11].

Examples of conditional search space problems. Conditional search space problems encompass several application cases: Full Model Solving, optimal trajectory problems, launch vehicle design, as well as some optimal layout problems. Some details are given for each of them in the following paragraphs.

Full Model Solving, also known as Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem [Tho+13], refers to the task of jointly determining the best machine learning algorithm and its corresponding hyperparameter settings for a given dataset and learning tasks. It involves simultaneously selecting the most suitable algorithms from a set of candidates and optimizing the hyperparameters of the chosen algorithms. It belongs to the CSS class of problems because the settings of the hyperparameters directly rely on the choices of the algorithm.

The problem of optimal design of a multi-gravity-assist space trajectory consists in minimizing the trajectory cost of a space mission for a spacecraft to travel from the departure planet to the target planet [GA11]. The spacecraft benefits from as many as needed swing-bys of other planets as well as deep space maneuvers (DSMs). This problem belongs to the category of conditional search space optimization problems as the design variables related to the trajectory (*e.g.*, number of DSM, departure and arrival dates, flight directions, *etc*) depends on the number of swing-bys. Figure 1.7 illustrates a multi-gravity-assist space trajectory from Earth to Saturn with 3 swing-bys (Mercury, Mars and Jupiter) and four DSMs.

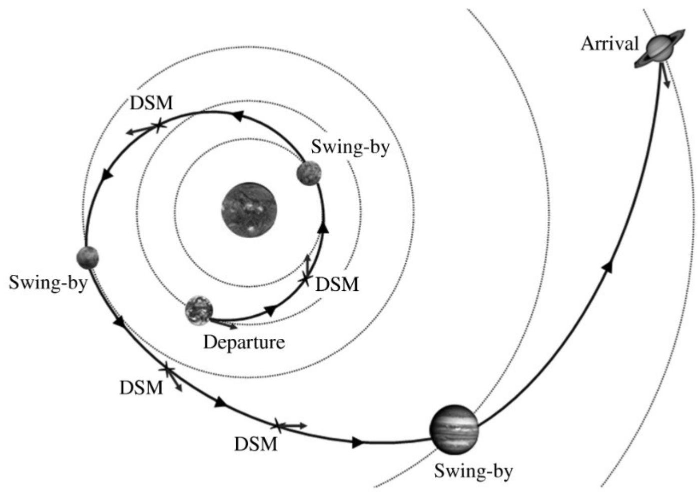


Figure 1.7: Illustration Trajectory mission [GA11].

In the field of aerospace concepts design, the multi-stage launch vehicle architecture optimization problem [Pel+21] requires to simultaneously determine the optimal number of stages characterizing the system (*e.g.*, 2 or 3) and determine the most suitable type of propulsion (*e.g.*, solid or liquid) for each stage in order to minimize the Gross Lift-Off Weight. Given that each propulsive alternative is characterized by different continuous and discrete design variables as well as different constraints, the resulting optimization problem presents a conditional search space. Figure 1.8 represents the possible launch vehicle architectures of the CSS multi-stage launch vehicle architecture problem [Pel+21].

Some conditional search space optimal layout problems have also been defined. Among them, conditional search space wind farms layout problems consist in optimizing both the number and locations of the turbines on the given domain [Rye+17].

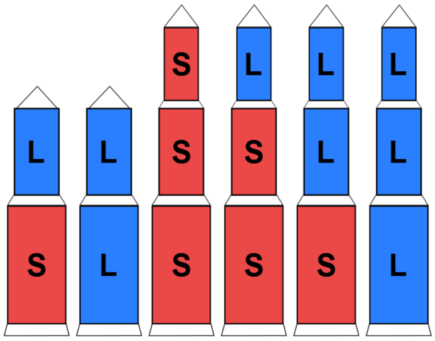


Figure 1.8: Possible launch vehicle architectures (S: Solid propulsion, L: Liquid propulsion) [Pel+21].

In the same way, conditional search space wireless sensor coverage problems optimizes both the number of sensors and their positions in the covered area [Rye+17]. The two aforementioned problems belong to the category of conditional search space optimal layout problems because the number and types of layout variables to optimize (*i.e.*, the coordinates of the turbines and sensors) as well as the constraint depends on the single conditional variable that specifies how many turbines or sensors are positioned in the given area.

Short overview of existing methods for conditional search space problems. Dealing with conditional search space (CSS) problems gives rise to additional challenges notably because of the introduction of conditional variables and the ensuing conditional complexity (*i.e.*, the number of subproblems linked to the architecture choices defined by the conditional variables). Then, dedicated methods have been developed to handle them.

First of all, the most direct method is the exact exploration of the conditional search space. The subproblems of the CSS problem at hand which correspond to each combination of the conditional variables are then optimized independently [AHD23; Fra+18].

The previous techniques can become inefficient or even unfeasible for complex CSS problems characterized either by a large number of combinations of conditional variables or by computationally expensive functions. Thus, other methods have been developed. Among them, direct search algorithms allow to explore such subproblems' space thanks to mesh-based or pattern search frameworks [AAD07; AD01; AD06; KAD01].

Moreover, metaheuristic methods have been extended in order to take advantage from their global exploration abilities as well as the variety of generic frameworks that can be used. Most of these techniques differ from classical metaheuristic method in the implementation of the chromosome (or list of design variables). Among them, variable-length evolutionary algorithms [Rye+17], dynamic-size multiple populations genetic algorithms [AG12], structured genetic algorithm [NAO15] and hidden-genes genetic algorithm or differential evolution [Abd13; GA11] have been developed.

More recently, Bayesian Optimization using Gaussian Processes has been extended for CSS optimization problems [AHD23; HO13a; Pel+21; ZH18b].

Finally, CSS problems have also been formulated as two nested optimization problems which can be solved thanks to bilevel approaches [Pel20; SMD17]. Those methods often involve two dedicated methods at each level which explore the conditional search space at the upper level and the fixed search space of the subproblems at the lower level.

1.3 Objectives and contributions of the thesis

As shown in the previous section, although conditional search space problems encompass quite a large number of application cases, they have rarely been extended to optimal layout problems. Furthermore, the existing conditional search space problems are defined with a limited number of conditional variables, resulting in a handful of subproblems (or combinations of conditional variables) [Rye+17]. Likewise, in the aforementioned conditional search space wind farms and sensor coverage optimal layout problems, the items to position (turbines or sensors) are identical and therefore, a single conditional variable, which corresponds to the number of considered components to position, is required [Rye+17]. However, in real-world applications, optimal layout problems often involve different types of components. For instance, in the design of a vehicle, two requirements might be to include a certain volume of fuel as well as to provide a certain battery power. Therefore, the fuel requirement could be achieved by placing either a single large fuel tank inside the vehicle or by using two smaller tanks derived from the original one. In the same way, different sizes and types of batteries can be integrated to the system to meet the battery power requirement. Consequently, one conditional variable is needed for each different type of components. The primary focus of this thesis is thus to develop algorithms dealing with conditional search space optimal layout problems.

In order to address conditional search space optimal layout problems, this thesis first aims at solving fixed search space optimal layout problems which correspond to the subproblems of the main conditional search space optimal layout problems. Also, problems involving single-container and multi-container configurations are addressed. Subsequently, conditional search space optimal layout problems with single and multi-container configurations are defined and addressed.

Thesis objective

This thesis aims to develop algorithms for fixed search space and conditional search space optimal layout problems, with single or multi-container configurations and heterogenous types of components.

Toward this goal, five key contributions were made:

- **Design and implementation of a Component Swarm Optimization algorithm based on a Virtual-force System (CSO-VF) for single-container fixed search space optimal layout problems.** This algorithm framework was developed to ensure that the fixed search space subproblems of the conditional search space of the main problem are dealt with properly in terms of constraints handling, convergence accuracy, robustness, *etc.*

- **Design and implementation of a two-stage approach combining genetic algorithm and the CSO-VF algorithm for multi-container fixed search space optimal layout problems.** This algorithm was developed in order to enhance the CSO-VF algorithm framework with the ability to deal with multi-container configurations.

- **Design and implementation of a hidden-variable mechanism for genetic algorithm for solving the conditional search space optimal layout problems at hand.** As the metaheuristic algorithms are the most employed methods in the literature for this typology of problems, an extension to hidden genes genetic algorithms is proposed to handle the problems tackled in the present work.

- **Design and implementation of a bilevel approach combining Bayesian Optimization and the aforementioned CSO-VF algorithm for solving single-container conditional search space optimal layout problems.** This algorithm framework has been developed in order to improve the exploration of the conditional search space and the constraint handling aspects in comparison to the previous metaheuristic approach.

- **Design and implementation of a tri-level approach combining bayesian optimization, genetic algorithm and the CSO-VF algorithm for multi-container conditional search space optimal layout problems.** This last algorithm is developed in order to enhance the previous bilevel approach and include multi-container configurations.

In this thesis, satellite layout problems are employed as a case study to evaluate the efficiency of each algorithm. While this serves as a focused context, one should note that the utility of those algorithms may be relevant to a much wider range of problems as discussed earlier. The simplified model of the telecommunication satellite INTELSAT-III described for instance in [Liu+18; WTS09b; ZTS08] is used as a performance benchmark. Indeed, it corresponds to a representative optimal layout problem which has been widely tackled in the literature.

The structure of this manuscript is organized in two parts. The first part is devoted to fixed search space (FSS) optimal layout problems and is made up of three chapters. Chapter 2 provides an overview of the existing methods for solving FSS optimal layout problems, with a focus on the methods identified as promising. Chapter 3 describes the proposed CSO-VF algorithm for single-container FSS optimal layout problems as well as the two-stage approach combining genetic algorithm and the CSO-VF algorithm for multi-container configurations. The two algorithms are evaluated in Chapter 4 with applications to satellite module optimal layout problems.

The second part of this manuscript is dedicated to conditional search space (CSS) optimal layout problems and is composed of four chapters. Chapter 5 provides an overview of the existing methods for solving CSS optimal layout problems, with a focus on the methods identified as promising in the context of this work. Chapter 6 describes both algorithms developed to handle CSS single-container optimal layout problems: the hidden-variables for genetic algorithm and the bilevel approach combining Bayesian Optimization and the CSO-VF algorithm. Both algorithms are evaluated in Chapter 7 based on their application to CSS satellite module optimal layout problems. Finally, Chapter 8 is devoted to the development and assessment of the tri-level approach combining Bayesian Optimization, Genetic Algorithm and the CSO-VF algorithm in order to deal with CSS multi-container satellite module layout problems.

Figure 1.9 represents the thesis structure.

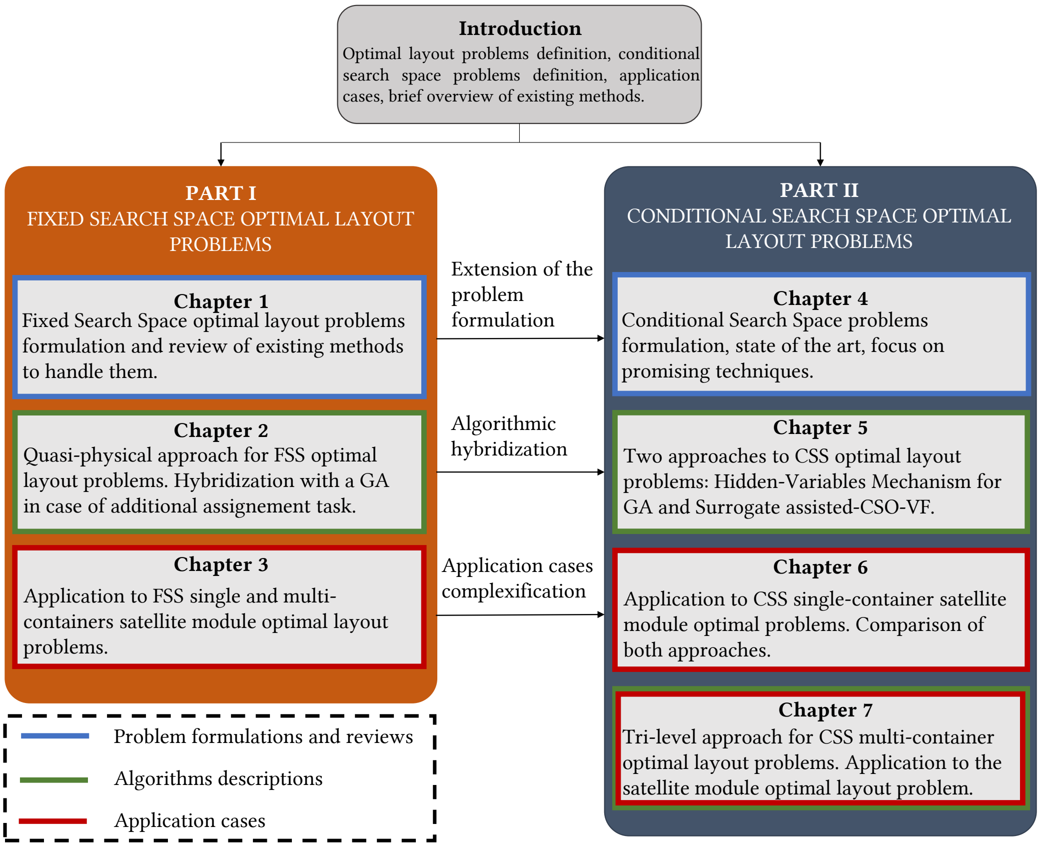


Figure 1.9: Thesis structure.

Part I

Fixed Search Space Optimal Layout Problems

Chapter 2

Fixed Search Space Optimal Layout Problems Statements and Related Works

Contents

Chapter goals

- Mathematical formulation of constrained mixed-variable fixed search space optimal layout problems.
- Survey of existing approaches to handle aforementioned problems.
- Extensive description of the most promising methods.

2.1 Introduction

The first part of this thesis is devoted to fixed search space optimal layout problems (OLPs). This chapter of literature-review serves as an introduction to the generic formulation of OLPs and to the existing methods to handle them. More precisely, the mathematical definition of constrained mixed-variable fixed search space OLPs is presented, the issues related to the defined problems are raised and discussed, and the existing techniques developed to handle them are surveyed. In the first section, fixed search space OLPs are defined. A general overview of the existing methods adapted to handle them is established in the second section which encompass exact, heuristic, metaheuristic, hybrid, quasi-physical and machine learning approaches. The third section is devoted to the techniques identified as promising and which will be considered throughout this thesis.

2.2 Formulation of fixed search space optimal layout problems

In this section, fixed search space (FSS) OLPs are detailed and mathematically defined.

2.2.1 General problem definition

In general, single-objective constrained mixed-variable FSS optimization problems are mathematically defined as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z}} \quad & f_{obj}(\mathbf{x},\mathbf{z}) \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x}, \mathbf{z} \in \prod_{d=1}^{n_z} F_{z_d} \\
\text{s.t.} \quad & \mathbf{h}(\mathbf{x},\mathbf{z}) = 0 \\
& \mathbf{g}(\mathbf{x},\mathbf{z}) \leq 0
\end{aligned}
\tag{2.1}
$$

where:

- f_{obj} corresponds to the objective function to minimize;
- $\mathbf{x}$ corresponds to the vector of n_x continuous design variables;
- $\mathbf{z}$ corresponds to the vector of n_z discrete or categorical design variables;
- $\mathbf{h}$ stands for the equality constraints;
- $\mathbf{g}$ stands for the inequality constraints.

The design variables, objective function and constraints are specified in the case of OLPs in the following subsections.

2.2.2 Design variables

In the case of generic OLPs, the design variables are detailed as follows:

- **Continuous variables: x**
  They are real numbers defined within given intervals. They can correspond to the centers of inertia and angular orientations of the components.

- **Discrete variables: z**
  Discrete variables can be quantitative *i.e.*, integers defined within given intervals, or qualitative *i.e.*, non-relaxable variables defined within a given set of choices. They are also referred to as categorical variables. In this thesis, quantitative and qualitative discrete variables are grouped together as discrete variables. For specific problems, due to engineering constraints, one or several coordinates of the centers of inertia of the components as well as their orientations can be considered as discrete variables [Jac+09; ZTS08]. In case of multi-container problems, discrete variables can also correspond to the number of the container in which each component is positioned [CZa19].

2.2.3 Objective function

Several objective functions can be considered and some of them are listed below:

- Minimize the occupied volume of the components in the container, for example in packing problems [Bor06; He+13; LRP22];
- Minimize the global inertia of the system, for example in complex systems design [CZa19; ST03; ZTS08];
- Minimize the distance between the center of gravity of the system and its geometrical center, for example in packing problems or complex systems design [Jac+09];
- Minimize or maximize any system performance as for instance power emission or consumption for example in wind farms, power plants or containers terminal OLPs [Hou+19; Wan+19a; Wei+10];
- Minimize or maximize any cost related to the system layout as for instance handling or manufacturing costs, for example in facility layout [MB96; Rip+13; SS06].

2.2.4 Constraints

Equality and inequality constraints **h** and **g** can be split into two categories:

- **Geometrical constraints**
  These constraints traduce geometric specifications on the layout which only depend on geometric and masses considerations. For instance, no overlap between the components is allowed [CSY02], the components must fully belong to the container [CSY02], the global center of mass must be placed at the geometrical center of the container also referred to as balancing or equilibrium constraints [He+13], *etc.*
- **Functional constraints**
  These constraints do not only depend on geometric consideration of the layout, but also on system-level specification (*e.g.*, incompatibility between two types of components - fuel and power, radiative level on the container in several zones, *etc.*). A variety of functional constraints can also be added to the problem formulation *via* the use of special components that induce further restrictions on the feasibility of the layout. It includes for instance proximity or minimal distance requirements between components [CZa19].

2.3 Overview of methods for fixed search space optimal layout problems

This section aims to review the main algorithms that are proposed in the literature in order to solve the aforementioned OLPs. They are classified into different categories:

exact methods, heuristic methods, metaheuristics, hybrid approaches, quasi-physical methods and machine learning techniques.

2.3.1 Exact methods

The exact methods guarantee the optimality of the solution. In that respect, many exact methods have been deployed in order to solve the mathematical models developed to define several OLPs.

The very first approach to layout problems is the Quadratic Assignment Problem (QAP) model first introduced in [KB57]. QAP is developed in order to find the optimal positions of equal-area items (*i.e.*, components, facilities, departments, *etc.*) on discrete locations. Many facility layout problems have been formulated as QAP models [KH87; Zho+17]. A more recent and alternative way to formulate layout problems is the Mixed Integer Linear Programming (MILP) first introduced in [Mon90]. The MILP model uses a continuous/discrete representation and is thus more realistic and accurate than the QAP model. It has been mainly explored in the field of facility layout problems as in [BMN02; Geo+99; PP02] where the facility layout MILP models are solved thanks to dedicated solvers as the General Algebraic Modeling System (GAMS) [BM04] and the CPLEX optimization package [BBL14]. Branch and Bounds (B&B) resolution methods have also been used in order to solve the aforementioned mathematical formulations and successfully applied to circular packing problems as well as multi-floor facilities layout problems [ChL13; KYS08; XS08]. Finally, Dynamic Programming (DP) techniques have been used in order to solve cutting or packing problems [Cin+08; CH06; FLS17], or designing interconnection networks [SSS19; TDT15].

Even if exact methods aim at systematically finding the optimal solutions, they are computationally expensive and can thus tackle only small-size instances [Fes14]. Moreover, the MILP formulation assumes the linearity of the objective and constraint functions which is not always a valid hypothesis. Finally, the introduction of complex constraints can pose challenges for most of the exact methods. Non-linear or non-convex constraints can induce difficulties to determine bounds and for instance generate subproblems for the B&B, limiting the effectiveness of the algorithms. Thus, with the increase in the complexity and number of items to position, constraints or objectives, most of the real-world OLPs are not solvable using exact methods [Fes14].

2.3.2 Heuristic methods

In order to obtain good quality results in a reasonable computational time in comparison to exact methods, a lot of research focused on heuristic methods. Heuristic algorithms do not provide a guarantee of optimality but are dedicated to the search of good quality solutions.

One of the oldest heuristics is the Construction Method. The principle of construction-type heuristics is to build a single solution from scratch by iteratively selecting and locating the items until the layout is complete. The items are either randomly selected or selected based on heuristic measure. They are then positioned based upon a heuristic scoring scheme related to the items already located in the layout. Two representative construction-based heuristic algorithms are the CORELAP and the ALDEP algorithms (Automated Layout Design Program). In [Lee67], the CORELAP algorithm (Computerized Relationship Planning) is introduced which aim is to construct layout by locating rectangular-shaped departments. It is a deterministic construction adjacency-based algorithm which uses the total closeness rating as a heuristic measure (the sum of the numerical values assigned to the closeness relationships between one item and the others) and which has been widely studied and applied to various cases [HI15; JRS20; Sem+18]. Another similar heuristic algorithm is the ALDEP algorithm introduced in [SE67]. It is also a construction adjacency-based method but including randomness and proposing many layouts instead of a single one. It has been successfully used for various facility layout application cases [BHS20; CE91; Des+16]. Other well-known construction-based algorithms for OLPs have been proposed such as the PLANET algorithm (Plant Layout ANalysis and Evaluation Technique) [VB66], the FLAG algorithms (Facilities Layout Algorithm using Graphics) [KM84] as well as the SHAPE algorithm [HHS86].

The other category of heuristic mechanisms corresponds to the Improvement algorithms which are the most popular heuristics. The principle of such algorithms is to start with an initial feasible layout and then try to improve it with exchanging items. The most widely used improvement-based heuristic procedure is the CRAFT (Computerized Relative Allocation of Facilities Technique) algorithm [AB63]. It computes the items locations and returns an estimation of the total interaction costs for the initial layout. Then, the algorithm chooses pairs of similar or adjacent items to be switched and evaluates the effect of the switch on the global interaction cost. The procedure terminates when no improving switches are available. Many variants and improvements of the CRAFT algorithm have been published over the years and have been successfully applied to many different facility layout problems [Joh82; LKM17; PRR14]. Other improvement-based algorithms have been developed in order to improve the CRAFT procedure that performs well only in case of equal-area facilities layout. Among them, the MULTIPLE algorithm (Multiple-floor Plant Layout Evaluation) [BME94] is based on spacefilling curve representation and allows to exchange items not restricted to rectangular shapes and not only adjacent.

The aforementioned improvement-based algorithms use a discrete representation. Some construction and improvement heuristics have been developed based upon continuous representation. In [KK00], the authors used a mixed integer programming model to define equal-area facility layout problems and then proposed a two-phase algorithm to solve it combining a construction phase and an improvement phase. In [FS22], the authors developed a two-level heuristic called hierarchical hyper-heuristic in order to position irregular polygons into squared or rectangular container. Finally, in [APT08], [DGR04] and [LC12a], greedy randomized adaptive search procedure (GRASP) is applied to several packing problems. GRASP is an iterative procedure combining a constructive phase and an improvement phase. During the constructive phase, the layout is constructed step by step thanks to a greedy function dynamically adapted. In the following improvement phase, a local search is usually implemented in order to substitute some items to produce an overall better solution [FR95].

Table 2.1 sums up the main constructive and improvement-based heuristic algorithms. Those algorithms have later been combined in order to take advantage of the strength of both construction and improvement phases.

Table 2.1: Heuristic algorithms for OLPs.

Reference	Name of the algorithm	Type
[Lee67]	CORELAP	Construction
[SE67]	ALDEP	Construction
[EGH70]	MAT	Construction
[Blo78]	FATE	Construction
[KM84]	FLAG	Construction
[WW84]	CREATE	Construction
[BM85]	FLING	Construction
[Kon85]	PLANET	Construction
[HHS86]	SHAPE	Construction
[KH87]	INLAYT	Construction
[MRG87]	MATCH	Construction
[WC87]	MICROLAY	Construction
[AB63]	CRAFT	Improvement
[Hil63]	H-63	Improvement
[HC66]	HC-66	Improvement
[Vol68]	COL	Improvement
[Kha73]	FRAT	Improvement
[TR76]	COFAD	Improvement
[FR78]	DA	Improvement
[Joh82]	SPACECRAFT	Improvement
[Emm+92]	STORM	Improvement
[BME94]	MULTIPLE	Improvement

Globally, heuristics may lack of global search capacities which lead to convergence to local optimum instead of the global optimum and are often over computationally expensive especially in case of highly dimensional and constrained problems [Des+15; QL16]. Finally, most of the heuristic algorithms highly depend on the problem at hand. Even if they take full advantage of the particularities of the problem, they lack of genericity and versatility. For these reasons, heuristic methods might be inadequate in case of complex search space characterized by a large number of design variables and constraints.

2.3.3 Metaheuristic methods

Metaheuristics are problem-independent techniques which aim at exploring more thoroughly the search space than heuristic methods and thus often provide better solutions. Consequently, during the past 20 years, mostly metaheuristic algorithms have been developed in order to solve OLPs. In [SS06], Singh *et al.* reviewed methods that have been applied to facility OLPs which are mainly metaheuristic techniques. In [Sia16], Siarry detailed the main metaheuristics and some of their applications. In this section, single-solutions metaheuristics are first considered: Simulated Annealing, Tabu Search and Variable Neighborhood Search. Subsequently, population-solutions metaheuristics are introduced: Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, Covariance Matrix Adaptation Evolution Strategy, Ant Colony Optimization, Scatter Search and Quality-Diversity algorithms. All the aformementioned metaheuritic approaches have been used to solve OLPs. Thus, in the following sections, brief descriptions of metaheuristics are provided with emphasis on the application to layout problems. The metaheuristics that will be used in this manuscript are then detailed in Section 2.4.

2.3.3.1 Single-solutions metaheuristics

Simulated Annealing. One of the first metaheuristic techniques that has been developed and applied to OLPs is the Simulated Annealing (SA) algorithm which is inspired by annealing in metallurgy [KJV83]. It iteratively explores the solution space by accepting better solutions and occasionally worse solutions with a decreasing probability determined by a cooling schedule. This allows the algorithm to escape local optima and converge towards better global solutions. One advantage of SA algorithms is that they only rely on one hyperparameter corresponding to the initial temperature and is thus easy to configure. SA algorithms have been adapted and applied to many OLPs [BG01; BR84; MB96; Tam92a]. Even if SA is able to handle mixed-variables problems [ZW93], it is known to have a slow convergence [Dee92; RSS96], which makes it not suitable for highly dimensional and constrained problems which are characterized by a complex search space.

Tabu Search. Other metaheuristic methods have been explored to solve OLPs. Tabu Search (TS) algorithms have been developed for OLPs [Glo86]. TS is a metaheuristic algorithm initially developed for combinatorial optimization dealing with discrete variables. It explores neighboring solutions, evaluates them, and maintains a tabu list to prevent revisiting recent solutions. It balances intensification and diversification by selectively allowing moves that improve the current best solution. TS is known to be relatively fast [CGM21] due to its intrinsic characteristics as well as a good trade-off between intensification and diversification capabilities and has thus been sometimes considered as an alternative to other metaheuristic methods for OLPs [CCC14; KM97; LC08]. In [LC08], the authors developed a TS algorithm enhanced with intensification and diversification procedures for various facility OLPs. Carrabs *et al.* [CCC14] used TS in order to solve the widely studied circular bin packing problem which consists in positioning circular items in the smallest circular container as possible. They enhanced the TS approach with some improvement strategies that simulate the movements of circles in the circumference of the container and try to free some space in the circumference to allow a reorganization of the circles and to produce better solutions.

Variable Neighborhood Search. One last metaheuristic technique reviewed in this manuscript is the Variable Neighborhood Search (VNS) technique which proceeds by performing local changes to improve an initial solution until a local optimum is found [MH97]. The VNS algorithm provides a flexibility in the neighborhood design making it adaptable to different problem types and neighborhood structures. This flexibility enables effective exploration of the search space based on the specific problem characteristics and contributes to its ability to escape local optima by switching between neighborhoods to continue exploring the search space and seek better solutions. Many variants of this algorithm have been developed in order to solve OLPs [CP13; HCD21; Rip+13]. However, exploring multiple neighborhoods and executing multiple iterations can increase the time complexity of the algorithm especially in case of high dimensional problems [Oua+20]. VNS is also very sensitive to the definition of the neighborhood and thus, insufficiently effective local search can limit VNS's ability to find high-quality solutions [TZ19].

2.3.3.2 Population-solutions metaheuristics

Genetic Algorithms. Among all the metaheuristic algorithms, Genetic Algorithms (GAs) received the most attention. GAs are inspired by Darwin's theory of natural evolution and is based on natural selection in order to repeatedly improve a population of solutions [Gol89; Hol92]. GAs are often effective in exploring the entire search space even in case of complex design search space. Moreover, they can be used to solve a wide variety of optimization problems, whether they are continuous, discrete, mixed-variables, unconstrained or constrained, single-objective or multi-objective and have thus been widely used for OLPs, even if studies show that the constraint satisfaction can be a bottleneck for such algorithms [KS05; MC11; Sal09]. In [Tam92b], Tam proposed a coding scheme design for the use of GAs to solve facility layout problems. In [TS95], a GA is applied to the QAP class of problems and provided better results compared to several heuristic counterparts. In [Jac+09], Jacquenot *et al.* proposed a placement algorithm based on a GA enhanced by a separation algorithm in order to facilitate the treatment of overlapping constraints and deal with polygonal items. In general, genetic algorithms have been used to solve various application cases like packing problems [Bor06; Jak96; LRP22; Qui+15], facility layout problems [ASE12; El-04; Pen+18], complex system design [MTK96; ST03; XXA07c], facility dispersion [EN10; MO04; TMX09; YK13], *etc.* As used in Chapter 3 and Chapter 8 of this manuscript, this algorithm will be detailed in Section 2.4.1.1.

Particle Swarm Optimization. Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behavior of bird flocks or fish schools [KE95]. It initializes a swarm of particles (corresponding to a potential solution of the problem at hand) in a search space and updates their velocities and positions based on personal and global bests. Through iterative exploration and interaction, particles converge towards solutions by balancing individual exploration and social cooperation. PSO algorithms are known to be easily implementable and to have often provide good convergence speed thanks to its ability to exploit rapidly promising areas of the search space [EHG05; Esp+11; Has+05]. In [Liu+18], Liu *et al.* described a multi-objective PSO algorithm to address facility layout problems including a mutation heuristic, a local search and an objective space division method. In [XXA07b], a SA and PSO-based algorithms are developed to solve circular bin packing problems with equilibrium constraints and outperformed previous algorithms. However, PSO algorithms are initially proposed for continuous optimization problems and additional challenges arise when dealing with mixed-variables. PSO algorithms have been proposed in case of discrete design variables exclusively [KE97; SAA02; SH06], but more rarely for mixed-variable problems [WZZ21].

Differential Evolution. Storn *et al.* [SP97] introduced Differential Evolution (DE) algorithm which generates trial solutions by perturbing and combining individuals based on the differences between randomly selected individuals. These trial solutions are then compared to the current population, and if they are better, they replace the corresponding individuals. DE has the advantage of relying on very few hyperparameters and have been used and adapted for OLPs. Wang *et al.* [Wan+17] addressed the wind farm layout problems by means of a DE algorithm with a new encoding mechanism considering each wind turbine as an individual in order to reduce the dimension of the search space as well as the number of hyperparameters. Zhao *et al.* [Zha+17] used the DE approach in order to design and position obstacles for panic evacuations to obtain a minimum leaving time for all the pedestrians. Their approach proved to improve the design and layout obtained by human intelligence and provided a good robustness of the results.

Covariance Matrix Adaptation-Evolution Strategy. Hansen [HO01] proposed a Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) which is particularly effective for handling problems with complex or noisy fitness landscapes. CMA-ES generates population according to a multivariate normal law and fits its parameters (mean and covariance matrix) in order to explore the search space in promising regions. The same application case than in [Wan+17] is considered in [Wag+11] and tackled by a CMA-ES to position up to 1000 wind turbines. In [AWT16], improved CMA-ES, GA and PSO algorithms are applied to unequal-area continuous facility layout problems and compared, giving advantage to the CMA-ES algorithm. As CMA-ES is based on continuous probability distribution, handling discrete variables is not inherent to the algorithm and several enhancements have been proposed [Ham+22].

Ant Colony Optimization Dorigo *et al.* [DBS06] developed Ant Colony Optimization (ACO) algorithms to solve optimization problems by simulating the behavior of ants searching for food. ACO uses a population of artificial ants that construct and follow paths through a graph-based problem representation. Each ant probabilistically chooses its next move based on pheromone trails, which represent the quality of the paths. The pheromone levels are updated based on the quality of the solutions found. ACO effectively balances exploration and exploitation, as the pheromone trails guide the ants towards promising regions while allowing exploration of new paths. ACO is originally developed to solve combinatorial problems. It has been adapted for solving continuous problems [She+07] and subsequently, variants have been proposed to handle mixed-variables formulations [Lia+13]. In [Has+17], ACO methods for facility layout problems are classified and their performance are analyzed and compared to other metaheuristic techniques. In [LD04], pure ACO and ACO enhanced with local search are used to solve bin packing and cutting stock problems. They proved that the pure ACO approach can compete with existing evolutionary methods, whereas the ACO-based memetic approach can outperform the best-known hybrid evolutionary solution methods for certain problem classes. Even if ACO has good performance when solving continuous or discrete problems, the convergence speed of the algorithm can slow down and the convergence accuracy might decrease in case of high-dimensional problems [YYL21].

Scatter Search. Scatter Search (SS) combines solutions from diverse regions of the search space to find optimal solutions [LM03]. It starts by generating a diverse set of initial solutions, called the reference set. The algorithm then iteratively explores the search space by combining pairs of solutions from the reference set to create new solutions. The new solutions are evaluated, and a subset of the best solutions is selected to update the reference set. This process continues until a termination condition is met. SS effectively balances exploration and exploitation by maintaining diversity in the reference set and combining promising solutions [MLG06; Glo86]. Kothari *et al.* [KG14] proposed four SS algorithms for the single row facility layout problem using different diversification techniques to generate an initial population containing good quality and diverse permutations. In [WST09], the authors addressed the circular bin packing problem with equilibrium constraints by improving a SS algorithm. A gradient descent algorithm and a simplex algorithm [BT97] are adopted to improve the trial solution generated in the SS framework, enhancing the trade-off between exploration and exploitation.

Quality-diversity algorithms. Recently, a new type of evolutionary algorithms has emerged, qualified as quality-diversity (QD) algorithms [LS11; MC15; PSS16]. These algorithms were designed to tackle complex optimization problems where finding a single optimal solution may not be sufficient or even feasible. Instead, they focus on discovering a set of diverse and high-quality solutions that cover different regions of the problem space. QD algorithms typically employ a two-step approach: first, they generate a diverse set of candidate solutions where diversity is quantified through additional functions called "features", and then they evaluate and refine these solutions based on a predefined quality measure. By promoting both quality and diversity, these algorithms can produce a rich set of solutions that offer alternatives to traditional single-solution optimization approaches. Among QD various approaches, MAP-Elites (Multi-dimensional Archive of Phenotypic Elites) algorithms have been developed and applied to OLPs [Gal+21; NDN22; Zha+23]. MAP-Elites focus on exploring and mapping the diversity of solutions across multiple dimensions by gradually building a map of the problem space, revealing the diversity of solutions that exists across different feature combinations [MC15]. Galanos *et al.* developed a QD algorithm called ARCH-Elites for urban design [Gal+21]. It corresponds to a MAP-Elites implementation that can reconfigure large-scale urban layouts at real-world location. In [NDN22], two MAP-Elites algorithms are derived in order to tackle knapsack problems. Zhang *et al.* also used MAP-Elites algorithms to optimize the warehouse layouts in order to coordinate hundreds of robots in large automated warehouses, with the objective of maximizing the throughput produced by a multi-agent path finding-based robot simulator while diversifying a set of user-defined measures, such as travel distances of the robots and the distribution of the shelves [Zha+23].

Table 2.2 sums up the metaheuristic methods applied to OLPs and their principal characteritics: discrete (D) or continuous (C) variables representation, single (S) or multi-objective (M), the maximum number of items (dimension), the considered constraints: boundary constraints (B), overlapping constraints (O), equilibrium constraints (E), functional constraints (F) and finally the case study: facility layout problems (FLP), quadratic assignment problems (QAP), packing problems (P), complex systems design (CS), wind farms and coverage layout problems (WF/C).

It must be noted that other metaheuristic methods exist and may have also been applied to OLPs like Artificial Bee Colony [SLZ11], Gray Wolf [LRP22] or Harmony Search [KC17] methods. As they are less frequent in the literature, they are not properly reviewed here for summary and clarity purposes.

Table 2.2: Metaheuristic methods for OLPs

Reference	Method	Variables	Objective	Dimension	Constraints	Case study
[BG01]	SA	D	S	30	B	FLP
[BR84]	SA	D	S	36	B	QAP
[Dow93]	SA	C	M	20	B	P
[Tam92a]	SA	D	S	30	B	FLP
[BAZ15]	TS	D	S	30	B	FLP
[CCC14]	TS	D	S	50	B/O	P
[LC08]	TS	D	S	28	B	FLP
[CP13]	VNS	D	S	26	B/F	WF/C
[HCD21]	VNS	D	S	50	B	FLP
[Rip+13]	VNS	D	S	15	B	FLP
[ASE12]	GA	D	M	20	B	FLP
[Bor06]	GA	C	S	5000	B	P
[El-04]	GA	D	S	26	B	FLP
[EN10]	GA	D	S	39	B	WF/C
[Jac+09]	GA	C/D	M	11	B/O	P
[Jak96]	GA	D	S	50	B	P
[LRP22]	GA	C	S	133	B/O	P

Continued on next page

Table 2.2: Metaheuristic methods for OLPs (Continued)

Reference	Method	Variables	Objective	Dimension	Constraints	Case study
[MO04]	GA	D	S	30	B	CS
[Pen+18]	GA	D	S	125	B	FLP
[ST03]	GA	C	S	53	B/O/E	CS
[Tam92b]	GA	D	S	30	B	FLP
[TS95]	GA	D	S	36	B	QAP
[XXA07c]	GA	C	M	55	B/O/E	CS
[YK13]	GA	C	S	100	B	WF/C
[Liu+18]	PSO	C	M	62	B/O	FLP
[XXA07b]	PSO	C	M	55	Overlap	P
[ZL17]	PSO	C	S	20	O/B	CS
[Wan+17]	DE	C	S	100	B/F	WF/C
[WLW22]	DE	C	S	50	B/F	WF/C
[Zha+17]	DE	C	S	3	B	Other
[AWT16]	CMA-ES	C	S	20	B/O	FLP
[Wag+11]	CMA-ES	C	S	1000	B/F	WF/C
[Has+17]	ACO	D	S	20	B	FLP
[CK04]	ACO	D	S	200	B/O	FLP
[LD04]	ACO	D	S	1000	B	P
[KG14]	SS	D	S	80	B	FLP
[WST09]	SS	C	S	40	B/O/E	P
[Gal+21]	QD	C	S	1000	B/F	Other
[NDN22]	QD	C	S	200	B/O	Other
[Zha+23]	QD	C	S	200	B/F	Other

2.3.4 Hybrid approaches

With the increase in the number of design objectives, constraints or design variables, other techniques have been proposed in order to improve the computational performance of the previously mentioned algorithms. Among them, cooperative co-evolutionary algorithms (CCEAs) first introduced by Potter and De Jong [Pot+94] and based on the divide-and-conquer method along with the biological model of co-evolution of cooperating species have been developed. CCEAs allow to tackle complex optimization problems by decomposing it into multiple subproblems, each solved with evolutionary algorithms. A complete problem solution is constructed by merging the representative members of each subpopulation (associated to one subproblem) through cooperative co-evolution mechanisms. Ma *et al.* recently proposed a survey on CCEAs [MLa18]. Those algorithms were successfully applied to optimal layout optimization [CZa19; DRW03; WTS09a]. Cooperative techniques have also been introduced within the frameworks of other algorithms. For instance, Mansour *et al.* introduced a cooperative version of a local search algorithm based on quality indicator for multi-objective knapsack problems [MBS18].

Metaheuristics have also often been hybridized between themselves. The goal is to take advantage from the strengths of several metaheuristic techniques to find improved global solutions with lower computational resources. In [LL02], the authors combined the ability of TS and SA algorithms to find local solutions rapidly to enhance GA's capabilities to find a global solution with lower computational efforts. The same idea was employed in [LPK17], where a GA has been hybridized with differential evolution, artificial bee colony (ABC) and PSO in order to optimize layouts for multi-robot cellular manufacturing systems and accelerate the convergence toward the global optimum. In [LC12b], the authors proposed a hybrid swarm intelligence based particle-bee algorithm (PBA) for construction site layout optimization. PBA integrates the ABC global search ability with the local search advantages of PSO and proved to perform better than non-hybridized ABC and PSO.

Metaheuristics have also been hybridized with exact methods and heuristic techniques. In [Che+18], the authors improved the convergence speed and robustness of DE by integrating a sequential quadratic programming technique to accelerate the local search during the DE process and solve satellite module layout problems. The OLPs were sometimes reformulated as two-stage problems, each stage addressed using both EAs and heuristic techniques. For instance, in [XXA07c], a GA optimizes the order of the items to place within a container while a greedy algorithm based on the order-based positioning technique and heuristic rules constructs the layout component by component thereafter. Then, the same authors then extended the placement heuristic rules to parallelepipedic items in [Xu+10].

2.3.5 Quasi-physical methods

Quasi-physical (QP) methods are techniques having both physical and mathematical roots and which aim is to mimic physical laws. Most of the time, they correspond to energy or force-based algorithms. In [HY11a], a quasi-physical global optimization method is described. A local search strategy based on an energetic and physical modeling of the circles has been coupled with a basin-hopping procedure to escape from local optimum traps. This quasi-physical strategy has been extended in [He+18] using a modified Broyden–Fletcher–Goldfarb–Shanno algorithm and a new basin-hopping strategy to solve unequal circular bin packing problems. Guo *et al.* developed an evolutionary approach for spatial architecture layout including a multi-agent finding system based on physical interactions between agents [GL17].

Another algorithm inspired from physical laws is the Gravitational Search algorithm (GSA). GSAs are population-based algorithms based on the law of gravity and mass interactions introduced by Rashedi *et al.* in [RNS09]. In GSAs, individuals of the population are modeled as virtual objects with a virtual mass. Thus, the heavier the individuals, the better the solution. According to the gravitational law of physics, the individuals attract each other by the gravity force which leads to a global movement of all objects towards the objects with heavier masses. To ensure the exploitation ability of the algorithm, heavy masses move more slowly than lighter ones. GSAs also belong to the population-based metaheuristic category of algorithms. In [SR17], a discrete GSA is developed for solving the knapsack problem. For this purpose, a new approach was introduced for discretely updating individuals' positions in the GSA. In addition, a new proper fitness function for 0–1 knapsack was introduced to improve the quality of the optimum. Abdessamia *et al.* proposed a strategy for virtual machine placement based on GSA to find the optimal energy consumption [AZT20].

Finally, one other quasi-physical approach is referred to as virtual-force method and consists in applying forces to the items in order to make them virtually move thanks to dynamic physical laws. Thus, the forces aim either at minimizing the objective functions or satisfying the constraint functions which could lead to an improved convergence speed. This approach is used in the field of robotics in order, for instance, to control a swarm of robots moving in a given area with one or several objectives (*e.g*, reaching a target destination), as well as hard constraints (*e.g.*, avoid collisions and forbidden zones) [Bra+13; KB86]. This approach is often referred to as Artificial Potential Field (APF) method for robot path planning [OMS19; Pat+19; VTM00]. Virtual-force systems-based methods have also been applied to coverage and packing problems, where elastic models resulting in virtual forces are applied to the items to position in order to satisfy the overlapping or balancing constraints [He+13; HY11b; Wan+02].

2.3.6 Machine Learning techniques

Recently, Machine Learning (ML) methods have been applied to layout optimization problems even if they are less investigated. Burggräf *et al.* conducted a study on the use of machine learning as resolution techniques for facility layout problems in [BWH21]. Unsupervised learning, supervised learning and reinforcement learning (RL) have been explored to solve these problems. More specifically, Tsuchiya *et al.* [TBT96] focused on a neural network (NN) approach to optimize the layout of N components on N^2 locations. Despite the small amount of articles using RL to tackle aerospace vehicles OLPs, some recent ones use this method to address chip floorplanning which is similar to a layout problem. In [Vas+20], a cyclic combination of RL and SA is proposed to solve integrated circuit placement. This cyclic application of RL and SA is used to provide a better initialization for SA. In [Mir+21], the authors developed a deep RL approach capable to learn from its experience of achieving the layout of the chip in order to become better and faster. A trained agent places the macros one by one on the chip and by the end obtains a reward which is function of the wirelength, the congestion and the density of the chip. Moreover, the authors created a neural network architecture in order to predict reward on new lists of components and use it as the encoder layer of their policy. In [CY21], a joint learning method for solving both placement and routing problems is detailed. It integrates reinforcement learning with a gradient-based optimization scheme. As in [Mir+21], a policy network learns how to maximize the rewards.

Bayesian Optimization has been also applied to OLPs [Des+22; ONY22; SL22]. This technique has the advantage of requiring few iterations in order to converge due to the fact that each evaluated point is carefully chosen thanks to an inherent optimization process [Sha+15]. Indeed, this method is particularly suitable in case of blackbox expensive-to-evaluate functions. As OLPs are often characterized by this type of functions, BO algorithms have been applied to this class of problems. In [ONY22], BO has been used for speeding up the thermal design optimization of an electronic circuit board layout with transient heating chips. The authors demonstrated that using BO instead of metaheuristic methods such as GA or PSO helped to divide the CPU time required to reach the optimum by 4 to 5. In [Des+22], the authors applied two BO algorithms dedicated to permutation spaces for optimal layout benchmarks (QAP and circuit integrated layout). They used kernels dedicated to the permutation spaces referred to as the Kendall and Mallows kernels [JV15] and adapted the acquisition function and their optimization thanks to heuristic searches. Sajedi *et al.* proposed a Deep generative BO for sensor placement in structural health monitoring [SL22]. They combined the BO framework with adaptive basis regression with deep neural network instead of classical Gaussian Process in order to enhance the convergence speed of the algorithm. Additionally, they developed an explorative local search algorithm in order to optimize the Expected Improvement acquisition function. They showed that their algorithm outperformed GA with the same number of function evaluations.

2.3.7 Literature summary

Table 2.3 sums up all the methods which have been surveyed for solving fixed search space OLPs. When possible, for each approach, the very first reference to this method or at least a representative reference is given as well as the main references dealing with OLPs from the present survey. First of all, exact and heuristic methods often require too much computational efforts to solve complex design space OLPs. To overcome this, population-based metaheuristic algorithms are the most employed techniques during the past 20 years, as they generally provide good global search capabilities as well as exploitation toward the most promising areas. Thus, a further description and analysis of these methods are provided in the next section.

Moreover, when it comes to highly dimensional and constrained OLPs, one key aspect of every algorithm is the choice of a constraint-handling technique. Consequently, quasi-physical techniques are also promising techniques as they provide an inherent way to handle the constraints thanks to dedicated forces or operators. Thus, these approaches have good convergence speed in comparison to most metaheuristic techniques and an extensive description of quasi-physical techniques is given in the following section.

Table 2.3: Summary of the literature reviews.

Approaches	**Algorithms**	**Definition**	**Main ref.**
Exact	GAMS/CPLEX	[BM04; BBL14]	[BMN02; Geo+99; PP02]
	B&B	[ER66]	[ChL13; KYS08; XS08]
	DP	[Edd04]	[Cin+08; CH06; FLS17]
Heuristic	Construction	[Lee67; SE67]	[BHS20; HI15; Des+16]
	Improvement	[AB63; BME94]	[Joh82; LKM17; PRR14]
Metaheuristic	SA	[KJV83]	[BG01; BR84; Tam92a]
	GA	[Joh82]	[Jac+09; Tam92b; TS95]
	TS	[Glo86]	[CCC14; LC08]
	SS	[LM03]	[BT97; KG14; WST09]
	PSO	[KE95]	[Liu+18; XXA07b]
	DE	[SP97]	[Wan+17; Zha+17]
	CMA-ES	[HO01]	[AWT16; Wag+11; Wan+17]
	ACO	[DBS06]	[Has+17; LD04]
	VNS	[MH97]	[CP13; HCD21; Rip+13]
	QD	[LS11; MC15; PSS16]	[Gal+21; NDN22; Zha+23]
Hybrid	Coevolution	[Pot+94]	[CZa19; DRW03; MBS18]
	M+M*		[LL02; LC12b; LPK17]
	M+E*/M+H*		[Che+18; XXA07c; Xu+10]
QP	GSA	[RNS09]	[AZT20; SR17]
	Energy-based		[He+18; HY11a]
	Virtual-force		[GC02; Ji+22; ZC03]
ML	NN	[Law93]	[TBT96]
	RL	[KLM96]	[CY21; Mir+21]
	BO	[Sha+15]	[Des+22; ONY22; SL22]

2.4 Focus on promising methods for solving fixed search space optimal layout problems

2.4.1 Population-based Algorithms

In this section, a brief description is given for the main population-based algorithms. The generic flowchart of population-based algorithms is as follows:

1. Initialization of the population: A population of individuals is randomly generated as an initial solution to the problem at hand. Each individual represents a potential solution.

2. Evaluation of the population: Each individual in the population is evaluated thanks to the objective function and is assigned a fitness score (most of the time corresponding to the value of the evaluation of the objective function).

3. Generation of the new population: In this step, new individuals (offspring) are created by combining genetic information from the current individuals (parents). The offsprings generated in the previous step replace some individuals in the current population. The replacement strategy can vary, but a common approach is to replace individuals with poor fitness scores, thus allowing the population to evolve towards better solutions over time.

4. Termination: The algorithm continues iterating through the previous steps of evaluating the current population and generating a new population until a termination condition is met. This condition can be a maximum number of generations, a stagnation of the convergence, or a predefined time limit.

By repeatedly applying the aforementioned steps, population-based algorithms explore and refine the population over generations, gradually converging towards better solutions to the problem at hand. Figure 2.1 shows a classification of population-based metaheuristic algorithms.

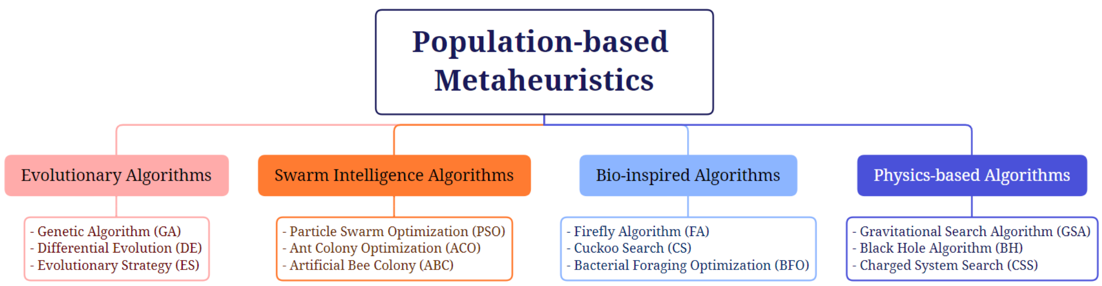


Figure 2.1: Classification of population-based metaheuristics.

In the following sections, three representative population-based metaheuristics are detailed: GA, PSO and CMA-ES.

2.4.1.1 Genetic Algorithms

GAs are the most used Evolutionary Algorithms for solving OLPs. They are based on different methods to encode the solution providing an easy way to perform reproduction operations through crossover and mutation. A solution is encoded as a chromosome were each gene corresponds to a design variable. The bit string representation or the real-value representation are the two main ways to encode a solution. Figure 2.2 illustrates the GA encoding of a population of solutions with a bit string representation. The encoding of solutions allows to perform evolutionary operators

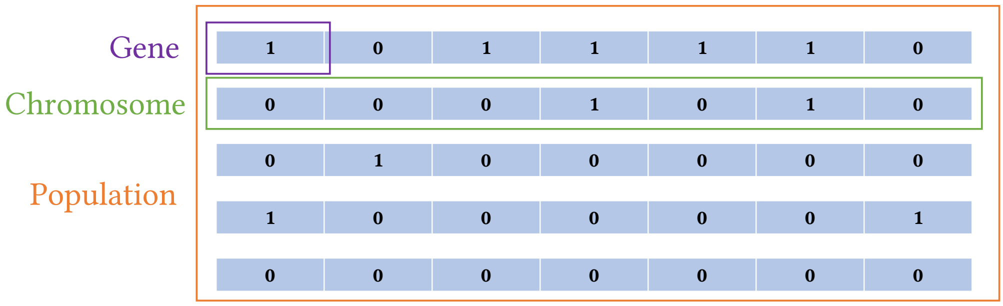


Figure 2.2: Encoding of a population of 5 individuals with 7 genes in GAs, with bit string representation.

in an easy way. In [Tal09], classical initialization, selection, crossover, mutation and replacement operators are detailed. They are illustrated in the case of GAs solution encoding in the following paragraphs.

Initialization First of all, different techniques exist in order to initialize the population. Usually, the population is initialized randomly [Tal09], but other techniques can be used such as sequential diversification [Dig83], parallel diversification [Tal09] or any heuristic rules. The initialized individuals are then evaluated in terms of objective function and constraints.

Selection Then, the selection mechanism aims at choosing the individuals that will be part of the reproduction process. Usually, the better is an individual (in terms of fitness value), the higher its chance to become a parent in order to lead the population toward better individuals. The most common strategy is the roulette wheel selection [LL12]. Other selection mechanisms can be employed such as tournament selection [Bli00] in which k individuals are selected randomly and a tournament is applied to the k members of the tournament group to select the best one, or even rank-based selection [Gre00].

In GAs, the reproduction step is usually performed using a crossover operator and a mutation operator which are detailed in the following paragraphs.

Crossover. The role of crossover is to inherit some characteristics of the two selected parents in order to generate offsprings. The design of a crossover operator depends on the encoding of the solutions and must answer two points:

- Heritability: the offsprings must inherit of the genetic material of both parents;
- Validity: the crossover operator should produce new valid solutions (which is not always possible in case of constrained problems).

The crossover operators rely on a crossover probability P_c, $P_c \in [0, 1]$ which is a hyperparameter of the GA and characterizes the proportion of parents on which a crossover operator will act.

The most simple crossover operator is the 1-point crossover and its generalized form is the n-point crossover [SJ91] which are originally developed for the binary representation. In the 1-point crossover, a uniform random distribution is used to select a crossover point within the parents chromosomes. The offsprings are obtained by recombination of the segments before and after the chosen point. Figure 2.3 illustrates a 1-point and a 2-point crossover operations. Other mechanisms for binary representation exist such as the uniform crossover.

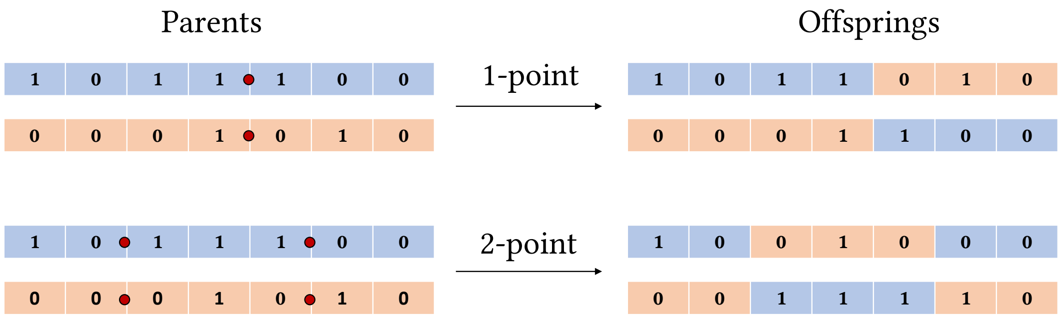


Figure 2.3: 1-point and 2-point crossover operator.

For the real-number representation some crossover operators have been developed. The most employed one is the Simulated Binary Crossover (SBX) [DA95]. Other crossover operators can be used with real-number representation like the unimodal normal distribution crossover [OK97], the half uniform crossover [Bue13] or the parent-centrix crossover [DJ02].

Mutation. The mutation operator corresponds to a small change in some selected individuals of the population. As for the crossover operator, the mutation probability P_m, $P_m \in [0, 1]$ is a hyperparameter and defines the probability to mutate each gene of the chromosome. The design of a mutation operator depends on the chosen representation and must rely on three points:

- Ergodicity: All individuals of the population might be reached;
- Validity: The mutation operator must produce valid individuals (it may not be possible when it comes to constrained problems);

- Locality: The mutation should produce a minimal change.

For binary representation, the most employed mutation operator is the flip operator: A randomly selected gene of a chromosome is flipped from 0 to 1 or from 1 to 0 [Tal09]. For real-value representation, one of the most employed operator is the Polynomial Mutation (PM) [Deb01]. Other mutation operators exist such as uniform random mutation or normally distributed mutation [Tal09].

Replacement. Finally, a strategy must be chosen in order to withdraw half of the parents and offsprings as the size of the population must remain constant. The most employed technique is the Non-Dominated Truncating [Deb+02] which consists in keeping the best individuals among the parents and offsprings populations. Other strategies exist such as the generational replacement [Hol92] where the offsprings systematically replace the parents, or the steady-state replacement [LHC09] in which the best offspring replaces the worst parent.

GAs are often effective in exploring the entire search space even in case of complex design search space. Moreover, they can be used to solve a wide variety of optimization problems, whether they are continuous, discrete, variable-mixed, unconstrained/constrained, single or multi-objective. They have thus been widely used for OLPs as reviewed in Section 2.3.

2.4.1.2 Particle Swarm Optimization

In Particle Swarm Optimization (PSO), a solution i is encoded as a particle defined by its position p_i (vector composed of the values of the design variables) and its speed v_i. The goal is thus to evolve the population as a coordinated behavior using local movements without any central control. At each iteration, each particle moves from one position to another in the search space with its associated velocity using no gradient information. The updated positions are influenced by their own success and the success of the other particles. Indeed, two factors are taken into account:

- The best position reached by the particle itself $\mathbf{p}_i^{best}$;
- The best position reached by the whole swarm $\mathbf{g}^{best}$.

A each iteration t, the velocity of each particle i is updated as follows:

$$\mathbf{v}_i^t = \omega \mathbf{v}_i^{t-1} + \rho_1 C_1(\mathbf{p}_i^{best} - \mathbf{p}_i^{t-1}) + \rho_2 C_2(\mathbf{g}^{best} - \mathbf{p}_i^{t-1}) \tag{2.2}$$

where ρ_1 and ρ_2 are random numbers between 0 and 1 and ω, C_1 and C_2 are hyperparameters of the algorithm.

Then, the position of each particle is updated as:

$$\mathbf{p}_i^t = \mathbf{p}_i^{t-1} + \mathbf{v}_i^t \tag{2.3}$$

Finally, the $\mathbf{p}_i^{best}$ vectors as well as the $\mathbf{g}^{best}$ vector are updated. Figure 2.4 illustrates the update of the position of a particle from step t-1 to step t.

The PSO algorithm has the advantage of being easily implementable and provides good global search exploration abilities [EHG05; Esp+11; Has+05]. Consequently, PSO have been widely used for solving OLPs as described in Section 2.3.3.1.

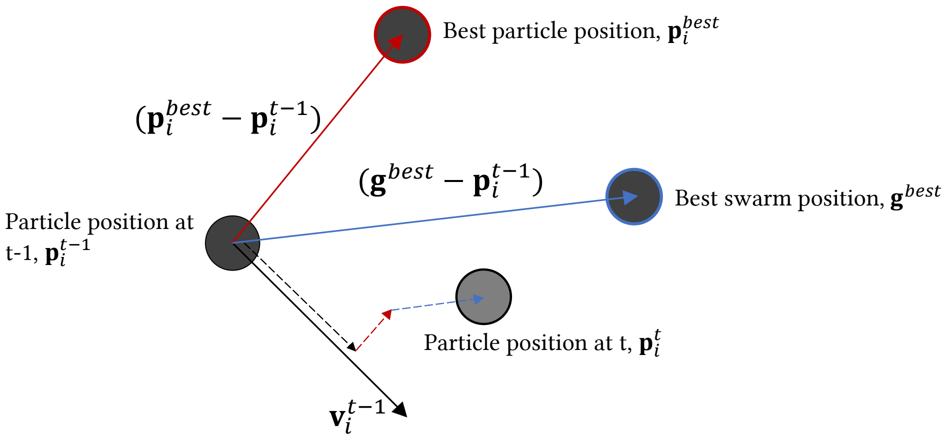


Figure 2.4: Illustration of a step of PSO algorithm [Tal09].

2.4.1.3 Covariance Matrix Adaptation-Evolution Strategy

Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) optimizes the objective function through generation of a population using a multi-variate Normal distribution. This distribution is parameterized by its mean and covariance matrix. CMA-ES is a second order approach estimating a positive definite matrix within an iterative procedure. The key idea of this algorithm is to dynamically adapt the covariance of the candidate distribution to better explore the search space and converge faster to optimal solutions [HO01]. The covariance matrix defines the pairwise dependencies between the variables in the distribution. Adaptation of the covariance matrix is based on learning a second-order model of the target objective function, which is reduced to the approximation of the inverse Hessian matrix in the quasi-Newton method, a traditional method in continuous optimization.

At each generation of CMA-ES, λ offspring solutions are generated from μ parents. In order to select new μ parents at the next generation, a (λ, μ)-selection is used in which the μ best offspring candidates are chosen based on their ranking according to their fitness value. The multivariate normal distribution is characterized by an ellipsoid delimiting a probable search hypervolume. Throughout the iterations, the search hypervolume is updated in order to converge and to shrink around the global optimum. CMA-ES generates the population by sampling a multivariate normal distribution, a iteration t:

$$\mathbf{x}_i^t \sim \mathbf{m}^{t-1} + \sigma^{t-1}\mathcal{N}(0, \mathbf{C}^{t-1}), \quad \text{for } i \in \{1, ..., \lambda\} \tag{2.4}$$

where x_i^t is an offspring candidate generated from a mean vector $\mathbf{m}^{t-1}$, a step size σ^{t-1} and a multivariate normal distribution $\mathcal{N}(0, \mathbf{C}^{t-1})$ with 0 mean and a covariance matrix $\mathbf{C}^{t-1}$. The update of the covariance matrix incorporates dependence between the past generations and between the μ best solutions from the previous generation. The mean vector characterizes the center of the next population and is determined by a combination process through the weighting of the μ best solutions. A more detailed description of the selection and update mechanisms can be found in [HO01]. Figure 2.5 gives the steps of CMA-ES.

CMA-ES is particularly effective in handling problems with complex, non-linear landscapes and is very competitive for continuous optimization problems and does

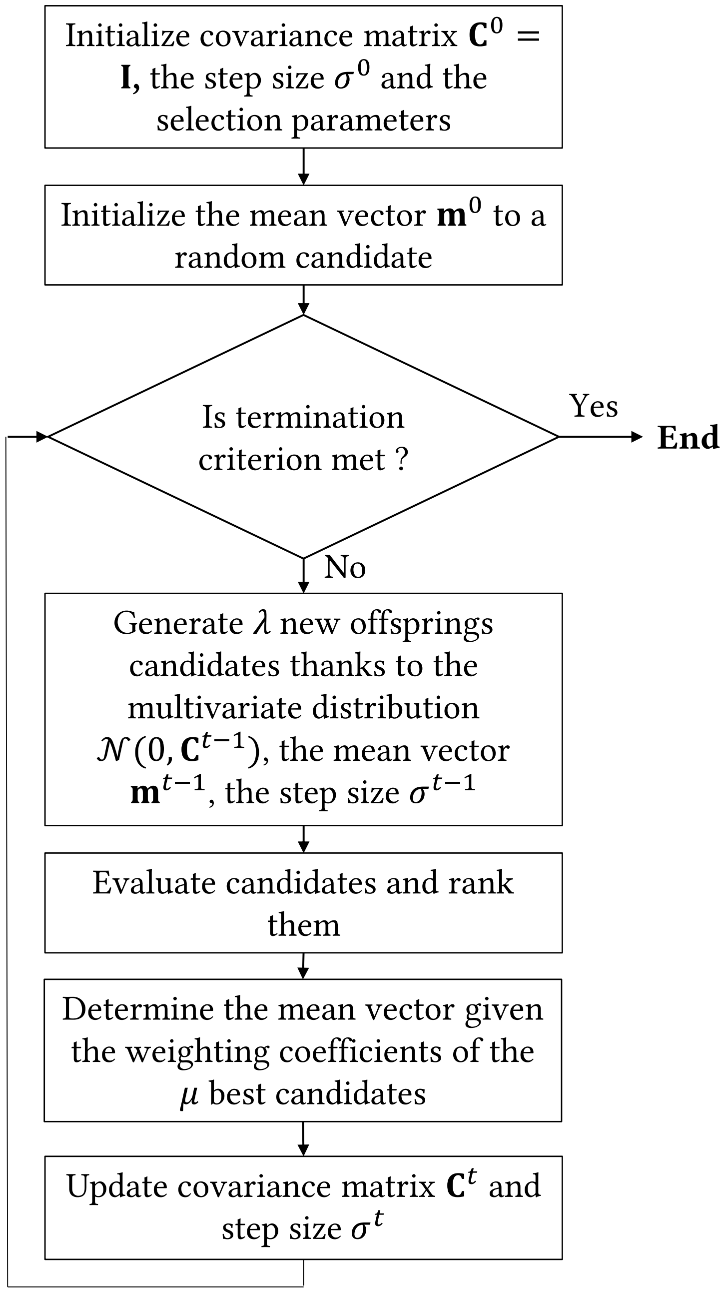


Figure 2.5: CMA-ES flowchart.

not require any parameters tuning. Then, it has been used for solving OLPs as described in Section 2.3.3.1.

2.4.1.4 Analysis of constraint-handling in population-based metaheuristic algorithms

For population-based metaheuristics, one key aspect is the choice of constraint handling techniques especially in case of highly constrained OLPs that may involve hundreds of non linear constraints.

Penalty-based approaches. The most employed approaches are the penalty-based approaches which consist in modifying the objective function by adding a penalty term which depends on the weighted constraint functions [Lag+23]. Many penalization techniques have been developed like death penalty [CC99], static penalty [HQL94], dynamic penalty [JH94] or adaptive penalty [HB97]. Different studies aim at comparing them on different benchmarks [HGR21; MMT03; Rah+23]. Globally, penalty-based approaches are easy to implement but require a precise weight tuning, even if some penalty techniques are parameters-free [MVV13; SBL11]. They often lack of precision toward the global optimum and lead to premature convergence. They can also be over time consuming [Rah+23].

Separation between objective and constraints. Alternative techniques consist in separating the objective and the constraints. Among them, the dominance-based constraint handling which is inspired from multi-objective evolutionary algorithms [Deb00]. The idea is thus to transform a constrained single-objective optimization problem into an unconstrained multi-objective problem, where each constraint represents a new criterion to be minimized. The following principles are used in a tournament [CM02]:

- An infeasible solution is dominated by a feasible one;
- If two individuals are feasible, the one with the worst objective function is dominated;
- If two individuals are infeasible, the one with greatest constraint violation is dominated.

Constraint-dominance techniques can be easy to implement [Agu+04; SR97; ZLK03]. However, they can restrict the diversity among the population as they promote convergence towards feasible solutions from the beginning of the search, which are not necessarily the most promising individuals. In order to overcome this trend, Stochastic Ranking was developed to add stochasticity in the constraint-dominance process and sometimes favors individuals with worse constraint violation but better objective function [RY00].

Retaining the infeasible individuals. In methods based on retaining the infeasible individuals in the population, the constraints are usually treated as an additional objective. Therefore, many infeasible solutions are saved during the evolution process [PLG17; Ray+09]. Ning *et al.* [Nin+17], proposed a mechanism in which the individuals with low Pareto rank and low rank of the constraint violation are selected. These methods make use of infeasible solutions which can increase the diversity of the population. However, a huge number of infeasible individuals in the population may cause low convergence speed of feasible solutions.

Repairing approach. Another constraint-handling technique consists in repairing infeasible solutions to obtain feasible ones [Arn11; Koc+15; Sal09]. Usually, the repairing strategy corresponds to repairing heuristics which are specific to the optimization problem at hand. Most of them are greedy heuristics. Then, the success of this strategy will depend on the availability of such efficient heuristics and might be very time consuming.

Hybrid approaches. Finally, hybrid methods can combine several constraint-handling techniques such as in [LDD18; Fan+19; QS11]. Hybrid methods may benefit from the strengths of each chosen technique. However, implementation challenges may arise and improper integration may cause low convergence speed.

It must be noted that in the case of the CMA-ES algorithm, a (1+1)CMA-ES algorithm which corresponds to a simplified version of the (μ, λ)CMA-ES algorithm has been proposed by Arnold and Hansen in [AH12] in order to deal with constrained optimization problems. However, this algorithm is inefficient for high dimensional optimization problems [AH12].

In general, handling constraints is often complicated for metaheuristics and remains a challenge for large constrained optimization problems [CM02]. As the constraint-handling technique is a key aspect in the resolution of highly-constrained OLPs, the following section focuses on quasi-physical methods and more precisely virtual-force-based methods which provide an inherent way to handle the constraints.

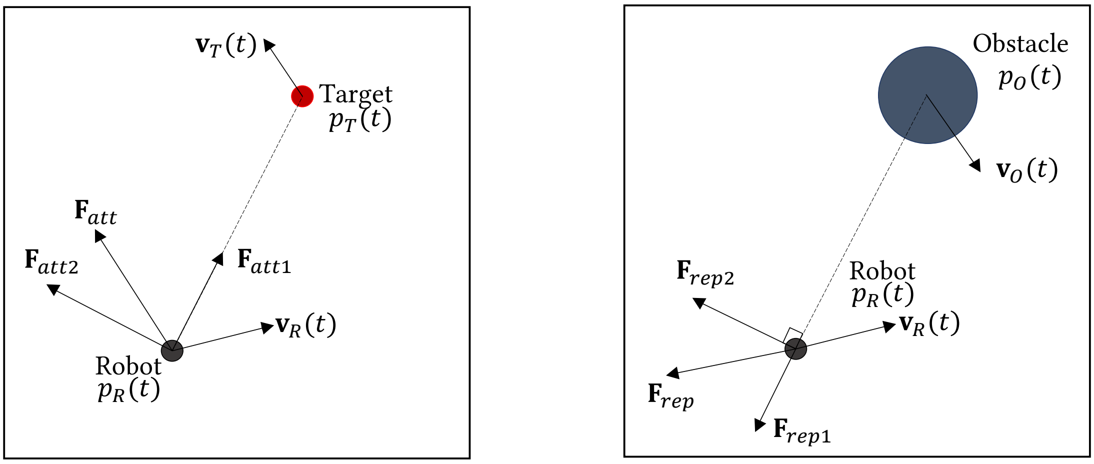
2.4.2 Virtual-force systems-based methods

Virtual-force systems-based methods refer to quasi-physical methods in which forces are applied in order to virtually move items and solve the optimization problems at hand [He+13]. More specifically, the forces aim at solving the constraints and converge toward the objective even with a large number of constraints. Algorithms based on a virtual-force system are used for robots path planning in order to move a swarm of robots toward a target and satisfying hard constraints as for instance avoiding obstacles [Bra+13; KB86]. This strategy has also been adapted in the literature for OLPs. More particularly, this technique has mostly been used for coverage problems as well as packing problems [He+13; ZC03; ZC04]. Thus, in this section, three virtual-force systems are detailed for robots path planning, coverage problems and packing problems.

2.4.2.1 Virtual-force systems for robots path planning

In [GC02], Ge *et al.* proposed an Artificial Potential Field (APF) method for mobile robot motion planning in a dynamic environment where both the target and obstacles are moving. The potential functions take into account the relative positions of the robot with respect to the target as well as the relative velocities of the robot with respect to the target and obstacles. Accordingly, the virtual force is defined as the negative gradient of the potential with respect to both position and velocity. The motion of the mobile robot is then determined by the total virtual force through the Newton's Law of Motion. Figure 2.6 shows the forces defined in [GC02].

Figure 2.6a defines the attractive force $\mathbf{F}_{att}$ dedicated to following the moving target which consists of two components: while the first component, $\mathbf{F}_{att1}(p_R(t), p_T(t))$



(a) Attractive forces

(b) Repulsive forces

Figure 2.6: Illustration of the virtual-force system employed in [GC02] for mobile robot path planning.

which depends on both positions of the robot and the target $p_R(t)$ and $p_T(t)$, pulls the robot to the target and shortens the distance between them, the second component, $\mathbf{F}_{att2}(v_R(t), v_T(t))$ which depends on both velocities of the robot and the target $v_R(t)$ and $v_T(t)$, tries to drive the robot to move at the same velocity of the target. Figure 2.6b defines the repulsive force $\mathbf{F}_{rep}$ dedicated to avoid moving obstacles and depends on the positions of both robot and obstacles, $p_R(t)$ and $p_O(t)$, as well as their velocities $v_R(t)$ and $v_O(t)$. The repulsive force component $\mathbf{F}_{rep1}$ will keep the robot away from the obstacle while the repulsive force component $\mathbf{F}_{rep2}$ will act as a steering force for detouring. The magnitude of each component of the forces is defined based on attractive and repulsive potential functions defined as functions of the relative position and velocity of the target with respect to the robot.

2.4.2.2 Virtual-force systems for coverage problems

In [ZC03; ZC04], the authors proposed a virtual-force system in order to optimize the layout of cluster-based sensor network architectures. Attractive and repulsive forces are applied to the sensors in order to optimize the coverage of the sensor layout. In [Ji+22], Ji *et al.* improved previous virtual-force systems by adding a vortex force in order to enhance the exploration capabilities of robots maximizing the coverage within unknown environments. They compared their algorithm with several previous virtual-force algorithms [RR16; Xie+19; ZC03]. Figure 2.7 illustrates the forces defined for the coverage problems in [Ji+22].

The repulsive forces are of three types: a repulsive force between sensors $\mathbf{F}^{i,j}_{RA}$ in case of overlap of the coverage zones of sensors i and j, a repulsive force from obstacles $\mathbf{F}^{i}_{RO}$ in case of overlap between the coverage zone of sensor i and an obstacle and a repulsive force from the border $\mathbf{F}^{i}_{RB}$ in case of overlap between the coverage zone of sensor i and the border. The vortex forces (in dotted lines on Figure 2.7) are of two types: a vortex force from obstacles $\mathbf{F}^{i}_{VO}$ which can be oriented toward the two directions along the obstacle in case of overlap between the coverage zone

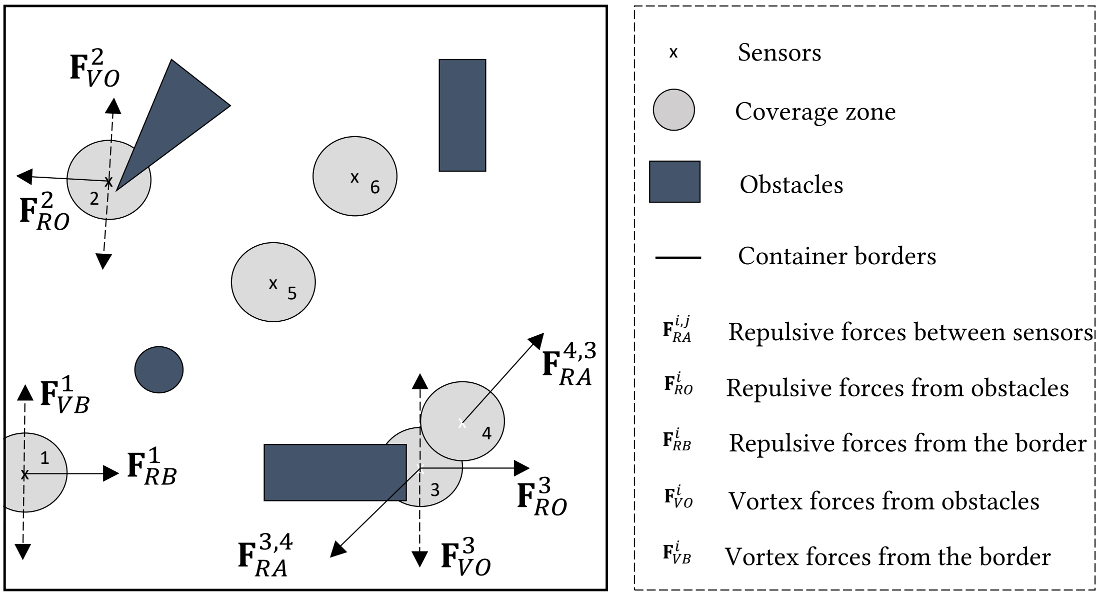


Figure 2.7: Illustration of the virtual-force system employed in [Ji+22] for sensors coverage problems.

of sensor i and an obstacle, and a vortex force from the border $\mathbf{F}^i_{VB}$ which can be oriented toward the two directions along the border in case of overlap between the coverage zone of sensor i and the border. The magnitudes of the repulsive forces are proportional to the relevant overlapping areas and the magnitudes of the vortex forces are proportional to the magnitude of the corresponding repulsive forces.

2.4.2.3 Virtual-force systems for packing problems

Mostly circular bin packing problems have been addressed using virtual-force system [HY11b; Wan+02]. In [He+13], the authors addressed the circular bin packing problems with equilibrium constraints using a coarse-to-fine virtual-force-based method. The objective of the packing problem is to minimize the container radius. The considered constraints are: no overlapping between the circles is allowed, no overlapping between the circles and the container is allowed and the centroid of the weighted circles must be in a tolerance zone centered at the geometrical center of the container. The optimal radius is searched by dichotomy and for each tested radius, two quasi-physical models are proposed, respectively dedicated to the overlapping constraints and to the balancing constraint. The algorithm consists in the application of the two quasi-physical models to guide the system to local minima combined with a basin hopping with tabu method to find new promising areas.

Radius Search. The optimal radius search is performed using a dichotomy process:

1. Set lower and upper bounds for the container radius such that the circles have to overlap with the lower radius R_{low} and can be put in the container without

overlapping with the upper radius R_{up}.

2. Set $R_{mid} = (R_{low} + R_{up})/2$ and try to solve the bin packing problem with a container of radius R_{mid}.

3. If a feasible solution have been found, set $R_{up} = R_{mid}$. If no feasible solution has been found then set $R_{low} = R_{mid}$.

4. Repeat the two previous steps until R_{low} and R_{up} are close enough. The solution of the circular bin packing problem is R_{up}.

Figure 2.8 illustrates the radius search process.

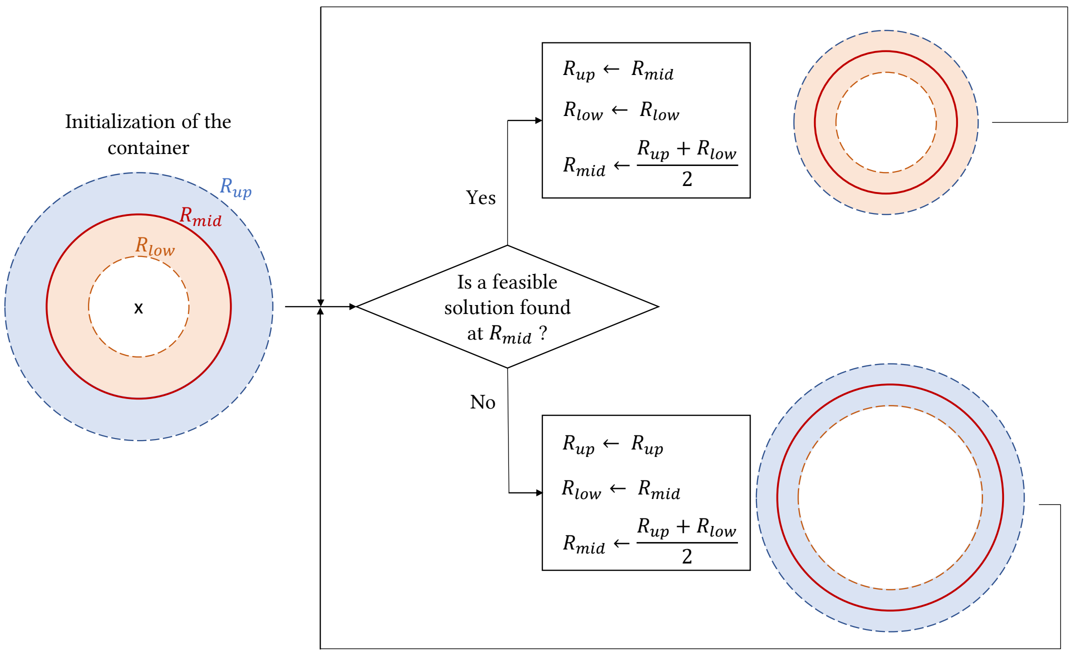


Figure 2.8: Illustration of the radius search process.

Quasi-physical model for overlapping constraints. This model considers the circles as elastic. A deformation degree d_{ij} is defined as the maximal distance of intersection between the overlapping disks i and j (or between a circle and the container). Then an elastic force $\mathbf{F}_{ij}^{el}$ is defined as follows:

$$\mathbf{F}_{ij}^{el} = k_e d_{ij} \frac{\mathbf{r}_{ij}}{||\mathbf{r}_{ij}||} \tag{2.5}$$

where k_e is a spring constant of the circles and $\mathbf{r}_{ij}$ denotes a vector from the center of j to that of i.

The total elastic force applied on one circle is defined as the sum of each force acting on this circle.

Quasi-physical model for the balancing constraint. The model to address the balancing constraint is a pull model based on the assumption that there exists an elastic rope connecting the centroid of the weighted circles (x_c, y_c) and the center of the container $(x_e, y_e) = (0, 0)$. If the centroid of the weighted circles does not coincide with the center of the container a pull force $\mathbf{F}^{pull}$ is applied to the centroid and thus to all the circles:

$$\mathbf{F}^{pull} = k_p \frac{(-x_e, -y_e)}{||(x_e, y_e)||} \tag{2.6}$$

where k_p is a spring constant of the imagined elastic rope. It must be noted that the balancing constraint is sometimes relaxed by considering the centroid of the circles as the geometrical center of the container [Wan+19b].

Algorithm description. The main steps of the algorithm are summarized as follows:

1. The circles are randomly put within the container.
2. The elastic and pull models are applied to move the circles as:

   $$\mathbf{r}_i^t = \mathbf{r}_i^{t-1} + h_e \mathbf{F}_i^{el,t-1} + h_p \mathbf{F}^{p,t-1} \tag{2.7}$$

   where h_e is a parameter for the step length under the elastic force, h_p is a parameter for the step length under the pull force, and $\mathbf{r}_i^t$ is the vector of the positions of circles i.
3. When the circles do not move any more, if the final layout is feasible then it is considered as an acceptable solution. Otherwise, a new configuration is generated by using a basin hopping strategy [He+13].

To sum up, virtual-force systems often rely on similar forces: attractive, repulsive or specific forces (*e.g.*, gradient-based or vortex forces). Those forces are related to a function of the problem and a dynamical law allows the system to explore the search space and to converge [GC02; He+13; ZC03; ZC04]. As illustrated by the three detailed examples, the virtual-force system depends on the characteristics of the problem at hand but rely on a common framework. Moreover, the forces rely on simple operations and are easily implementable as well as often compatible with GPU computation which promotes reasonable resolution times even for high numbers of items to lay out, constraints and objective functions. Therefore, virtual-force system-based methods are versatile enough with strong convergence capabilities and are promising when it comes to deal with highly constrained OLPs.

2.5 Conclusion

In this chapter, the mathematical formulation of fixed search space design space OLPs is presented as well as the existing methods allowing to handle them. The survey of the methods allowed to identify two promising categories of methods in order to solve highly dimensional and constrained OLPs: evolutionary algorithms and virtual-force systems-based methods. Evolutionary algorithms provide various frameworks that can be easily adapted to the problems at hand and benefit from a large number of references in the literature. Virtual-force systems-based methods are versatile methods which provide an inherent way to handle the constraints and are thus promising candidate techniques to solve highly constrained problems.

In this chapter, the formulation of OLPs remains very generic. Thus, in the following chapter, the OLPs addressed in this thesis are specified in terms of geometry, design variables, objective function and constraint functions. In particular, single and multi-container configurations are considered. Moreover, the limits of existing virtual-force systems for solving the formulated OLPs are outlined and a Component Swarm Optimization algorithm based on a Virtual-Force system is developed for solving the single-container OLPs and overcome the outlined limits. This algorithm is then hybridized with a genetic algorithm in order to address multi-container configurations.

Chapter 3

Quasi-Physical Approach for Single- and Multi-container Optimal Layout Problems

Contents

Chapter contributions

- Formulation of the generic fixed search space (FSS) single and multi-container optimal layout problems.
- Development of the Component Swarm Optimization algorithm based on Virtual-Force system (CSO-VF) devised to solve for FSS single-container optimal layout problems.
- Development of the two-stage approach based on Genetic Algorithm and the CSO-VF algorithm devised to solve for FSS multi-container optimal layout problems.

3.1 Introduction

Most often, optimal layout problems (OLPs) can be characterized by the presence of single or multiple containers [HH09; TTi+16]. First of all, single-container OLPs involve optimizing the layout of a given set of components in one container while satisfying geometrical and functional constraints. As mentioned in Chapter 2, quasi-physical techniques have been developed for packing problems which consist in positioning non-overlapping circles in the smallest circular container with or without balancing constraints [He+13; HY11b].

In this chapter, those aforementioned packing problems are extended in order to take into account some specificities related to more realistic OLPs. Indeed, polygonal components and containers are considered, exclusion zones are added to the container, functional constraints are defined and the objective function is generalized to any possible function. Limitations of previous virtual-force systems used in classical packing problems with respect to the single-container OLPs are highlighted and a quasi-physical approach which consists in a Component Swarm Optimization algorithm based on a Virtual-force System (CSO-VF) is developed.

When the components must be laid out within several containers, the OLP is referred to as a multi-container problem. In this case, an assignment task is added to the layout task in order to assign each component to one of the containers. The assignment scheme of the components to the appropriate containers has a critical impact on the performance of the system. Indeed, a poor-quality assignment scheme might lead to non feasible or suboptimal layouts. Multi-container OLPs have several industrial applications. For instance, the multi-container loading problems consist in loading several containers like trucks with pallets or cartons of different sizes.

The numbers and types of loaded cartons as well as their layouts within each assigned truck both influence its dynamical features *i.e.*, mass, stability, intertia, *etc.* [Alo+19; CLS95; TTi+16]. Multi-floor layout problems which require to optimally place departments, storage and elevators within a high-rise building, as means to reduce costs, is yet another frequent application [HLG21; IES14; MIY14]. Finally, multi-module satellite OLPs consist in positioning components within different compartments or surfaces of a satellite module in order to optimize dynamic or space requirements [CSY02; CXW18; Ten+09]. All these test cases involve similar features that are an assignment task in order to assign items to a container and an underlying layout task. In this chapter, an algorithm framework based on the aforementioned quasi-physical algorithm and enhanced by a genetic algorithm is proposed to handle this category of problems.

Then, the rest of the chapter is organized as follows: in the first section, generic single-container OLPs are formulated and the CSO-VF algorithm devised to handle them is subsequently detailed. In the second section, the GA-assisted CSO-VF algorithm framework for solving previously formulated multi-container OLPs is defined. In each section, the formulation of the OLP is proposed, followed by the description of the proposed approaches. The application of the proposed methods will be introduced in Chapter 4.

3.2 CSO-VF: a quasi-physical approach for solving single-container optimal layout problems

3.2.1 Generic single-container optimal layout problems

3.2.1.1 Definition of the container and components

- The N components to position in the container can be of any shape: circles, rectangles, squares, polygons, *etc.*;
- The container corresponds to the available zone in which components must be positioned. In the same way, it could take on any shape. Moreover, exclusion zones can be added to the container. They correspond to areas where components cannot be positioned (*e.g.*, electric buses in complex systems, structural beams in facility layout problems).

Figure 3.1 illustrates different layout configurations.

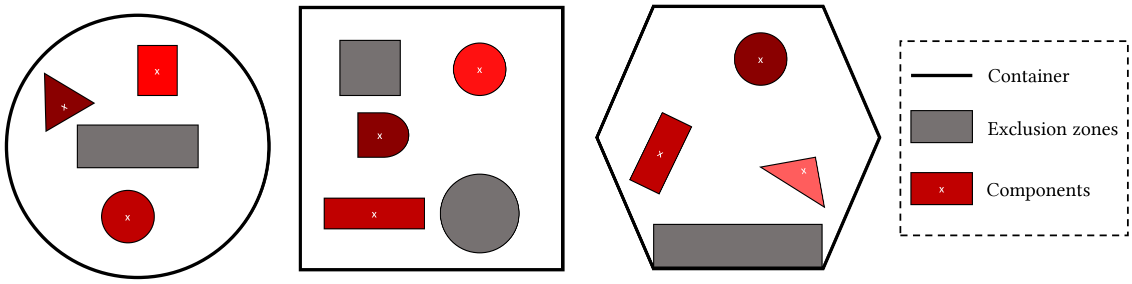


Figure 3.1: Illustration of various layout configurations. The redder the components, the heavier their corresponding masses. The white crosses correspond to the centers of inertia of the components.

3.2.1.2 Design variables

The design variables to optimize are:

- The positions of the center of inertia of each component which are considered as continuous variables in this thesis: $\mathbf{x}_p = \{\mathbf{x}_{p,i}\}$, for $i \in \{1, ..., N\}$, where $\mathbf{x}_{p,i} = \{x_{x,i}, x_{y,i}\}$;
- The orientations of non-circular components. According to the problem specifications, orientations may either be continuous or discrete variables: $\mathbf{x}_\alpha = \{x_{\alpha,i}\}$ or $\mathbf{z}_\alpha = \{z_{\alpha,i}\}$, for $i \in \{1, ..., N\}$.

3.2.1.3 Objective functions

The objective functions that can be optimized are for instance:

- Any dynamical requirements (*e.g.*, overall mass, inertia or stability of the system);
- Any costs related to the layout (*e.g.*, manufacturing or handling costs);
- Any performance of the layout (*e.g.*, power consumption or emission).

3.2.1.4 Constraints functions

The constraints that can be considered are listed and illustrated as follows:

- **Overlapping constraints.** They encompass three non-overlapping requirements: 1) A component must not overlap any other component 2) the components must all be contained within the container (*i.e*, they must be fully overlapping the container) 3) The components must not overlap the exclusion zones. Figure 3.2 illustrates these overlapping constraints.
- **Balancing constraint.** One or several coordinates of the center of gravity of the layout configuration must be positioned within a tolerance zone centered about a chosen position of reference.

- **Functional constraints.** Some components must be close or far from each other for functional reasons (*e.g.*, radiative tolerance of the components). Figure 3.4 illustrates the functional constraints. On the one hand, some components must be close from each other. A proximity zone is defined around one component and the other corresponding functional components must be positioned inside the proximity zone. On the other hand, some components must be spaced apart from each other. In the same way, a distancing zone is defined around one component and the other incompatible components must be situated outside of the distancing zone.

- Any additional constraints related to the geometry of the layout configuration or the functionality of the components can be added.

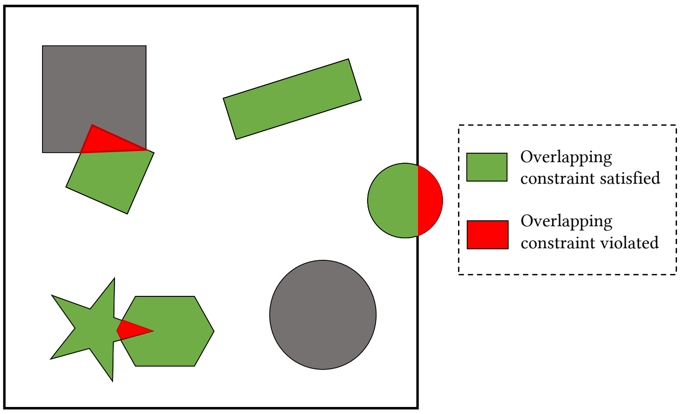


Figure 3.2: Illustration of the overlapping constraints. Components violating the overlap constraints are striped in red and the components that satisfy the overlapping constraints are striped in green.

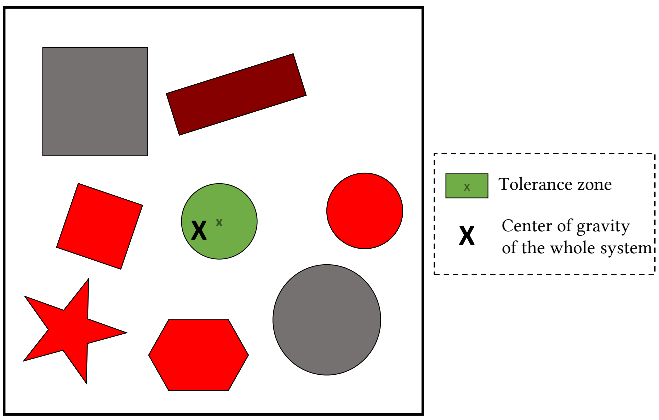


Figure 3.3: Illustration of the balancing constraints.

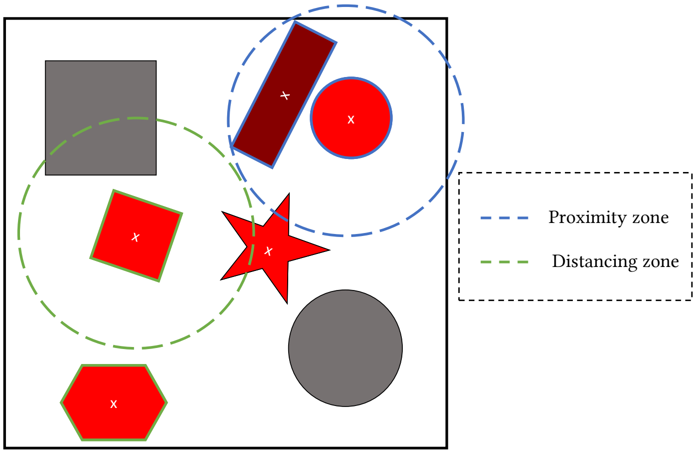


Figure 3.4: Illustration of the functional constraints. Components submitted to the proximity functional constraint are highlighted in blue while components submitted to the distancing functional constraint are highlighted in green.

3.2.2 Limits of existing virtual-force systems

Previous quasi-physical approaches based on virtual-force systems have been developed to solve packing problems. However, the virtual-force systems may present some limits for solving the OLPs described in the previous sections, for the main following reasons:

- **Geometry of the components:** Most of the existing virtual-force systems deal with circular components [He+13; HY11b; Wan+02; Xie+19]. Thus, polygonal components might not be directly laid out because the forces only act on the positions of their centers of inertia. Consequently, their orientations can not be updated.

- **Objective function(s):** The objective function of conventional packing problems tackled by existing virtual-force system (VFS) algorithms consists in minimizing the radius of the enclosing container. Moreover, this objective function is not optimized directly by the VFS. Indeed, the VFS optimizes the layout of the items within a fixed container and its size is decreased step by step (most of the time by dichotomy) if a feasible solution is found [He+13; HY11b; Wan+02]. Thus, existing VFSs do not encompass the minimization of the objective function. Then, they may be ineffective when it comes to minimizing objective function like for instance the global inertia of the system, the position of the center of gravity, any costs or power performance, *etc.*

- **Balancing constraint:** Most of the time, in existing quasi-physical approaches based on VFSs, the balancing constraint is tackled by relaxation [He+13; Wan+02]. In other words, the container is translated to the center of gravity of the components at the end of each iteration which ensures an automatic satisfaction of the balancing constraint. However, this technique can not be used when exclusion zones (*e.g.*, fixed components) belong to the container.

Indeed, a "shift" of the container to the center of gravity of the components is impossible as it also shifts the exclusion zones and may consequently results in the violation of the overlapping constraints between components and exclusion zones, at each iteration. Then, the existing VFSs [He+13; Wan+02] do not provide a balancing constraint satisfaction mechanism when exclusion zones are accounted for.

- **Other constraints:** Mostly overlapping constraints are considered in existing VFSs for packing problems [He+13; Wan+02]. They may not allow to tackle directly any constraint functions.

Given the aforementioned limits, a new quasi-physical approach based on a VFS is needed to solve the problems defined in Section 3.2.1. The following sections detail the proposed Component Swarm Optimization algorithm based on a Virtual-Force System which allows to overcome the previous VFS limitations and thus allows to tackle the formulated single-container OLPs.

3.2.3 Component Swarm Optimization algorithm based on a Virtual-Force System (CSO-VF)

The proposed quasi-physical approach for solving the described optimal layout problems is the Component Swarm Optimization algorithm based on a Virtual-Force system (CSO-VF). The main focus of the algorithm is to define dedicated operators for the evolution of a dynamical system of the components based on the fundamental principle of dynamics to efficiently satisfy the constraints while minimizing the objective function. In the CSO-VF algorithm, each component is assumed to be a particle in a swarm. At each iteration of the algorithm, depending of the virtual forces that are applied to the particle, each of them moves within the container until the objective function is minimized and all constraints are satisfied.

It is important to note that this type of algorithm differs from classical Particle Swarm Optimization (PSO) algorithms. Indeed, in the CSO-VF algorithm, each particle of the swarm corresponds to a single component and thus, to a subset of the entire solution while in PSO algorithms, a particle corresponds to an entire solution *i.e.*, an entire layout.

3.2.3.1 Virtual-force system

In the CSO-VF algorithm, each component i is described by its dynamic features:

- Its translational and rotational accelerations $\mathbf{a}_i$ and $\mathbf{L}_i$;
- Its translational and rotational speeds $\mathbf{v}_i$ and $\boldsymbol{\omega}_i$;
- Its position $\mathbf{x}_{p,i}$ and orientation $\mathbf{x}_{\alpha,i}$.

N_F forces ($\mathbf{F}_i^k, k \in \{1, ..., N_F\}$, for component i) with a resultant $\mathbf{F}_i$ as well as N_T torques ($\mathbf{T}_i^k, k \in \{1, ..., N_T\}$, for component i) with a resultant $\mathbf{T}_i$ are applied to the component in order to move it at each iteration as illustrated on Figure 3.5. It must be noted that torques are applied to non-circular components only.

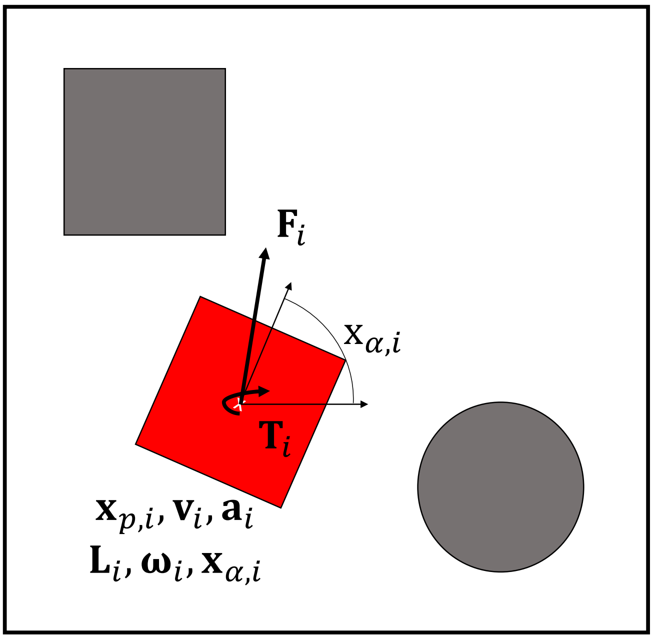


Figure 3.5: Definition of a component.

By the end of an iteration, the resulting force $\mathbf{F}_i$ and torque $\mathbf{T}_i$ are calculated for each component using Equations 3.1 and 3.2. The Fundamental Principles of the Dynamics of rotation and translation are applied in order to update the positions and orientations of the swarm of components at step $t+1$ (separated from step t by Δt which corresponds to a time unit) according to Equations 3.3, 3.4 and 3.5 (detailed for component i with mass m_i and solid inertia I_i).

$$\mathbf{F}_i = \begin{cases} \displaystyle\sum_{k=1}^{N_F} \mathbf{F}_i^k & \text{if } \left\| \displaystyle\sum_{k=1}^{N_F} \mathbf{F}_i^k \right\| \leq F_{max} \\ \dfrac{\sum_{k=1}^{N_F} \mathbf{F}_i^k}{\left\| \sum_{k=1}^{N_F} \mathbf{F}_i^k \right\|} F_{max} & \text{otherwise.} \end{cases} \tag{3.1}$$

$$\mathbf{T}_i = \begin{cases} \displaystyle\sum_{k=1}^{N_T} \mathbf{T}_i^k & \text{if } \left\| \displaystyle\sum_{k=1}^{N_T} \mathbf{T}_i^k \right\| \leq T_{max} \\ \dfrac{\sum_{k=1}^{N_T} \mathbf{T}_i^k}{\left\| \sum_{k=1}^{N_T} \mathbf{T}_i^k \right\|} T_{max} & \text{otherwise.} \end{cases} \tag{3.2}$$

where F_{max} and T_{max} are hyperparameters corresponding to the maximum value of the norm of the resulting force and torque vectors, respectively.

$$\mathbf{a}_{i,t+1} = \frac{\mathbf{F}_i}{m_i} \quad (3.3)$$

$$\mathbf{v}_{i,t+1} = \mathbf{v}_{i,t} + \mathbf{a}_{i,t+1}\Delta t \quad (3.4)$$

$$\mathbf{x}_{p,i,t+1} = \mathbf{x}_{p,i,t} + \mathbf{v}_{i,t+1}\Delta t \quad (3.5)$$

$$\mathbf{L}_{i,t+1} = \frac{\mathbf{T}_i}{I_i} \quad (3.6)$$

$$\boldsymbol{\omega}_{i,t+1} = \boldsymbol{\omega}_{i,t} + \mathbf{L}_{i,t+1}\Delta t \quad (3.7)$$

$$\mathbf{x}_{\alpha,i,t+1} = \mathbf{x}_{\alpha,i,t+1} + \boldsymbol{\omega}_{i,t+1}\Delta t \quad (3.8)$$

Each of the forces and torques of the VFS aims at solving for the constraints and optimizing the objective function(s). If the objective or constraint functions take as arguments the positions of the components, a force is applied. If they take as arguments the orientations of the components, a torque is applied.

The forces and torques applied to a component i are employed to:

- Minimize an objective function: force $\mathbf{F}_i^{objective}$ and torque $\mathbf{T}_i^{objective}$;
- Solve for the overlapping constraints between components i and j: force $\mathbf{F}_{ij}^{overlap}$ and torque $\mathbf{T}_{ij}^{overlap}$;
- Solve for the overlapping constraints between component i and an exclusion zone (EZ): force $\mathbf{F}_{i/EZ}^{overlap}$ and torque $\mathbf{T}_{i/EZ}^{overlap}$;
- Solve for the overlapping constraints between component i and the container: force $\mathbf{F}_i^{container}$ and torque $\mathbf{T}_i^{container}$;
- Solve for the functional constraints between two components i and j: force $\mathbf{F}_i^{functional}$ and torque $\mathbf{T}_i^{functional}$;
- Solve for the balancing constraint: force $\mathbf{F}_i^{CG}$ (CG stands for center of gravity);
- More generally, to solve for any constraint function: force $\mathbf{F}_i^{constraint}$ and torque $\mathbf{T}_i^{constraint}$.

Generally speaking, any force or torque can be added to the VFS in order to address additional constraints or objective functions.

The forces and torques of the virtual-force system are detailed and formulated as follows (for a component i):

- **The overlapping constraint force and torque between two components:** If two components i and j are overlapping each other, repulsive forces $\mathbf{F}_{ij}^{overlap}$ and $\mathbf{F}_{ji}^{overlap}$ are applied to each of them as illustrated in Figures 3.6 and 3.7. The overlap force is expressed as:

$$\mathbf{F}_{ij}^{overlap} = \begin{cases} -\dfrac{\mathbf{x}_{p,j} - \mathbf{x}_{p,i}}{\|\mathbf{x}_{p,j} - \mathbf{x}_{p,i}\| + \epsilon} v_{max} - \mathbf{v}_i & \text{if } \Delta S_{ij}(\mathbf{x}_{p,i}, \mathbf{x}_{p,j}) \neq 0, \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (3.9)$$

where v_{max} is a hyperparameter corresponding to the maximum value of the norm of the speed vector, ϵ ensures numerical stability and $\mathbf{0} = (0,0)$ is the null vector. ΔS_{ij} is the area of intersection between components i and j.

Moreover, similar to [SCL22], additional torques are applied to non-circular components in order to solve the overlapping constraint as illustrated on Figure 3.6.

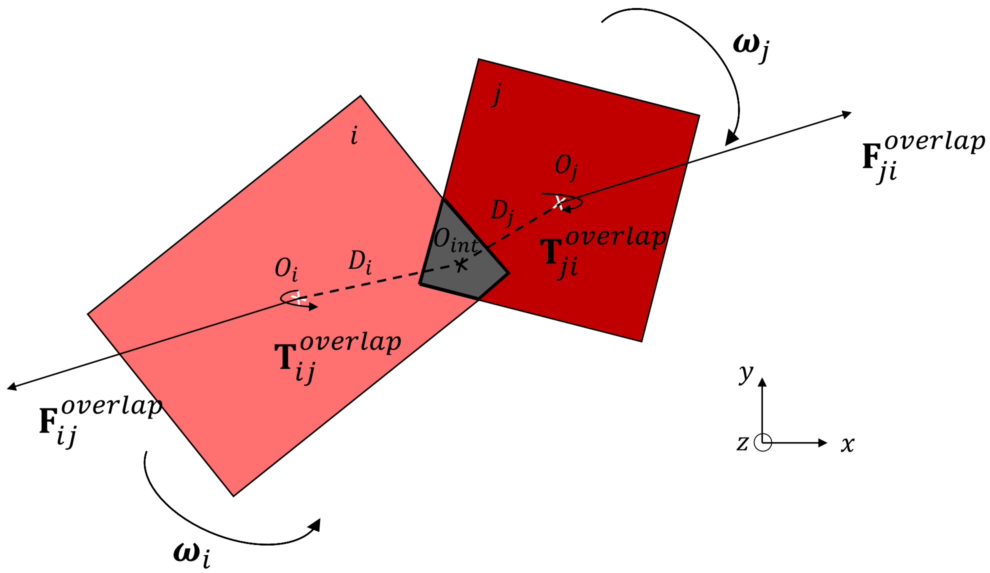


Figure 3.6: Definition of the overlapping forces and torques.

The overlapping forces are initially applied to the O_{int} point which corresponds to the geometrical center of the polygon of intersection between the two components. Then, the mechanical action torsors for components i and j calculated at the center of inertia of each component O_i and O_j are expressed as:

$$\{\mathcal{T}_i\} = \left\{ \begin{array}{c} \mathbf{F}_{ij}^{overlap} \\ \mathbf{0} \end{array} \right\}_{i/O_{int}} = \left\{ \begin{array}{c} \mathbf{F}_{ij}^{overlap} \\ \mathbf{T}_{ij}^{overlap} = \mathbf{O_i O_{int}} \wedge \mathbf{F}_{ij}^{overlap} \end{array} \right\}_{i/O_i} \tag{3.10}$$

$$\{\mathcal{T}_j\} = \left\{ \begin{array}{c} \mathbf{F}_{ji}^{overlap} \\ \mathbf{0} \end{array} \right\}_{j/O_{int}} = \left\{ \begin{array}{c} \mathbf{F}_{ji}^{overlap} \\ \mathbf{T}_{ji}^{overlap} = \mathbf{O_j O_{int}} \wedge \mathbf{F}_{ji}^{overlap} \end{array} \right\}_{j/O_i} \tag{3.11}$$

where $||\mathbf{O_i O_{int}}|| = D_i$ and $||\mathbf{O_j O_{int}}|| = D_j$.

Then, the resulting torque applied at point O_i is given by:

$$\mathbf{T}_{ij}^{overlap} = \mathbf{O_i O_{int}} \wedge \mathbf{F}_{ij}^{overlap} \tag{3.12}$$

- **The overlapping constraint force and torque between a component and an exclusion zone:** If a component is overlapping an exclusion zone, a repulsive force $\mathbf{F}_{i/EZ}^{overlap}$ is applied to the component as illustrated in Figure 3.7. This force is expressed as:

$$\mathbf{F}_{i/EZ}^{overlap} = \begin{cases} -\dfrac{\mathbf{x}_{p,EZ} - \mathbf{x}_{p,i}}{\|\mathbf{x}_{p,EZ} - \mathbf{x}_{p,i}\| + \epsilon} v_{max} - \mathbf{v}_i, & \text{if } \Delta S_{i/EZ}(\mathbf{x}_{p,i}, \mathbf{x}_{p,EZ}) \neq 0, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3.13}$$

where $\mathbf{x}_{p,EZ}$ corresponds to the position vector of the considered exclusion zone. $\Delta S_{i/EZ}$ is the intersection area between component i and the exclusion zone. Similarly to the previous constraint, the related torque is calculated as follows:

$$\mathbf{T}_{i/EZ}^{overlap} = \mathbf{O_i O_{int}} \wedge \mathbf{F}_{i/EZ}^{overlap} \tag{3.14}$$

where O_i is the center of inertia of component i and O_{int} is the geometrical center of the polygon of intersection between component i and the exclusion zone.

- **The overlapping constraint force and torque between a component and the container:** If a component is not fully overlapping the container, then an attractive force $\mathbf{F}_i^{container}$ is applied to the component as illustrated in Figure 3.7. The container force is expressed as:

$$\mathbf{F}_i^{container} = \begin{cases} \dfrac{\mathbf{x}_{p,C} - \mathbf{x}_{p,i}}{\|\mathbf{x}_{p,C} - \mathbf{x}_{p,i}\| + \epsilon} v_{max} - \mathbf{v}_i, & \text{if } \Delta S_{i/C}(\mathbf{x}_{p,i}, \mathbf{x}_{p,C}) < S_i, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3.15}$$

where $\mathbf{x}_{p,C}$ corresponds to the position vector of the geometrical center of the container. $\Delta S_{i/C}$ is the intersection area between component i and the exclusion zone and S_i is the surface of component i. The related torque is calculated as follows:

$$\mathbf{T}_i^{container} = \mathbf{O_i O_{out}} \wedge \mathbf{F}_i^{container} \tag{3.16}$$

where O_i is the center of inertia of component i and O_{out} is the geometrical center of the area of component i situated outside the container.

- **The functional constraint force and torque between two components:** If a component i is too close to an incompatible component j, a repulsive force $\mathbf{F}_i^{functional}$ is applied to component i as illustrated on Figure 3.8. The functional force acts as an overlapping force between component i and the influence zone of component j. Then, the functional force is expressed as:

$$\mathbf{F}_i^{functional} = \begin{cases} -\dfrac{\mathbf{x}_{p,j} - \mathbf{x}_{p,j}}{\|\mathbf{x}_{p,j} - \mathbf{x}_{p,i}\| + \epsilon} v_{max} - \mathbf{v}_i, & \text{if } \Delta S_{i/SZj}(\mathbf{x}_{p,i}, \mathbf{x}_{p,j}) \neq 0, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3.17}$$

where $\Delta S_{i/SZj}$ is the intersection area between component i and the functional security zone surrounding component j.

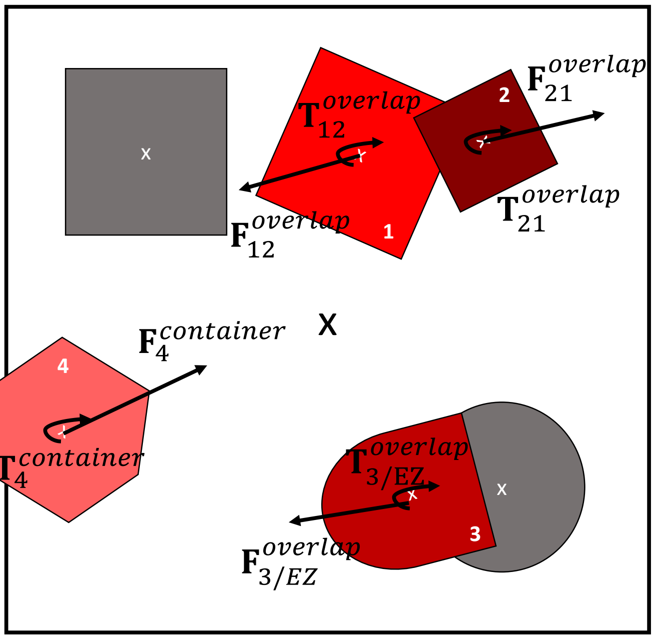


Figure 3.7: Illustration of the forces and torques to solve overlapping constraints between two components, between a component and an exclusion zone and between a component and the container.

The related torque is calculated as follows:

$$\mathbf{T}_i^{functional} = \mathbf{O_i O_{zj}} \wedge \mathbf{F}_i^{functional} \tag{3.18}$$

where O_i is the center of inertia of component i and O_{zj} is the geometrical center of the area of component i overlapping the functional security zone surrounding component j.

- **The balancing constraint force:** In order to position the center of mass of the components in a tolerance zone centered at the geometrical center of the container, gradient-based forces are applied along the opposite of the gradient of the position of the global center of mass according to the position of the center of inertia of each component. This force is named $\mathbf{F}_i^{CG}$. It is illustrated in Figure 3.9. The balancing force is expressed as:

$$\mathbf{F}_i^{CG} = \begin{cases} -\alpha_{CG} \nabla \mathbf{h}_{CG}^F(\mathbf{x}_{p,i}), & \text{if } \sqrt{(x_{CG}-x_e)^2 + (y_{CG}-y_e)^2} \leq \delta_{CG}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3.19}$$

where α_{CG}^F is a step-size hyperparameter of the algorithm and $\nabla \mathbf{h}_{CG}(\mathbf{x}_{p,i})$ corresponds to the gradient of the position of the center of gravity of the components with respect to the position of the considered component i. $\{x_{CG}, y_{CG}\}$ are the coordinates of the current center of gravity, $\{x_e, y_e\}$ are the coordinates of the position of reference on which the tolerance zone defined by δ_{CG} is centered at. This type of force is inspired by gradient-based descent algorithms.

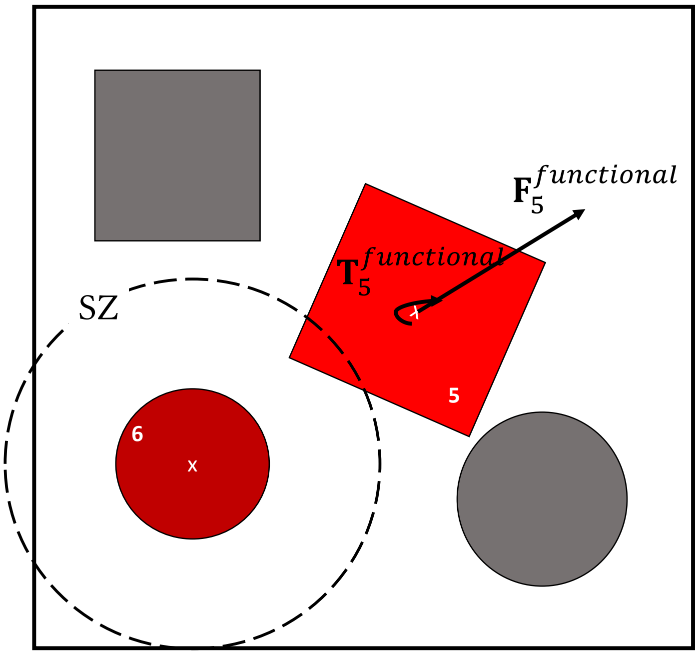


Figure 3.8: Illustration of the functional force and torque.

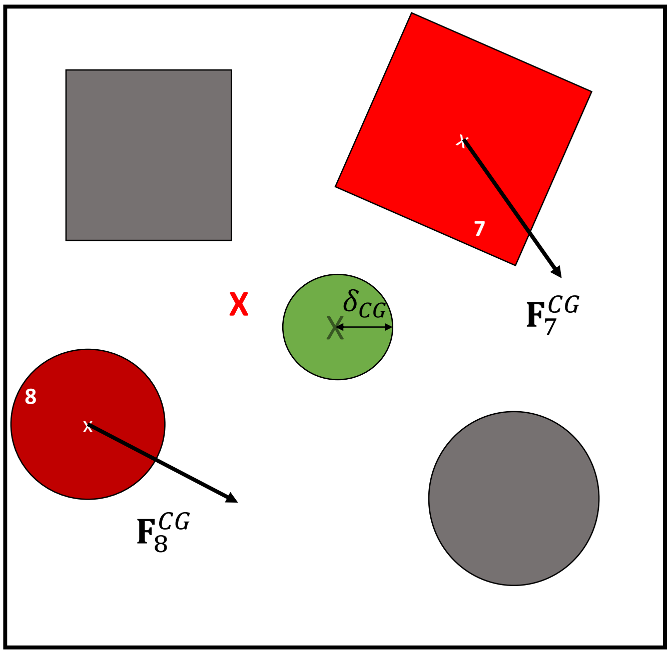


Figure 3.9: Illustration of the balancing forces. The green area corresponds to the tolerance zone where the current center of gravity must be positioned. The red cross is the current center of gravity of the components.

- **The differentiable constraint function force and torque:** Generally, any differentiable constraint function (DF) can be addressed thanks to gradient-based forces and torques, expressed as:

$$\mathbf{F}_i^{DF} = -\alpha_{DF}^{F} \nabla \mathbf{h}_{DF}(\mathbf{x}_{p,i}) \tag{3.20}$$

$$\mathbf{T}_i^{DF} = -\alpha_{DF}^{T} \nabla \mathbf{h}_{DF}(\mathbf{x}_{\alpha,j}) \tag{3.21}$$

  for constraints expressed as $\mathbf{g}(\cdot) \leq 0$ and where α_{DF}^{F} and α_{DF}^{T} are "step-size" hyperparameters of the algorithm. $\nabla \mathbf{h}_{DF}(\mathbf{x}_{p,i})$ corresponds to the gradient of any differentiable constraint function with respect to the position of the considered component i. $\nabla \mathbf{h}_{DF}(\mathbf{x}_{\alpha,j})$ corresponds to the gradient of any differentiable constraint function with respect to the orientation of the considered component i.

- **The objective function forces and torques:** To minimize one objective function f, a gradient-based force and torque can be applied. The objective function force and torque are expressed as:

$$\mathbf{F}_i^{obj} = -\alpha_{obj}^{F} \nabla f_{obj}(\mathbf{x}_{p,i}) \tag{3.22}$$

$$\mathbf{T}_i^{obj} = -\alpha_{obj}^{T} \nabla f_{obj}(\mathbf{x}_{\alpha,i}) \tag{3.23}$$

  where α_{obj}^{F} and α_{obj}^{T} are "step-size" hyperparameters of the algorithm. $\nabla f_{obj}(\mathbf{x}_{p,i})$ corresponds to the gradient of the objective function with respect to the position of the considered component i and $\nabla f_{obj}(\mathbf{x}_{\alpha,i})$ corresponds to the gradient of the objective function with respect to the orientation of the considered component.

It must be noted that in the case where the orientation variables are defined as discrete (*i.e.*, non-cylinder components taking orientation values in a defined subset), they are handled as relaxed continuous orientations. The updated orientation is calculated as a continuous variable and a threshold is set to define the discrete orientation from the subset. For instance, in the case where the non-cylinder components take their orientation in the subset (0°,90°), if the positive updated continuous orientation is greater (respectively lower) than 45° the updated discrete orientation is 90° (respectively 0°).

3.2.3.2 Swap operator

In [FHL13; Wan+19b; Zen+16], swap operators are proposed when solving packing problems. They mainly consist in exchanging the positions of items with similar dimensions with a certain probability when convergence reaches a plateau, that is, when the algorithm stagnates as a result of, perhaps, opposing forces. The swap operator introduced in the CSO-VF is an extended version of the previous swap operators. The swap operator exchanges the positions of a pair of components after a given number of iterations if it is improving the objective function(s) or decreasing the violation of some selected constraint(s). Thus, the swap operator can be used to find promising configurations in terms of constraint satisfaction and objective function and therefore, does not necessarily need the configuration to be somewhat stuck in order to be employed in CSO-VF algorithm. For instance, the swap operator can be used to exchange components such that the global center of gravity of the system quickly enters the given tolerance zone as illustrated on Figure 3.10.

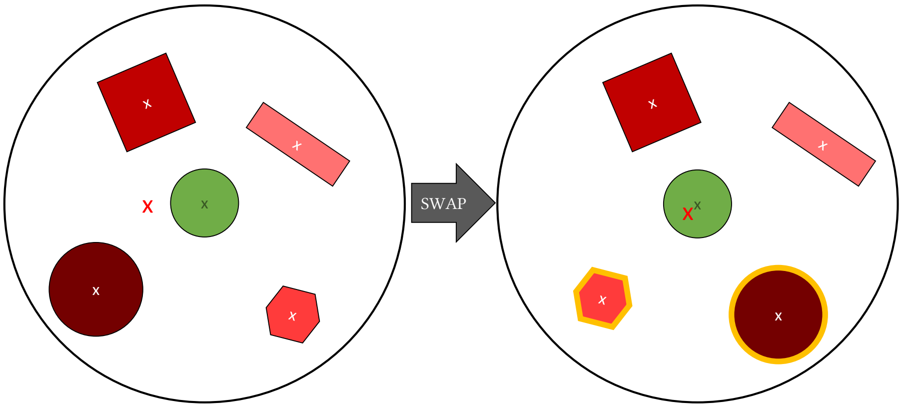


Figure 3.10: Example of the swap operator employed to improve the balancing constraint. The green area is the tolerance zone in which the current center of gravity of the components, illustrated thanks to the red cross, must be placed. The redder the components, the heavier their corresponding masses. The swapped components are highlighted in yellow.

Similarly, exchanging two components at any time of the convergence can improve the global inertia as illustrated in Figure 3.11. In this figure, two swaps occur allowing to position heavier components closer to the geometrical center of the module and therefore leading to a smaller global inertia of the layout.

Thus, the swap operator implemented in the CSO-VF algorithm has the following characteristics:

- The swap operator can exchange all the components by pairs, not only similar ones;
- Two components are swapped if the swap leads to an improvement of the objective function or a decrease in the violation of some chosen constraint(s), while not deteriorating the other constraint(s) from a relaxation factor r;
- The swap operator is called straight from the first iteration. Then, it can occur throughout the optimization process.

The swap operator is called at a frequency depending on the convergence status. Indeed, the swap operator induces a reconfiguration. Consequently, too frequent swaps might prevent the layout from converging. On the contrary, too spaced swaps might lead to configurations being stucked for too many iterations and thus to a

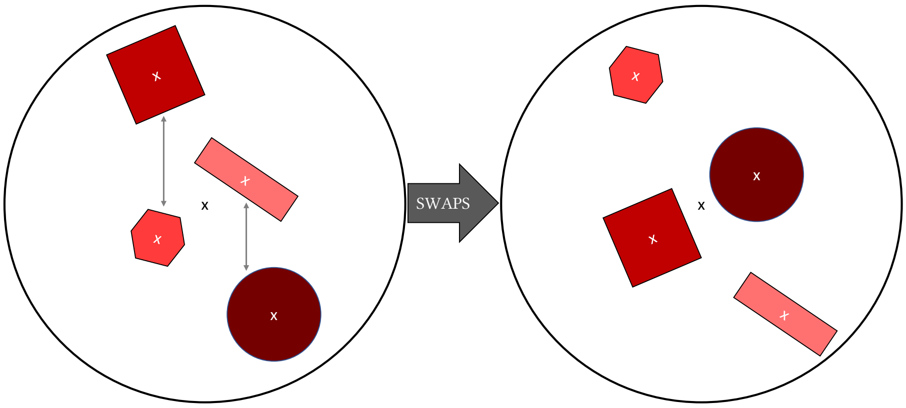


Figure 3.11: Example of the swap operator employed to improve the global inertia of the layout.

stagnating or slow convergence rate. The minimal step between two iterations during which the swap is called is s_{min} and the maximum step is s_{max}. Each time the swap operator is used, it can only be applied at a subsequent iteration which is calculated based on s_{min} and s_{max}, following the procedure:

- While no feasible layout is found, the step is fixed to s_{min}. Then, at any current iteration t_c during which the swap operator is employed, the next swap iteration noted t_{swap} is calculated as:

$$t_{swap} = t_c + s_{min} \tag{3.24}$$

  Consequently, the swap operator is called at a lower frequency and help finding a feasible solution.

- Once a feasible solution is found, each time the swap operator is called, the convergence curve is interpolated and the next iteration of the swap operator is calculated as:

$$t_{swap} = t + (1-\gamma)s_{min} + \gamma s_{max} \tag{3.25}$$

  where γ is a factor characterizing the convergence state as:

$$\gamma = -\frac{\arctan\left(\left.\frac{\mathrm{d}f_{interp}(t)}{\mathrm{d}t}\right|_{t=t_c}\right)}{\pi/2} \tag{3.26}$$

  where f_{interp} is the interpolation function of the convergence curve. Therefore, at the beginning of the convergence, the derivative of the convergence curve might be high and γ close to 1 and thus the step size close to s_{max}. On the contrary, when the convergence stagnates, the factor γ may be close to 0 and the swap is called more frequently. Consequently, as soon as a feasible solution is found, the swap operator is employed to explore new configurations when convergence is stagnating and when the layout is not improving anymore.

Algorithm 3.1: gives the pseudo code of the swap operator in the case where the swap operator is used to improve the objective function f_{obj} to minimize. It can be summarized as follows: if the current iteration corresponds to t_{swap}, the swap operator is called. If the objective function is improved and if the corresponding constraint violation CV_{Swap} of the N_{cst} constraints functions ($\mathbf{g}_k$ for $k \in \{1, ..., N_{cst}\}$) remains lower than or equal to the previous constraint violation CV_{NoSwap} times the relaxation factor r, then the swap between the two components occurs.

Algorithm 3.1: The Swap Operator

Input: t_{swap}, r, $\mathbf{p}$
Output: new vector of positions $\mathbf{p}$
if t is t_{swap} **then**
  for $i = 0$ **to** N **do**
    for $j = i + 1$ **to** N **do**
      $\mathbf{x}_{p,swap} \leftarrow \mathbf{x}_p$ with $\mathbf{x}_{p,i}$ and $\mathbf{x}_{p,j}$ exchanged.
      $CV_{Swap} \leftarrow 0$
      $CV_{NoSwap} \leftarrow 0$
      for $k = 0$ **to** N_k **do**
        $CV_{Swap} = CV_{Swap} + \mathbf{g}_k(\mathbf{x}_{p,swap})$ #*Constraint violation if the swap occurs*

        $CV_{NoSwap} = CV_{NoSwap} + \mathbf{g}_k(\mathbf{x}_p)$ #*Constraint violation if not*
      end for
      if $f_{obj}(\mathbf{x}_{p,swap}) < f_{obj}(\mathbf{x}_p)$ **and** $CV_{Swap} \leq r \times CV_{NoSwap}$ **then**
        $\mathbf{x}_p \leftarrow \mathbf{x}_{p,swap}$
      end if
    end for
  end for
  Update t_{swap}
end if

3.2.3.3 Initialization

As the CSO-VF algorithm locally refines a solution for non-convex problems with multiple local minima, the initialization of the design variables might have a critical impact on the optimization process. Rather than a random initialization of the positions and orientations of the components, optimized Latin Hypercube Sampling (LHS) is employed [DCI13; Li+17; PM12]. In its original formulation, LHS distributes N (*i.e.*, the number of components) sample points within equiprobable intervals of the search space [Loh96]. LHS techniques have been improved by minimizing a space filling criterion. Among existing space filling criteria, the centered L^2-discrepancy called C_2 [FLS06] is considered and is minimized using a Simulated Annealing method. Figure 3.12 shows an optimized LHS and random initializations of 10 samples within a two-dimensional square domain in the range [0, 10].

Thus, the LHS distributes the samples in a more uniform way in the search space. In the case of the optimal layout problems, this technique should improve the constraint resolution as well as providing a feasible solution in more rapidly *i.e.*,

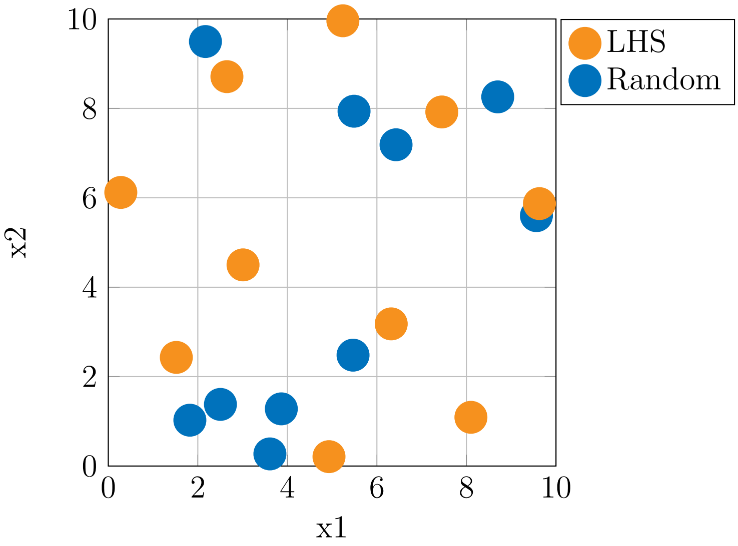


Figure 3.12: Optimized LHS and random initializations of 10 samples.

in less iterations than would otherwise be necessary with a random initialization.

Moreover, as each instance of the algorithm depends on the initialization, a multi-start technique is employed. Indeed, N_{start} independent initializations are run and the output of the CSO-VF algorithm corresponds to the best layout obtained out of the N_{start} initializations.

3.2.3.4 Hyperparameters

The hyperparameters of the CSO-VF algorithm are provided below:

- Dynamic features: maximum values of the norm of the force and torque vectors F_{max} and T_{max} as well as maximum values of the norm of the translational and rotational speed vectors v_{max} and ω_{max};

- The gradient-based forces and torques scale. Their total number depends on the virtual-force system;

- The swap operator parameters: r, s_{min}, s_{max};

- The number of multistarts N_{start}.

Figure 3.13 describes one instance of the CSO-VF algorithm.

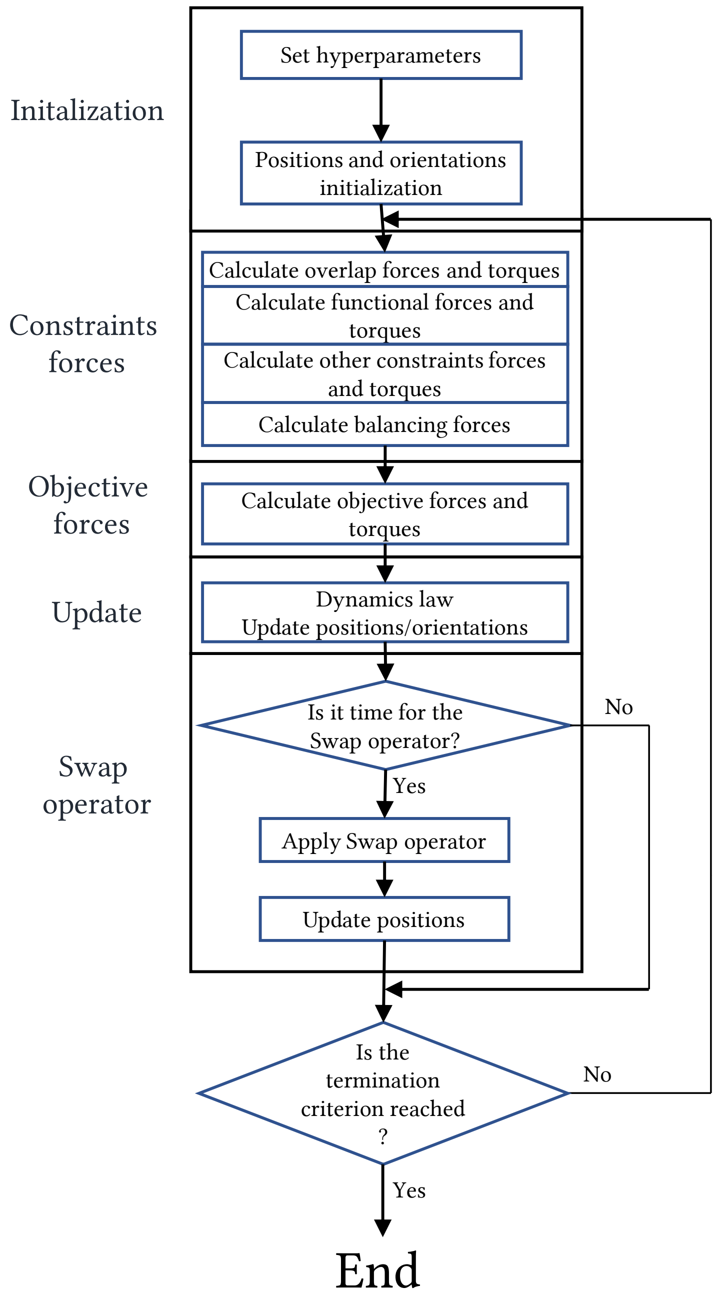


Figure 3.13: The CSO-VF algorithm.

3.3 Two-stage approach combining Genetic Algorithm and CSO-VF for multi-container optimal layout problems

3.3.1 Generic multi-container optimal layout problems

Multi-container OLPs consist in assigning N components to n containers along with optimizing their layout in the assigned containers. In other words, two tasks must be solved in order to obtain an optimal layout: the assignment task and the layout task. The assignment task consists in assigning each component to a container. The layout task consists in optimizing the layout of the assigned components in each container. The following sections are dedicated to the description and formulation of such problems.

3.3.1.1 Geometry of the components and containers

Multi-container OLPs involve:

- A set of N components of any shape;
- A set of n containers of any shape.

In this manuscript, it is considered that all the components can be positioned in the containers. In other words, the areas of all the components is smaller than the available area of all the containers. This condition is translated mathematically as follows:

$$\sum_{i=1}^{N} A_i^{comp} \leq \sum_{j=1}^{n} A_j^{cont} \tag{3.27}$$

where A_i^{comp} is the area of component i and A_j^{cont} is the area of container j.

Figure 3.14 illustrates the multi-container OLPs.

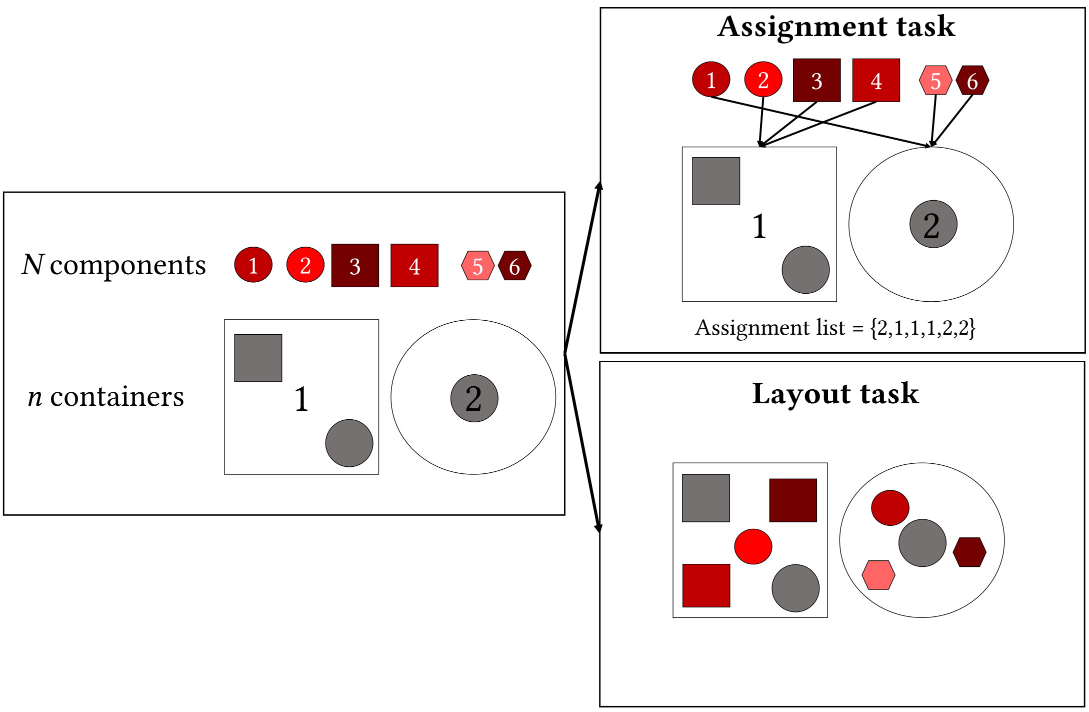


Figure 3.14: Illustration of the multi-container OLPs.

3.3.1.2 Design variables

The design variables that require to be optimized are the following:

- The index of the container assigned to each component, which corresponds to discrete unordered variables *i.e.*, categorical variables, $\mathbf{z}_c = \{z_{c,i}\}$ with $z_{c,i} \in \{1, ..., n\}$ for $i \in \{1, ..., N\}$;
- The positions of the center of inertia of each component in its assigned container which are considered as continuous variables $\mathbf{x}_p$;
- The orientations of the components whose shape is not circular. According to the problem specifications, design variables relative to components orientations can correspond either to continuous variables or to discrete variables. For illustrative purposes, orientations are considered continuous variables in this section: $\mathbf{x}_\alpha$.

If N components must be laid out in n containers, and that there are N_{cyl} cylinders among the components, the number N_{DV} of design variables is:

$$N_{DV} = 3N_{cyl} + 4(N - N_{cyl}) \tag{3.28}$$

Indeed, cylinders are defined by their assigned container and their position coordinates within the container which results in 3 design variables while the orientations of non-circular components must also be optimized resulting in 4 design variables instead.

3.3.1.3 Formulations of multi-container optimal layout problems

Multi-container OLPs can be split into different categories depending on their mathematical formulation. Two cases are considered:

- **Case 1:** In various multi-container OLPs, the assignment and layout tasks are not separable. In other words, both tasks are solved simultaneously by optimizing an unique objective function taking as arguments all design variables. For instance, in [WA19], the authors studied the multi-floor warehouse layout problem taking into consideration the assignment of the cross-dock door at the warehouse location. They presented an integrated mixed-integer programming model that solves the warehouse layout problem and cross-dock door assignment decisions simultaneously to minimize the total material handling cost. In [HLG21], a multi-floor hospital facility layout problem with a double-row layout on each floor has been solved. The problem has been addressed in an unique optimization process with two objectives: minimizing the total movement distance of patients and maximizing the total closeness rating score. Ahmadi *et al.* [APJ17] provided a review on multi-floor OLPs. Among the reported techniques, the single-stage algorithms encompass the methods in which all decisions are taken at one phase, including the assignment of departments to floors and specifying their locations [BT04; Hos+14; LRJ05].

Therefore, the optimization process is defined thanks to the global objective function written $f_{global}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c)$. The equality and inequality constraints related to the assignment are written $\mathbf{h}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c)$, $\mathbf{g}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c)$ and the equality and inequality constraints functions related to the layout are written $\mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c)$, $\mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c)$. The assignment constraints can for instance correspond to maximum occupation rates of the container [AJ16]. Functional constraints can also be considered if some components must be positioned in the same container for functional reasons or on the contrary be assigned to different containers [KSN22]. The layout constraints can correspond to all the constraints defined for single-container OLPs in Section 3.2.1.4. The problem at hand can then be mathematically formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c} \quad & f_{global}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \\ \text{w.r.t.} \quad & \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}}, \mathbf{x}_\alpha \in F_{x_\alpha} \subseteq \mathbb{R}^{n_{x_\alpha}}, \mathbf{z}_c \in F_{z_c} \subseteq \mathbb{N}^{n_{z_c}} \\ \text{s.t.} \quad & \mathbf{h}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) = 0 \\ & \mathbf{g}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \leq 0 \\ & \mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) = 0 \\ & \mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \leq 0 \end{aligned} \tag{3.29}$$

- **Case 2:** In other multi-container OLPs, the assignment and layout tasks can be solved in two different optimization processes. Indeed, the additional design variables and constraints of such problems induce a more complex design space than single-container problems. For N components to be laid out in n containers, there are n^N possible assignment schemes. Solving both the assignment task and the n containers corresponding layouts in the same optimization process might result in poor resolution performance in terms of ability to solve the constraints, convergence speed and convergence accuracy due to the highly dimensional, combinatorial and constrained characteristics of the problem [APJ17]. Thus, another possible approach consists in decomposing the problem into two optimization processes dealing with respectively the assignment task (*i.e.* the upper stage) and the layout task (*i.e.* the lower stage). The problem can then be defined as bi-objective with two sets of constraints related to each task. Two formulations can be derived from the multi-objective definition:

  - **Case 2.1: Nested formulation.**
    The layout optimization process is defined using objective and constraint functions which depend on the assignment scheme as well as the $\mathbf{x}_p$, $\mathbf{x}_\alpha$ design variables optimized in this lower level: $f_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c^*)$, $\mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c^*)$, $\mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c^*)$. In the previous notations, the * exponent is employed to highlight the design variables which are not optimized in the corresponding optimization process. Considering the assignment task level, the objective and constraint functions are written $f_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c)$, $\mathbf{h}_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c)$, $\mathbf{g}_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c)$. Then, the mathematical for-

mulation is written:

$$\begin{aligned}
\min_{\mathbf{z}_c} \quad & f_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c) \\
\text{s.t.} \quad & \mathbf{h}_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_\mathbf{c}) = 0 \\
& \mathbf{g}_{assignment}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c) \leq 0 \\
\text{w.r.t.} \quad & \mathbf{z}_c \in F_z \subseteq \mathbb{N}^{n_{z_c}} \\
& \{\mathbf{x}_p^*, \mathbf{x}_\alpha^*\} = \text{argmin } f_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \\
& \text{w.r.t.} \quad \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}}, \mathbf{x}_\alpha \in F_{x_\alpha} \subseteq \mathbb{R}^{n_{x_\alpha}} \\
& \text{s.t.} \quad\;\; \mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) = 0 \\
& \qquad\quad\;\; \mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \leq 0
\end{aligned} \tag{3.30}$$

The two optimization processes are therefore solved sequentially. Algorithms like the surrogate assisted-CSO-VF algorithm developed in Chapter 6 can then be used to address this kind of problem formulation.

- **Case 2.2: Two-stage formulation.**
  In most of the cases addressed in the literature, the assignment and layout tasks can be fully separated. Consequently, the objective function related to the assignment only relies on the design variables of the assignment scheme, while the layout objective function takes as argument the $\mathbf{x}_p$ and $\mathbf{x}_\alpha$ layout design variables as well as the assignment scheme found by the upper loop and described by the $\mathbf{z}_c$ variables.
  The corresponding mathematical formulation is as follows:

$$\begin{aligned}
\min_{\mathbf{z}_c} \quad & f_{assignment}(\mathbf{z}_c) \\
\text{s.t.} \quad & \mathbf{h}_{assignment}(\mathbf{z}_\mathbf{c}) = 0 \\
& \mathbf{g}_{assignment}(\mathbf{z}_c) \leq 0 \\
& \mathbf{h}^*_{layout}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c) = 0 \\
& \mathbf{g}^*_{layout}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{z}_c) \leq 0 \\
\text{w.r.t.} \quad & \mathbf{z}_c \in F_z \subseteq \mathbb{N}^{n_{z_c}} \\
& \{\mathbf{x}_p^*, \mathbf{x}_\alpha^*\} = \text{argmin } f_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \\
& \text{w.r.t.} \quad \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}}, \mathbf{x}_\alpha \in F_{x_\alpha} \subseteq \mathbb{R}^{n_{x_\alpha}} \\
& \qquad\quad\;\; \mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) = 0 \\
& \qquad\quad\;\; \mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c) \leq 0
\end{aligned} \tag{3.31}$$

  where $f_{assignment}$, $\mathbf{h}_{assignment}$ and $\mathbf{g}_{assignment}$ are respectively the objective function, the equality and the inequality constraints functions related to the assignment taking as argument only the $\mathbf{z}_c$ design variables. f_{layout} is the objective function related to the layout and taking as arguments all design variables. $\mathbf{h}^*_{layout}$ and $\mathbf{g}^*_{layout}$ ensures that a feasible layout has been found by the lower loop.
  Ahmadi *et al.* [APJ17] reviewed multi-stage formulations of multi-floor facility layout problems in which the first stage was dedicated to the

assignation of each facility to one floor and the subsequent levels were dedicated to the layout of the floors as well as other details (*e.g.* determining the number of elevators or the dimensions of certain facilities) [AJ16; BA12; CL06]. Other recent multi-floor facility layout problems decomposed in two- or three-stage optimization processes are proposed in [KSE17; KSN22; MB97]. In the field of aerospace concepts design, multi-container (*i.e.*, multi-module) OLPs have also been decomposed in two stages in order to solve the assignment of components and the layout of each module sequentially. For instance, in [CZa19; Tia+23; Zha+22a; ZXT19; XZT19] multi-container satellite module OLPs are addressed thanks to two-stage approaches. Each component is first allocated to one module in an upper stage and the layout of the modules are then optimized subsequently in a lower stage.

Figure 3.15 sums up the aforementioned categories of multi-container optimal layout problems.

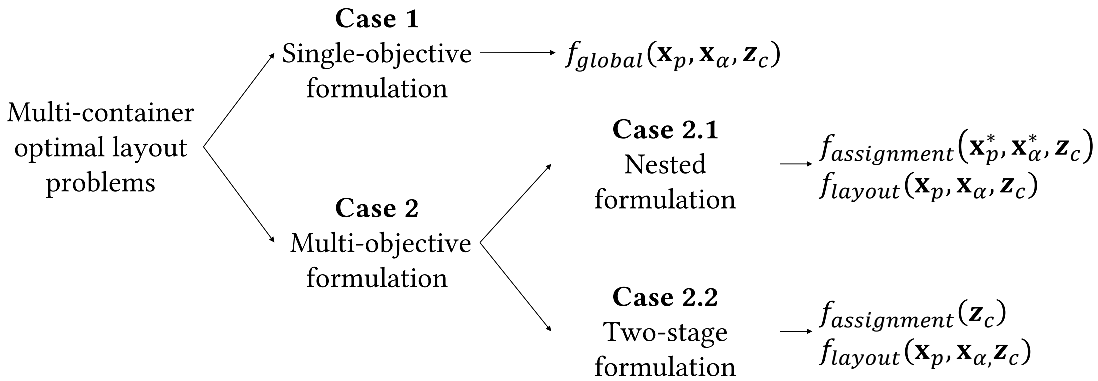


Figure 3.15: Formulations of multi-container OLPs.

In the following, the considered multi-container OLPs are similar to the two-stage class of problems for instance considered in [APJ17; AJ16; CZa19; KSN22; Zha+22a; ZXT19] *i.e.* the case 2.2 formulated with Equations 3.31. The following section aims at developing a two-stage algorithm framework for such category of problems. Subsequently, in Chapter 4, a multi-container satellite module OLP will be described and the global inertia along with the 3-dimensional axes is used as objective function to minimize. The analysis of the inertia equations allows to formulate the problem as a two-stage problem.

3.3.2 Two-stage algorithm based on Genetic Algorithm and CSO-VF for multi-container optimal layout problems

To address the formulation detailed in Equations 3.31, a two-stage algorithm is proposed. Each of the upper and lower stage aims at solving respectively the assignment task and the layout task. Each stage is described as follows.

3.3.2.1 Upper stage

The upper stage solves the assignment task. It takes as input the list of geometrical features of the components and the containers (dimensions, masses, exclusion zones). The output of the upper stage consists in an archive of N_{AS} assignment schemes corresponding to N_{AS} lists of assigned containers to each component. Thus, the design variables to optimize are: $\mathbf{z}_c = \{z_{c,i}\}_{i \in \{1,...,N\}}$ where $z_{c,i} \in \{1, ..., n\}$. The problem to solve is a constrained categorical combinatorial problem. The size of the design space is n^N. Metaheuristics have been widely employed to solve this kind of problems [BAS12]. Among them, Genetic Algorithms are chosen because they benefit from good global exploration ability when it comes to multi-modal objective functions and can deal with a high number of variables unlike other methods like Bayesian Optimization. Moreover, they have been widely employed to solve highly combinatorial problems and can inherently deal with categorical variables in comparison to other metaheuristic techniques like PSO. Finally, they do not require the definition of a neighborhood structure in the design space unlike SA, TS or VNS for instance, which makes arising additional challenges in a categorical search space.

3.3.2.2 Lower stage

The lower stage solves the layout task. This stage takes as input the assignment list optimized by the upper stage. The outputs are the lists of positions and orientations of each component in its assigned container. It is considered in this thesis that the layout of each container can be solved independently. In other words, the constraints functions are defined for each container, no constraint function requires a simultaneous resolution of the containers layouts. The CSO-VF algorithm described in the first section of this chapter is chosen and is run for each container.

3.3.2.3 Algorithm framework

The algorithm proceeds as follows: the upper stage optimizes the assignment list using a GA. The output corresponds to an archive with N_{AS} assignment schemes corresponding to the N_{AS} final best solutions, sorted based on their objective function values. Subsequently, the CSO-VF algorithm is called to optimize the layout of each of the containers for the first assignment scheme of the archive. If a final feasible layout is found, the output corresponds to the best obtained layout. If the lower level does not lead to a feasible solution, the CSO-VF algorithm is called to optimize the layout of the containers with the second assignment scheme of the archive, and so on. The termination criterion is defined as a maximum number of sequential resolutions *i.e.*, the size N_{AS} of the archive given by the upper level. The algorithm framework is illustrated on Figure 3.16.

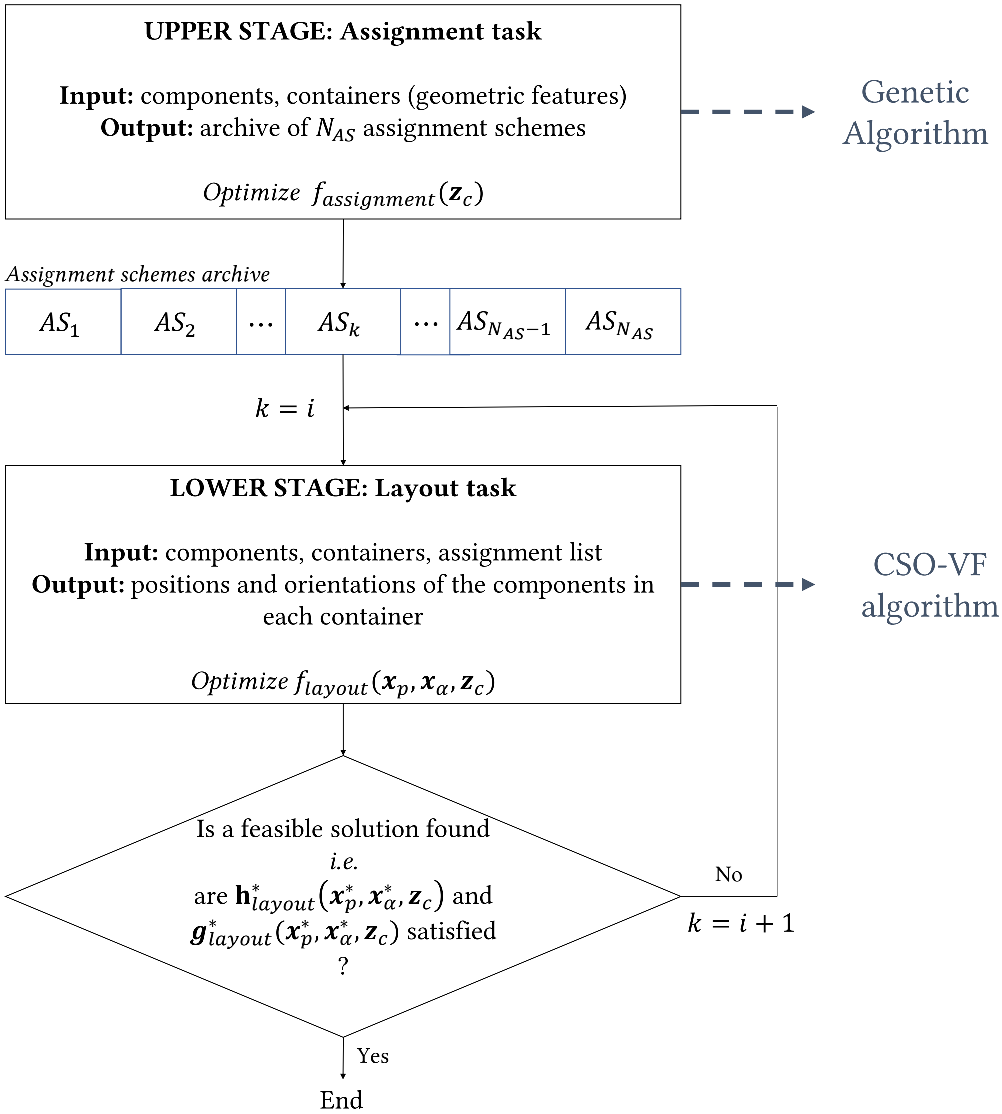


Figure 3.16: Two-stage algorithm framework for solving multi-container OLPs.

3.4 Conclusion

In this chapter, generic single and multi-container OLPs have been formulated and two algorithms have been devised to handle them.

It has been highlighted that existing VFSs might be limited for solving single-container OLPs which involve polygonal components, various objective and constraint functions differentiable with respect to the design variables, as well as balancing constraint along with exclusion zones in the container. Thus, a new quasi-physical approach based on a VFS, named CSO-VF algorithm, has been developed for solving generic OLPs corresponding to extended packing problems which have been previously tackled in the literature by similar methods. The new VFS encompasses gradient-based forces designed to deal with all possible differentiable objective and constraint functions, as well as torques allowing to update the orientations of non circular components based on the fundamental principle of the dynamics of rotation. Subsequently, a swap operator has been introduced which allows to exchange the positions of components with the aim of either improving the objective function or to decrease some constraints violations. The swap operator call frequency depends on the convergence state. As the CSO-VF algorithm is a local search algorithm and locally improves the non convex objective function, the initialization of the layout may have a critical impact on the final obtained layout and the convergence speed. Thus, LHS initialization and multistart are employed.

Thereafter, generic multi-container OLPs have been described and formulated. As they involve an assignation task responsible for the distribution of the components in the appropriate containers, existing quasi-physical approaches as well as the proposed CSO-VF algorithm can not be used directly. Moreover, solving both assignment and layout tasks in the same optimization process may suffer from poor convergence capabilities due to the high number of additional design variables and constraints related to the assignation tasks which induce a very highly combinatory design space. Thus, a two-stage algorithm framework was proposed in order to solve both assignment and layout tasks sequentially, in the case where the design variables related to the assignment and to the layout are separable. The assignment task is solved thanks to a Genetic Algorithm in an upper stage. Subsequently, the layouts of each container are optimized thanks to the CSO-VF algorithm as they correspond to single-container OLPs.

In the next chapter, both algorithms are applied to single and multi-container satellite module layout problems. Their configuration and setups are discussed and their global performance are analyzed in comparison to other existing counterparts.

Chapter 4

Applications to Satellite Module Optimal Layout Problems

Contents

Chapter contributions

- Formulation of the single-container and multi-container satellite module layout problems.
- Description and analysis of the CSO-VF algorithm settings, and comparison of its global performance to GAs and CMA-ES for single-container satellite module layout problems.
- Application of the GA assisted-CSO-VF algorithm, analysis of its performance and comparison of the obtained results with previous published results for multi-container satellite module layout problems.

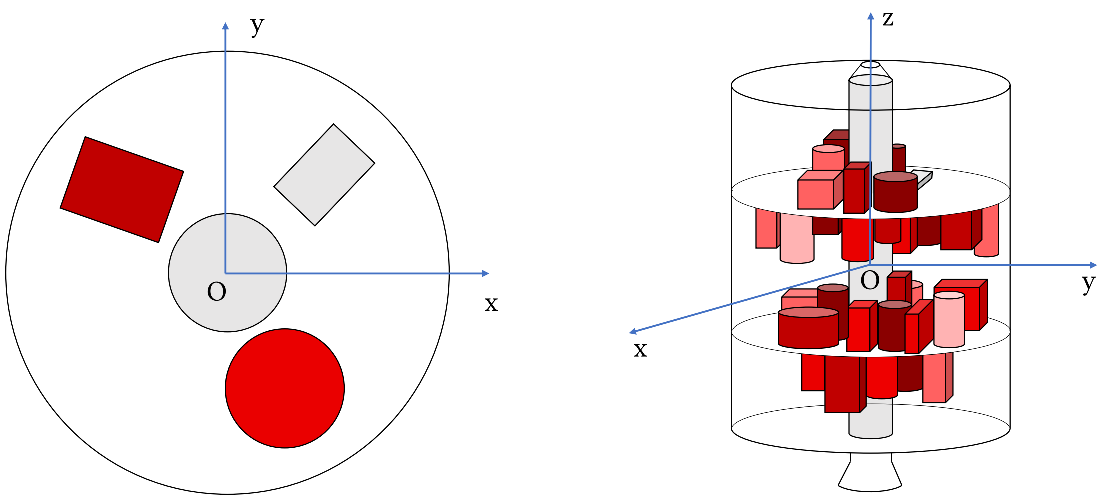
4.1 Introduction

The optimal layout of the simplified model of the international commercial communication satellite module (INTELSAT-III) is considered in this thesis as a representative application case of optimal layout problems (OLPs). Indeed, it allows to take into account the specifities of OLPs introduced in Chapter 3 like for instance the exclusion zones in the containers, the highly dimensional or highly constrained aspects (*i.e.*, overlapping, balancing, functional constraints). Those problems are referred to as Satellite Module Layout Problems (SMLP) and are widely employed as a benchmark for optimal layout methods [CXW18; CZa19; Ten+09; ZTS08]. Mosltly metaheuristic methods have been employed in order to solve the SMLP. Among them, methods based on Simulated Annealing [FT16; LLC10], Particle Swarm Optimization [ZTS08], Genetic Algorithm [Liu12; ZTS08], Ant Colony Optimization [ST03; Xu+10], *etc.* More recently, with the increase in the number of variables and constraints, cooperative coevolutionary evolutionary strategies have been employed [CXW18; CZa19; HT09; Ten+09; ZWT13]. Other techniques have also been used such as neural networks [ZTS08], Wang-Lau strategy [Liu+16b] or human-computer cooperation approaches [LT08; ZWT13].

In this chapter, two configurations of the INTELSAT-III satellite module are considered:

- Single-container configuration: the container to be laid out corresponds to a one-sided plate. Two exclusion zones are considered: a central bus and a rectangular bus as illustrated in Figure 4.1a;
- Multi-container configuration: the container to be laid out corresponds to the top and bottom surfaces of two bearing plates as illustrated in Figure 4.1b. A central bus corresponds to exclusion zones on each surface.

Figure 4.1 shows the single and multi-container configurations of the INTELSAT-III satellite module. In these benchmark configurations, the containers are supposed circular and the components are cuboids or cylinders, but this is not mandatory to assess the algorithms developed in Chapter 3.



(a) Single-container configuration (b) Multi-container view

Figure 4.1: Illustrations of the single and multi-container configurations of the simplified model of the INTELSAT-III satellite module. Light grey areas correspond to fixed components (*i.e.*, exclusion zones). The redder the components, the heavier their corresponding masses.

The algorithms detailed in Chapter 3 and dedicated to both single and multi-container optimal layout problems are applied in this chapter to the single and multi-container SMLP in order to assess their performance.

The rest of the chapter is organized as follows: Section 4.2 is devoted to the single-container SMLPs. In this section, the problem at hand is described and mathematically formulated. The CSO-VF algorithm configuration is subsequently detailed and examined. Finally, the global performance of the algorithm is analyzed and compared to the one obtained with GAs and CMA-ES. Section 4.3 is devoted to multi-container SMLPs and follows the same organization than Section 4.2. The problem at hand is formulated and the GA assisted-CSO-VF algorithm is configured. The obtained results are compared with those published in the literature.

4.2 Single-container satellite module optimal layout problems

4.2.1 Problem formulation

The fixed search space SMLPs involves positioning N components within a container in order to minimize the global inertia of the module. Several constraints must also be satisfied such as geometrical ones(*e.g.*, overlapping constraints, balancing constraints) or functional ones (*e.g.*, constraints relative to functional compatibility between components due to electromagnetic or heat radiations threshold within the container).

4.2.1.1 Definition of the container and components

In the single-container configuration, the container is a one-sided bearing plate defined by its outer radius R_{out}. Two exclusion zones are defined as follows:

- A central circular exclusion zone defined by its radius R_{in} and centered at the geometrical center of the plate;
- A rectangular exclusion zone defined by its dimensions (L_{bus}, l_{bus}) and its positions on the plate in the cylindrical system of coordinates (r_{bus}, θ_{bus}).

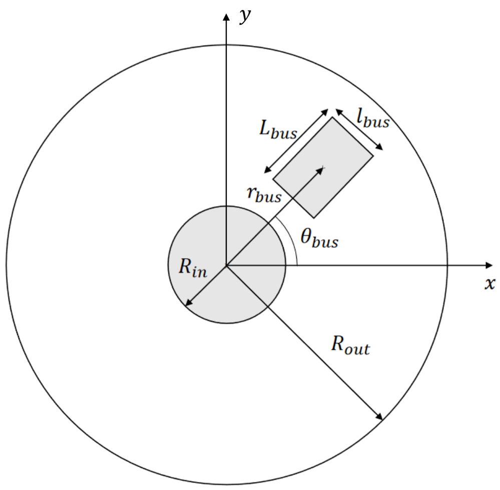
Figure 4.2a illustrates the geometrical definition of the module.

The components, N_{cyl} cylinders and N_{cub} cuboids, are defined by their dimensions and masses. They are considered as rigid bodies of homogeneous density and are located thanks to the position of their centers of inertia. They are of three types:

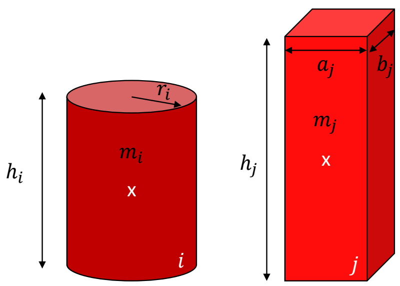
- Fuel components;
- Energy components;
- Other components.

Cylinders are defined using their radius r_i, height h_i and mass m_i. Cuboids components are defined using their base dimensions a_i, b_i, their height and their mass.

Figure 4.2b illustrates the components. All dimensions as well as the list of the components considered for the fixed search space single-container SMLP configuration are reported in Table A.1 in Appendix A.



(a) Geometrical definition of the container.



(b) Geometrical definition of the components.

Figure 4.2: Geometrical definition of the container and the components for the FSS single-container SMLP configuration.

4.2.1.2 Design variables

In the single-container configurations, the design variables are:

- The positions of the centers of inertia of the components considered as continuous variables $(x_{x,i}, x_{y,i})$, $i \in \{1, ..., N\}$.
- The orientations of all cuboids components considered as continuous variables, $x_{\alpha,i}$, $i \in \{1, ..., N_{cub}\}$.

Thus, the number of design variables n_{var} is given by:

$$n_{var} = 2N_{cyl} + 3N_{cub} \tag{4.1}$$

4.2.1.3 Objective function

One of the most popular objective functions to minimize in OLPs is the inertia of the whole module [CZa19; Ten+09; ZTS08]. In this section, the inertia is calculated for the single-container configuration. Two systems of coordinates are adopted in order to calculate the global inertia of the module at the geometrical center of the plate:

- $O"x"y"z"$: the local coordinate system attached to each component. $O"$ corresponds to the center of inertia of the component and the axes are defined by the symmetry axes of the components;

- $Oxyz$: the global coordinates system attached to the container. O is the geometric center of the container and the axes are its symmetry axes.

The two coordinate systems are illustrated in Figure 4.3.

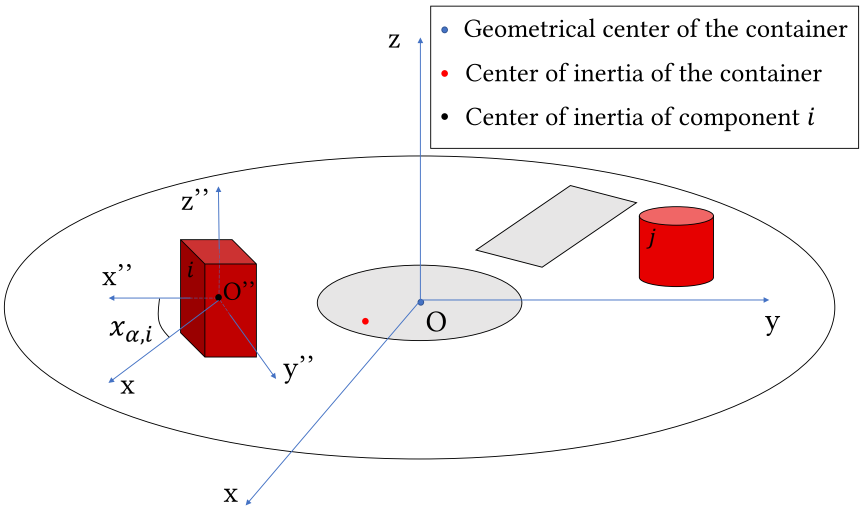


Figure 4.3: Sketch of the coordinates systems.

The total inertia is calculated in the $Oxyz$ system of coordinates and is defined as:

$$I_{tot}(\mathbf{x}) = I_x(\mathbf{x}) + I_y(\mathbf{x}) + I_z(\mathbf{x}) \tag{4.2}$$

where $\mathbf{x}$ corresponds to the continuous design variables. The $\mathbf{x}$ vector encompasses the positions of the centers of inertia of all the components as well as their orientations.

Each component has a solid inertia which depends on its shape and is expressed in its $O"x"y"z"$ coordinates system. The inertia along the three axes are denoted $I_{x",i}$, $I_{y",i}$ and $I_{z",i}$ and are derived for each possible geometry of the components (*i.e.*, cylinders or cuboids) in Appendix B. Thus, the inertia of the component in the system of coordinates $Oxyz$ is:

$$I_{x,i} = I_{x",i}\cos(x_{\alpha,i})^2 + I_{y",i}\sin(x_{\alpha,i})^2 + m_i(x_{y,i}^2 + x_{z,i}^2) \tag{4.3}$$
$$I_{y,i} = I_{y",i}\cos(x_{\alpha,i})^2 + I_{x",i}\sin(x_{\alpha,i})^2 + m_i(x_{x,i}^2 + x_{z,i}^2) \tag{4.4}$$
$$I_{z,i} = I_{z",i} + m_i(x_{x,i}^2 + x_{y,i}^2) \tag{4.5}$$

where m_i is the mass of the component i, $x_{x,i}, x_{y,i}, x_{z,i}$ are the coordinates of its center of inertia in the system of coordinates $Oxyz$ and $x_{\alpha,i}$ its orientation. N components have to be laid out in the container. Consequently, the inertias of the module along each axis calculated at the point O are expressed as follows:

$$I_x = \sum_{i=1}^{N+N_{EZ}} I_{x,i} \tag{4.6}$$
$$I_y = \sum_{i=1}^{N+N_{EZ}} I_{y,i} \tag{4.7}$$

$$I_z = \sum_{i=1}^{N+N_{EZ}} I_{z,i} \tag{4.8}$$

where N_{EZ} is the number of exclusion zones that are fixed components and thus have a mass and a solid inertia.

4.2.1.4 Constraint functions

The constraints considered in this thesis fall into 2 categories: geometrical constraints and functional constraints. Geometrical constraints that are considered are listed below:

- **Overlapping constraints between the components:** no overlapping between the components is allowed. Figure 4.4 shows two different layouts. On the leftmost layout, the overlapping constraint between two components is violated as they are clearly overlapping. The rightmost layout depicts a case in which the overlapping constraint is, on the contrary, fully satisfied. With the same formalism as in Section 4.2.1.3, the overlapping constraint is expressed as:

$$h^C_{overlap}(\mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \Delta A^C_{ij}(\mathbf{x}) \tag{4.9}$$

  where ΔA^C_{ij} is the area of intersection between the 2-dimensional projections of components i and j, and is function of the positions of the centers of inertia of the components and their orientations.

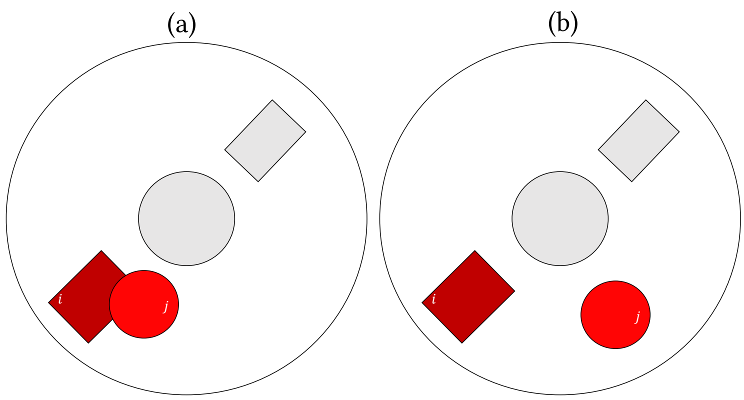


Figure 4.4: Overlapping constraint between the components (a) violated (b) satisfied.

- **Overlapping constraints between the components and the exclusion zones:** no overlapping between the components and the exclusion zones is allowed. Figure 4.5 shows two layouts where the overlapping constraint is respectively violated and solved. With the same formalism introduced in Section

4.2.1.3, this overlapping constraint is expressed as:

$$h^{E}_{overlap}(\mathbf{x}) = \sum_{i=1}^{N} \Delta A^{E}_{i}(\mathbf{x}) \tag{4.10}$$

where ΔA_i^E is the area of intersection between the 2-dimensional projections of a component and the exclusion zones, and is a function of the positions of the centers of inertia and orientation of each component (as well as the position and shape of each exclusion zones).

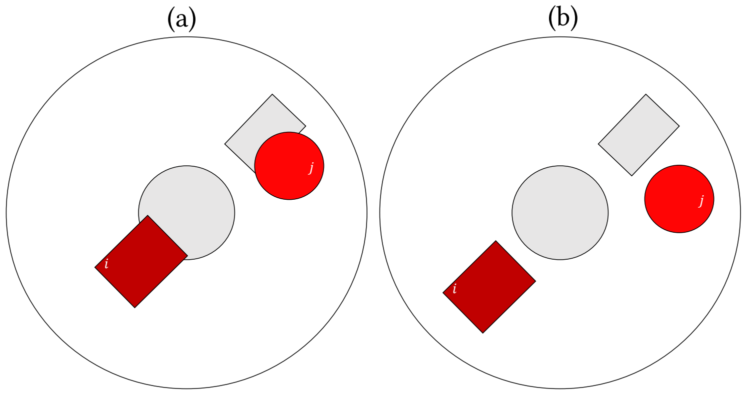


Figure 4.5: Overlapping constraint between the components and the exclusion zones (a) violated (b) satisfied.

- **Balancing constraints:** the center of gravity (CG) of the whole container must be positioned within a tolerance zone centered at the geometric center of the container. Figure 4.6 illustrates two layouts where the balancing constraint is respectively violated because the CG is not accurately positioned, and solved because the CG is in the tolerance zone. Using the same formalism as before, this constraint can be expressed as:

$$g_{CG}(\mathbf{x}) = \sqrt{(x_{x,c} - x_{x,e})^2 + (x_{y,c} - x_{y,e})^2} - \delta \tag{4.11}$$

where $(x_{x,c}, x_{y,c})$ are the coordinates of the current CG of the whole module and $(x_{x,e}, x_{y,e})$ are the expected coordinates of the CG. It is considered to be the geometric center of the container $(0, 0)$ in this thesis. δ represents a small tolerance which corresponds, in this thesis, to a circle centered about the origin of the $Oxyz$ coordinates system and whose radius is set to 1% of the outer radius of the container, without loss of generality.

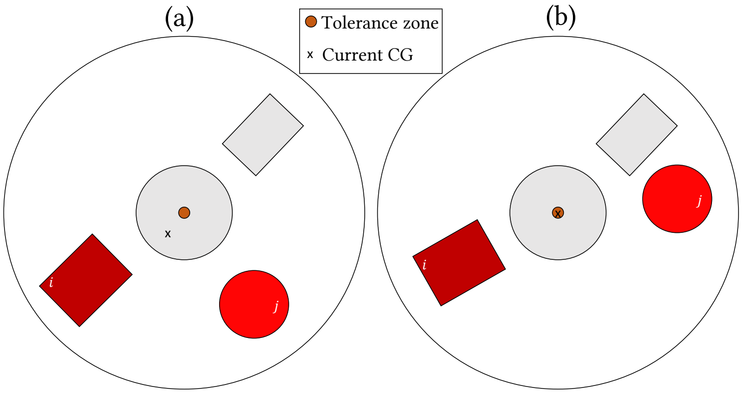


Figure 4.6: Balancing constraint (a) violated (b) satisfied.

A series of **functional constraints** can also be added to the problem formulation via the use of special components that introduce further restrictions on the feasibility of the layout. The existence of specific components may involve some restrictions in the layout. For example, in this thesis, fuel components and energy components must be kept at a certain distance from each other for functional reasons such as not exceeding a radiation (*e.g.*, heat, electromagnetic) threshold at any point of the module. Figure 4.7 illustrates the functional constraint. Each functional component has thus a forbidden zone that must not be violated by incompatible components. This zone is represented by the dotted lines. Figure 4.7 shows two layouts where the functional constraint is respectively violated because the energy and fuel components are too close to each other and solved because the minimal distance between them is respected. With the same formalism as before, considering a layout with N_E energy components and N_F fuel components, the functional constraint is expressed as:

$$h_{functional}(\mathbf{x}) = \sum_{i=1}^{N_E} \sum_{j=1}^{N_F} \Delta A_{ij}^{EF}(\mathbf{x}) + \Delta A_{ji}^{EF}(\mathbf{x}) \tag{4.12}$$

where ΔA_{ij}^{EF} corresponds to the area of intersection between the 2-dimensional projections of the forbidden zone of component i and the component j, and ΔA_{ij}^{EF} corresponds to the area of intersection between the 2-dimensional projections of the forbidden zone of component j and the component i.

4.2.1.5 Occupation rate

The occupation rate OR of the container is defined as:

$$OR = \frac{\sum_{i=1}^{N} A_i}{A_{container}} \tag{4.13}$$

where A_i is the area of each component i and $A_{container}$ is the area of the container. The performance of the proposed CSO-VF algorithm will be studied for four increasing values of the occupation rate: 30%, 40%, 50% and 60%. The occupation rate is a critical specificity of the problem. Indeed, the higher the occupation rate, the more difficult it is to satisfy the constraints.

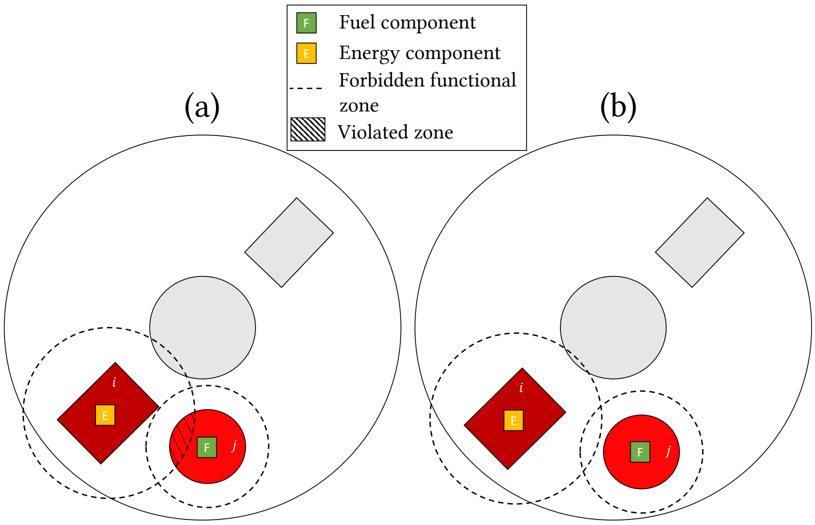


Figure 4.7: Functional constraint (a) violated (b) satisfied. Dotted lines correspond to the forbidden zones of energy or fuel components.

4.2.1.6 Mathematical formulation

The fixed search space single-container SMLPs described in the previous sections are mathematically defined as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & I_{tot}(\mathbf{x}) \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_{var}} \\ \text{s.t.} \quad & \mathbf{h}^C_{overlap}(\mathbf{x}) = 0 \\ & \mathbf{h}^E_{overlap}(\mathbf{x}) = 0 \\ & \mathbf{h}_{functional}(\mathbf{x}) = 0 \\ & g_{CG}(\mathbf{x}) \leq 0 \end{aligned} \tag{4.14}$$

4.2.2 Parameter settings of CSO-VF

4.2.2.1 Virtual-force system

With respect to the previous problem formulation, the virtual-force system is defined with the following forces and torques:

- Force $\mathbf{F}^{overlap}_{ij}$ and torque $\mathbf{T}^{overlap}_{ij}$ are used to solve for the overlapping constraints between components i and j;
- Force $\mathbf{F}^{overlap}_{i/EZ}$ and torque $\mathbf{T}^{overlap}_{i/EZ}$ are used to solve for the overlapping constraints between components i and an exclusion zone;
- Force $\mathbf{F}^{container}_{i}$ and torque $\mathbf{T}^{container}_{i}$ are used to solve for the overlapping constraints between components i and the container;
- Force $\mathbf{F}^{functional}_{i}$ and torque $\mathbf{T}^{functional}_{i}$ are used to solve for the functional constraints between two components i and j;

- The force $\mathbf{F}_i^{CG}$ is used to solve for the balancing constraint;
- Force $\mathbf{F}_i^{inertia}$ and torque $\mathbf{T}_i^{inertia}$ are used to minimize the inertia of the system.

4.2.2.2 Swap operator

The swap operator defined in Chapter 3 is used in order to improve both the balancing constraints and the objective function. In other words, two components are allowed to exchange their positions if the swap improves either the position of the center of gravity or the global inertia, in a limit of additional overlapping and functional constraints violation defined by a hyperparameter denoted r.

Figure 4.8 illustrates the effect of the swap operator on the objective function for a 30% occupation rate configuration. The figure shows three consecutive feasible configurations with decreasing objective function values and with two successive calls of the swap operator. The first swaps contribute to separate the incompatible fuel and energy components located at the bottom of the container which allows to bring closer the energy components. The second swaps notably enable the two heavy fuel components to be relocated closer to the center of the container which contributes to improve the global inertia.

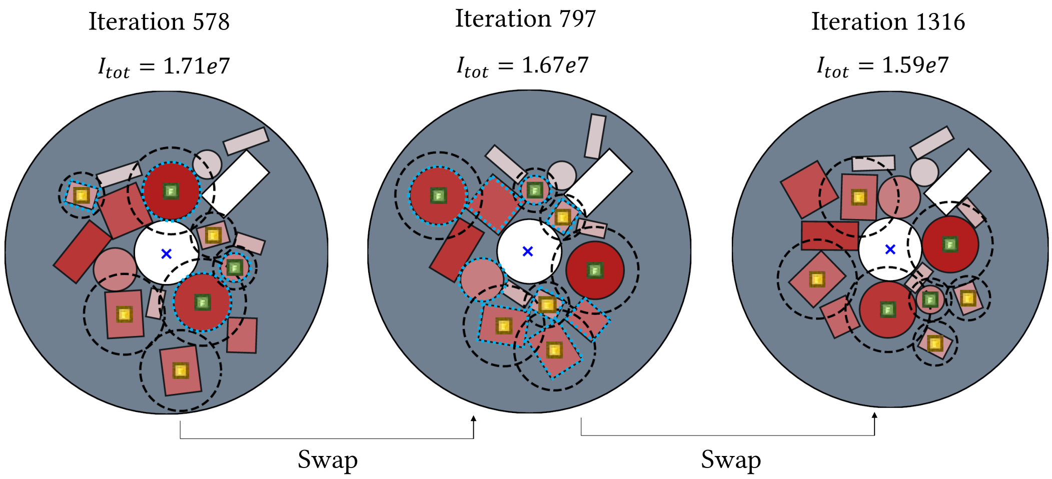


Figure 4.8: Illustration of the swap operator on an example for a 30% occupation rate. The swapped components are highlighted in blue.

4.2.2.3 Initialization

As described in Chapter 3, an optimized Latin Hypercube Sampling strategy [DCI13; Li+17; PM12] is used to initialize both the positions and orientations of the components. Moreover, as the objective function corresponds to the global inertia of the module, an additional heuristic rule is introduced: the components are distributed on the generated positions following the principle that the heaviest components occupy the closest generated positions to the center of the plate (and thus the lightest, the farthest). Figure 4.9 illustrates the evolution between an initialized 40% occupation rate configuration and the corresponding optimal layout obtained using the CSO-VF algorithm.

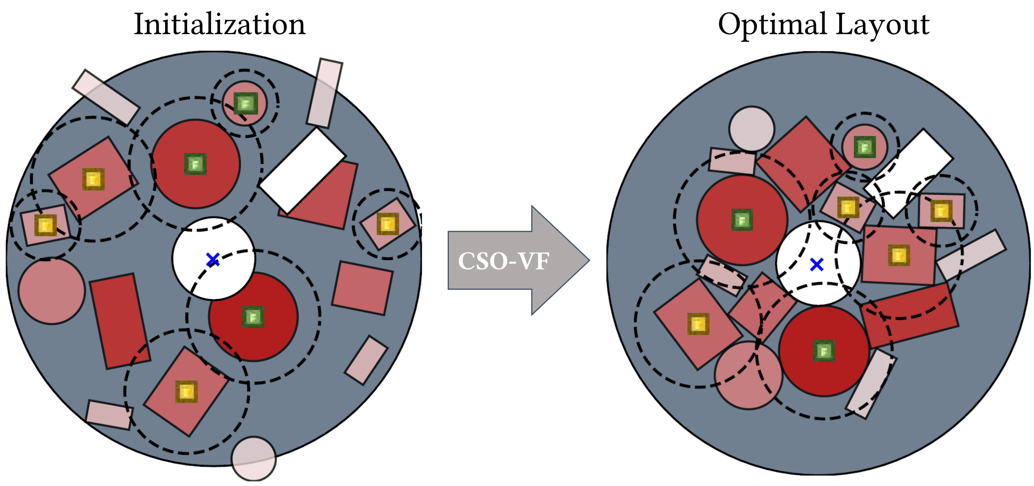


Figure 4.9: One initialization and the corresponding optimal layout for a 40% occupation rate configuration.

4.2.2.4 Hyperparameters

The 10 hyperparameters of the described CSO-VF algorithm are:

- The four dynamical hyperparameters: maximum force F_{max}, torque T_{max}, translation speed v_{max}, rotational speed ω_{max};
- The gradient-based forces and torques stepsizes: α^F_{CG} for the balancing force, $\alpha^T_{inertia}$ and $\alpha^F_{inertia}$ for the inertia force and torque;
- The swap operator parameters: the minimum and maximum steps s_{min}, s_{max} and the relaxation factor r.

4.2.3 Analysis of the settings of the CSO-VF algorithm

In this section, the single-container SMLP is used in order to analyze the CSO-VF algorithm settings: the hyperparameters settings, the choice in the search space of the orientation variables, the initialization and swap operators, as well as the computational cost.

4.2.3.1 Analysis of the hyperparameters settings

In this section, a sensitivity analysis is performed over the hyperparameters listed in Section 4.2.2.4 and for an intermediate occupation rate of 40% of the container.

The principle of estimating sensitivity indices using Sobol's method is to simulate two samples from the input quantities (*i.e.*, estimate the objective function of the layout configuration obtained thanks to hyperparameters samples) and then to estimate the conditional variance with respect to each input quantity by combining these two samples. The advantage of sensitivity analysis with Sobol's method is that it doesn't make any assumptions about the properties of the studied function (*i.e.*, the objective function corresponding to a layout configuration obtained with the sampled hyperparameters), and is suitable whatever the model.

It must be noted that the Sobol's method requires a sufficiently large sample size to ensure convergence of the indices according to the complexity of the measurement model. For this reason, the model inputs are generated using Saltelli's extension of the Sobol' sequence. The Sobol' sequence is a widely employed quasi-random low-discrepancy sequence used to generate uniform samples of input space [Sob01]. Saltelli's scheme extends the Sobol' sequence in a way to reduce the error rates in the resulting sensitivity index calculations [Sal02; Sal+10].

Figure 4.10 shows the first-order and total-order Sobol's indices for each hyperparameter [Sob01] over the final objective function. The hyperparameters samples are generated using the Saltelli sampler of the SALib python library [HU17], resulting in 6144 samples. The bounds defined for each hyperparameter have been set using a parametric study and are reported in Appendix C.

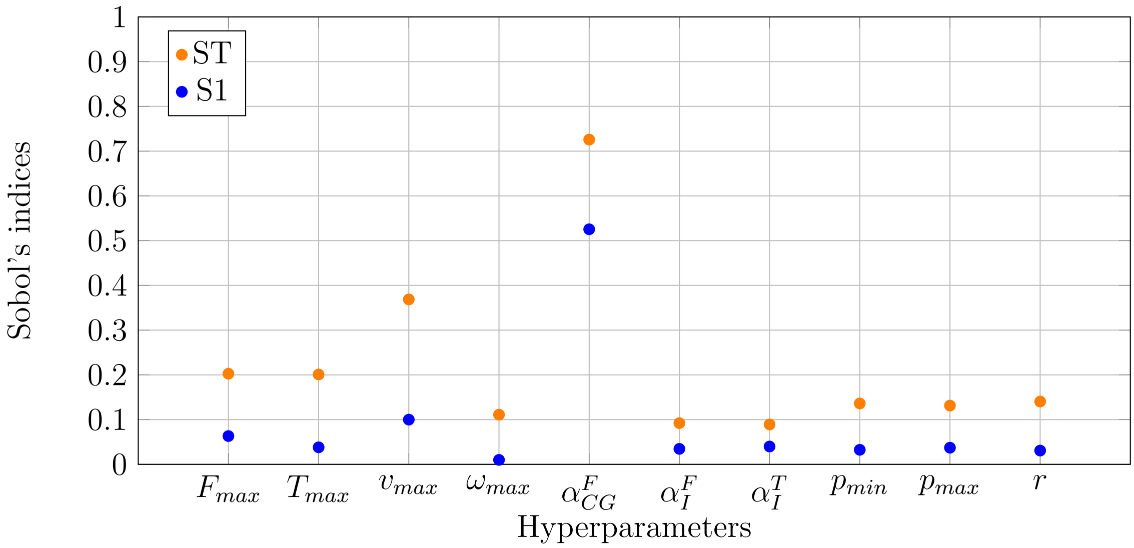


Figure 4.10: First-order (S1) and total-order (ST) Sobol's indices for the hyperparameters.

The α_{CG}^F parameter has the most influence on the final layout. Indeed, if this parameter is too small, the resulting balancing force will be smaller than the other constraint forces and the center of gravity may never reach the tolerance zone. Consequently, no feasible solution will be found for inadequate settings of this parameter. The maximum translation speed also has a strong impact on the results. Indeed, if v_{max} is set to a too small value, the components will not move fast enough within the container. Consequently, either no feasible layout is found during the maximum number of iterations or a suboptimal layout may be reached. On the contrary, if v_{max} is set to a too large value, the components move too fast to solve the constraints and no stabilized feasible solution will be found. It can be observed that the other parameters have a smaller yet similar effect on the final results.

In the following, the hyperparameters are optimized for each occupation rate of the container using Bayesian Optimization [Sha+15]. Indeed, Bayesian Optimization has been widely employed to tune hyperparameters of algorithms [AL21; VM21; Wu+19] and provides fast convergence when it comes to expensive-to-evaluate black-box objective functions.

To do so, a bilevel algorithm is defined combining Bayesian Optimization with Gaussian Process surrogate modeling at the upper level and the CSO-VF algorithm at the lower level. The design variables of the upper level correspond to the hyperparameters of the lower level and the objective function is the output of the lower level *i.e.*, the final objective function. The algorithm process is as follows: the upper level output is a list of hyperparameters that is given to the lower level as an input. The lower level solves the optimal layout problem with the CSO-VF algorithm employed with multi-start (with a number of starts $N_{start} = 10$). The best objective function obtained corresponds to the output of the lower level which is returned to the Bayesian Optimization level in order to refine the Gaussian Process surrogate modeling and propose the new most promising list of hyperparameters in terms of final objective function of the layout configuration. The hyperparameter list is refined sequentially with a given budget of an initial training data set of $N_{DoE} = 50$ samples and $N_{it} = 100$ iterations. The bounds of each hyperparameter value were defined using a parametric analysis.

4.2.3.2 Note on the geometry of the components

Even if the aforementioned application case is restricted to cylinders and cuboids components, the CSO-VF algorithm can handle various shapes of components as shown on Figure 4.11.

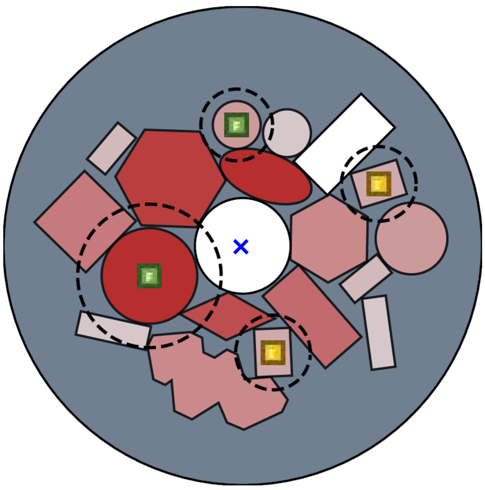


Figure 4.11: Example of feasible layout with various shapes of components.

4.2.3.3 Analysis of the choice of the orientation parametrization

In most of the studies addressing the SMLP problem, the orientations of the components are set as discrete variables [CXW18; CZa19; Ten+09; ZTS08]. This section aims at comparing the choice between discrete and continuous orientations in terms of final obtained layouts. The CSO-VF algorithm is performed over 50 random initializations and over 5000 iterations for each orientation configuration and four occupation rates: 30%, 40%, 50% and 60%. Table 4.1 sums up the obtained numerical results in terms of:

- The number of successful runs *i.e.*, whether or not a feasible solution is found at the end of the optimization process;
- The median of the final objective function value;
- The interquartile range (IQR) of the final objective function value;
- The best run in terms of final objective function;
- The mean of the iteration providing the first feasible solution *i.e.*, the iteration from which all the constraints have been simultaneously solved.

Figure 4.12 shows the best final layout for each discrete and continuous orientation configuration and the four occupation rates.

	OR	Dicrete	Continuous
Nb of successful runs (a feasible solution is found)	30%	**50**	**50**
	40%	**50**	**50**
	50%	39	**50**
	60%	4	**31**
Final objective function (median)	30%	**1.635e7**	1.641e7
	40%	2.238e7	**2.192e7**
	50%	2.759e7	**2.731e7**
	60%	-	**3.311e7**
Final interquartile range (IQR)	30%	1.142e6	**1.040e6**
	40%	1.487e6	**1.261e6**
	50%	1.975e6	**1.741e6**
	60%	-	**2.421e6**
Best layout (objective function)	30%	**1.414e7**	1.432e7
	40%	1.990e7	**1.971e7**
	50%	2.500e7	**2.444e7**
	60%	**2.937e7**	3.019e7
Generation of first feasible solution (mean)	30%	**191.14**	199.28
	40%	420.82	**416.2**
	50%	1413.15	**1361.56**
	60%	-	**2512.56**

Table 4.1: Numerical results obtained for the four configurations of initialization and swap operator and for the three occupation rates.

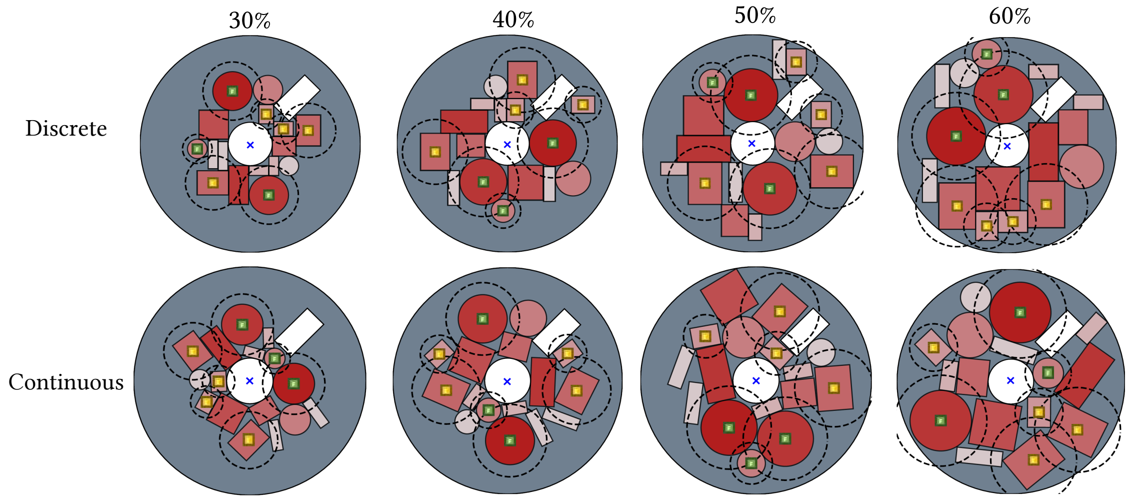


Figure 4.12: Best layout obtained for both fixed and free orientations and the 4 studied occupation rates.

First of all, it can readily be shown that when the occupation rate is increased, there are less and less successful runs. That is even more so for the discrete orientation whose performance starts deteriorating from an occupation rate of 50%. Indeed, the constraints become increasingly difficult to solve as observed in Figure 4.12. However, considering the highest occupation rate *i.e.*, 60%, the discrete configuration enables to provide 4 successful runs over the 50 instances against 31 for the continuous configuration. As there are only 4 final results for the discrete configuration, the statistical metrics *i.e.*, the median of the final result, the IQR of the final results and the mean of the iterations providing the first feasible solution are not computed in this case. It must be due to the fact that continuous orientations provide a wider search space which allows to solve more easily the constraints than discrete orientations that often lead to blocked configurations in case of high occupation rates.

Regarding the other metrics, both configurations seem to provide similar performance. The continuous configuration provides very slightly better median and IQR values over the final successful for occupation rates greater or equal than 40%. Moreover, no trend is observed in terms of the best obtained layout. Finally, the first feasible solutions are reached at similar iterations for both configurations with a slight advantage for the continuous configuration from the 40% occupation rate.

Then, due to the ability of the continuous configuration to find feasible solutions even for high occupation rates, this configuration is used in the following studies. Furthermore, to the author's best knowledge, this configuration has rarely been studied when it comes to SMLP benchmarks.

4.2.3.4 Analysis of the swap and initialization operators

In this section, the influence of the initialization and of the swap operators on the constraint resolution and final layouts are studied. For this purpose, two configurations are compared for the four occupation rates:

- Random initialization and no swap operator (R);
- The LHS and heuristic rule initialization and the swap operator (LS).

Figure 4.13 illustrates initialized layouts obtained randomly or with the LHS and Heuristic rule (LHS+Heuristic) technique for the four occupation rates.

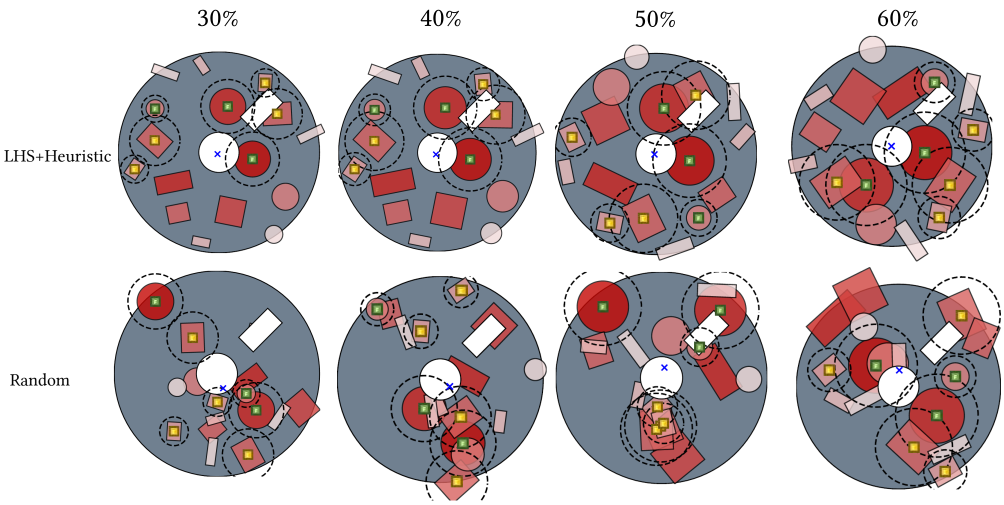


Figure 4.13: Initial layouts obtained randomly or with the LHS+Heuristic technique for four occupation rates.

As predicted, the LHS+Heuristic initialization allows to distribute the components in a more homogeneous way than the random initialization. Consequently, it is observed that the initial total constraint violation of the initial layouts is smaller for the LHS+Heuristic initialization than the random one.

50 randomly initialized instances of the CSO-VF algorithm are run for each configuration, for the four occupation rates and during 5000 iterations.

Constraint resolution Figure 4.14 reports the mean of the first iteration from which each of the following constraints are either independently solved or simultaneously solved:

- **Overlap:** It encompasses the 3 overlapping constraints, *i.e.* the overlapping between the components, between the components and the exclusion zones and between the components and the container;
- **CG:** The balancing constraint;
- **Functional:** The functional constraint;
- **All:** All the previous constraints are solved, *i.e.* a feasible solution is found.

As shown in the previous section, the more the occupation rate increases, the more iterations it takes to solve the constraints and reach a feasible solution. The LHS initialization in addition to the swap operator helps to provide first feasible solutions faster than the random and no swap configuration, and this for all the

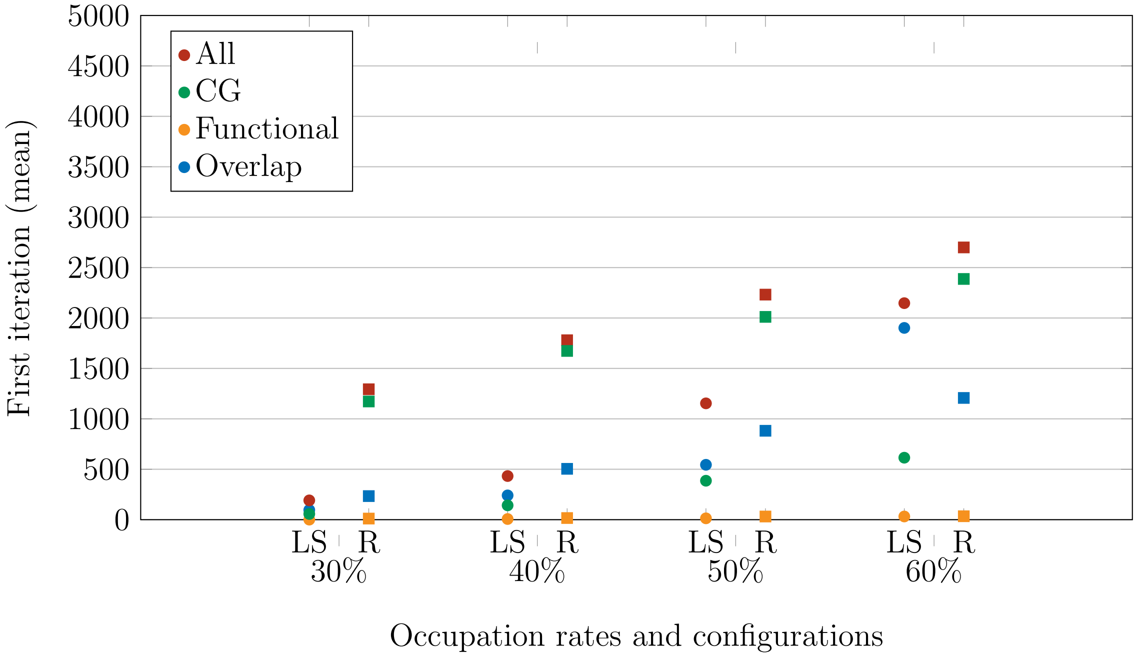


Figure 4.14: Constraints resolution.

occupation rates. Indeed, the relative improvements related to the first iteration providing a feasible solution are respectively 85.2%, 75.7%, 48.3% and 20.4% from the 30% to the 60% occupations rates configurations. Moreover, Figure 4.14 shows that for the CSO-VF algorithm enhanced with the LHS initialization and the swap operator, the functional constraint is the first constraint to be satisfied, followed by the balancing constraints and finally the overlapping constraints. For the random initialized and no swap CSO-VF algorithm, the balancing constraint is the last constraint to be satisfied. This is due to the fact that the LHS initialization distributes the components in more homogeneous way than the random initialization, which favors the placement of the global center of gravity. Moreover, the first calls of the swap operator also helps to satisfy this constraint. When it occurs, the swap operator might add some overlapping constraints violations which is consequently the last constraint to be solved.

It must be noted that the number of maximum iterations allocated should be adapted to the occupation rate of the container. Indeed, for the 60% occupation rate, the first feasible solution is provided around the 2500th iteration which corresponds to half of the optimization process. On the contrary, the 30% occupation rate provides a first solution in a few hundreds of iterations.

Final numerical results. Table 4.2 sums up the numerical results corresponding to the final obtained layouts.

The numerical results first show that the CSO-VF algorithm enhanced with the LHS initialization and the swap operator allows to improve the success rate in comparison to the randomly initialized and no swap CSO-VF configuration. Indeed, from the 40% occupation rate, the success rate is improved respectively by 8%, 22%

	OR	R	LS
Nb of successful runs (a feasible solution is found)	30%	**50**	**50**
	40%	46	**50**
	50%	39	**50**
	60%	18	**31**
Final objective function (median)	30%	1.665e7	**1.641e7**
	40%	2.234e7	**2.192e7**
	50%	2.770e7	**2.736e7**
	60%	3.335e7	**3.221e7**
Final interquartile range (IQR)	30%	**9.759e5**	1.040e6
	40%	1.379e6	**1.261e6**
	50%	**1.678e6**	1.715e6
	60%	**1.729e7**	2.421e6
Best layout (objective function)	30%	1.527e7	**1.464e7**
	40%	1.990e7	**1.971e7**
	50%	2.491e7	**2.425e7**
	60%	3.107e7	**3.019e7**

Table 4.2: Numerical results obtained for the four configurations of initialization and swap operator and for the three occupation rates.

and 53%. It is due to the fact that with the random initialization, the first feasible solution is reached later than the other configuration. Since higher occupation rates increase the difficulty to solve the constraints, less feasible solutions might be reached during the given maximum number of iterations. Moreover, the median of the final objective function as well as the best obtained layout are all better for the LHS initialization and swap configuration. The interquartile range is however mostly smaller for the random initialization and no swap CSO-VF algorithm configuration.

In Appendix A, successive feasible layout throughout the iterations are shown, for one instance of the CSO-VF algorithm, using the 40% occupation rate and the LS configuration.

4.2.3.5 Analysis of the computing time

The CSO-VF algorithm relies on the computation of forces and torques which number of operations directly depends on the number of components to be laid out. The swap operator has also an impact on the computing time as it loops over all the pairs of components. However, because its call frequency depends on the current convergence state and therefore on the initialization, it might be difficult to quantify its influence on the computing time. 50 randomly initialized instances of the CSO-VF algorithm are run for 10 to 50 components and 1000 to 10000 iterations. The occupation rate do not influence the computing time and is kept constant equal to 40% for all the instances. The routines are implemented in Python language, and executed on a PC with an Intel Core i7, 16 GB of RAM and Windows operating system. The means of the computing time for each configuration are reported (in minutes) in Figure 4.15.

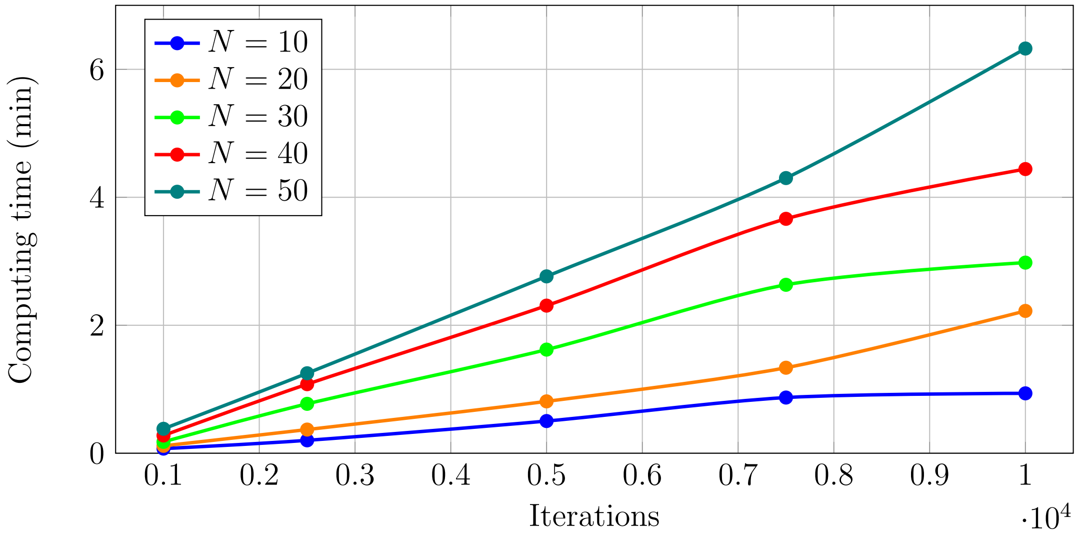


Figure 4.15: Mean of the computing time for $N = 10$ to $N = 50$ components and 1000 to 10000 iterations.

It is observed that the computing time tends to vary linearly with the number of components and the number of iterations. Moreover, the computing times of the CSO-VF instances are only a few minutes long with the allocated numerical resources. For example, instances with 30 components and 5000 iterations are solved in less than 2 minutes. It must be noted that the computing time strongly depends on the implementation language used as well as the numerical resources allocated and is therefore hardly comparable with other computing times from the literature. Furthermore, the CSO-VF algorithm is compatible with GPU programming that would considerably improve the computing time. However, from an industrial point of view, the effective computing times remain reasonable.

4.2.4 Global performance of the CSO-VF algorithm

This section aims at analyzing the global performance of the CSO-VF algorithm in comparison to other metaheuristic algorithms: Genetic Algorithm (GA) [Joh82] and Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) [HO01]. Indeed, as outlined in Chapter 2, metaheuristics are the most employed techniques for OLPs. Moreover, they provide generic framework that can be easily adapted to the problems at hand. Finally, GA and CMA-ES are two different strategies which allow to handle constrained problems with a potential large number of variables.

4.2.4.1 Experimental setups

Increasing occupation rates from 30% to 60% are studied. For each occupation rate, the experimental setups are defined and the following parameters are set:

- **Computational budget of objective function evaluations:** for each occupation rate, a maximum number of objective function evaluations is set. As mentioned in the previous section, the higher the occupation rate, the harder it is to satisfy the constraints. Thus, the number of objective function evaluations is increased along with the occupation rate. The computational budgets are summarized in Tables 4.3, 4.4, 4.5 and 4.6.

- **Population size:** for each occupation rate, three population sizes are studied. For the CSO-VF algorithm the population size corresponds to the number of independent starts as explained in Chapter 3. For each occupation rate, the CSO-VF configuration with only one start *i.e.*, with no multistart employed and two configurations employing multistart with increasing number of starts are studied. For the GAs, three increasing population sizes are studied. Moreover, the population sizes are globally increased along with the occupation rate, for the same reason than for the budget settings. For CMA-ES, the population size follows the recommendations found in the literature [HO01; Var+18]. It must be noted that the maximum number of iterations N_{it} of each configuration is set based on the budget and population size parameters P as:
$$N_{it} = \frac{Budget}{P} \tag{4.15}$$
The studied population sizes are summarized in Tables 4.3, 4.4, 4.5 and 4.6.

- **Number of simulations:** each configuration, defined by an occupation rate, an algorithm and a population size, is run over 50 independent initializations. It must also be noted that each simulation of the CSO-VF and GA algorithms are initialized using the same initialized design variables.

- **Configuration of operators and hyperparameters:** The CSO-VF configuration has been detailed in the previous section. The GA's appropriate operators are chosen on the basis of a parametric analysis: Tournament selector, SBX crossover operator, PM mutation operator. The hyperparameters of both algorithms are optimized using Bayesian Optimization. The $(1 + \lambda)$ CMA-ES parameters are set using standard settings detailed in the literature [Var+18].

Tables 4.3, 4.4, 4.5 and 4.6 sum up the experimental setups for each studied occupation rate.

Algorithm	Budget=200000	Nb of simulations	HP settings
CSO-VF	P1=1,P2=10,P3=50	50	Optimized with BO
GA	P1=10,P2=50,P3=100	50	Optimized with BO
CMA-ES	P=24	50	Standard settings

Table 4.3: Experimental setups for the 30% occupation rate configurations. P1, P2 and P3 stand for different number of individuals.

Algorithm	Budget=350000	Nb of simulations	HP settings
CSO-VF	P1=1,P2=25,P3=50	50	Optimized with BO
GA	P1=25,P2=50,P3=150	50	Optimized with BO
CMA-ES	P=24	50	Standard settings

Table 4.4: Experimental setups for the 40% occupation rate configurations. P1, P2 and P3 stand for different number of individuals.

Algorithm	Budget=500000	Nb of simulations	HP settings
CSO-VF	P1=1,P2=50,P3=100	50	Optimized with BO
GA	P1=50,P2=100,P3=200	50	Optimized with BO
CMA-ES	P=24	50	Standard settings

Table 4.5: Experimental setups for the 50% occupation rate configurations. P1, P2 and P3 stand for different number of individuals.

Algorithm	Budget=750000	Nb of simulations	HP settings
CSO-VF	P1=1,P2=100,P3=150	50	Optimized with BO
GA	P1=100,P2=150,P3=250	50	Optimized with BO
CMA-ES	P=24	50	Standard settings

Table 4.6: Experimental setups for the 60% occupation rate configurations. P1, P2 and P3 stand for different number of individuals.

4.2.4.2 Results and analysis

In order to analyze the performance of each algorithm, the following metrics are adopted:

- **Median of convergence curves:** For each occupation rate, Figures 4.16a, 4.19a, 4.22a and 4.25a show the median of convergence curves of the 50 simulations for each configuration characterized by an algorithm and a population size.

- **Best run:** For each occupation rate, Figures 4.16b, 4.19b, 4.22b and 4.25b show the convergence curves of the simulation leading to the best layout *i.e.* the smaller final inertia, for each configuration.

- **Iterations to best median (mean):** For each occupation rate, Figures 4.17, 4.20, 4.23 and 4.26 show the mean of iterations needed by each median convergence curve to reach 25%, 20%, 15%, 10% and 5% of the best final obtained median over the configurations (defined by the different algorithms and population sizes). This metric characterizes both the convergence speed and the relative margin between each configuration and the best one in terms of median convergence curve.

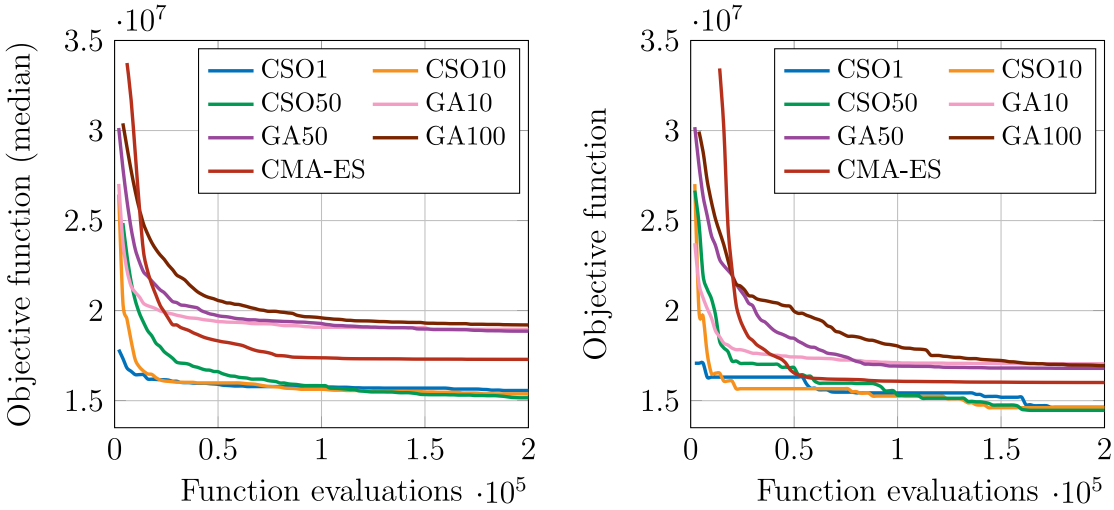
- **Successful runs to target:** For each occupation rate, Figures 4.18, 4.21, 4.24 and 4.27 show the number of simulations of each configurations (among the 50) which manages to reach 25%, 15%, 10%, 5%, 3% and 1% of the best obtained median. This metric characterizes the success rate and the robustness of each configuration.

- **Ranks of the algorithms:** Figure 4.28 indicates for each occupation rate the number of times each algorithm ranked from first to seventh place in terms of final objective function value for the 50 simulations.

Table 4.7 sums up the numerical results obtained for each configurations *i.e.* occupation rate, algorithm and population sizes. The metrics are:

- The success rate, *i.e.* the number of simulations providing a feasible solution;
- The median of the final objective function values;
- The interquartile range of the final objective function values;
- The best objective function values over the 50 simulations;
- The mean of the iteration for which a feasible solution is reached for the first time.

Finally, Figure 4.29 shows the best obtained layouts for each occupation rates.



(a) Medians, $OR = 30\%$ (b) Best run, $OR = 30\%$

Figure 4.16: Median and best convergence curves for the 30% occupation rate.

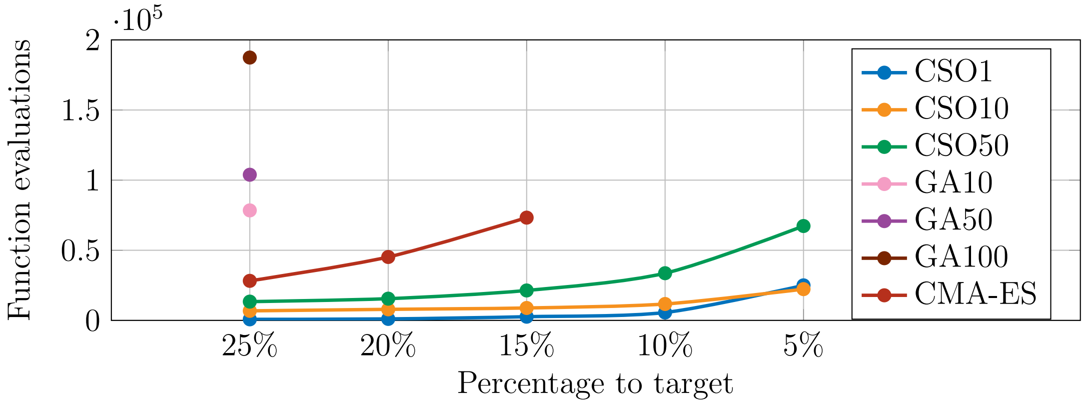


Figure 4.17: Function evaluations to percentages of the median target, $OR = 30\%$.

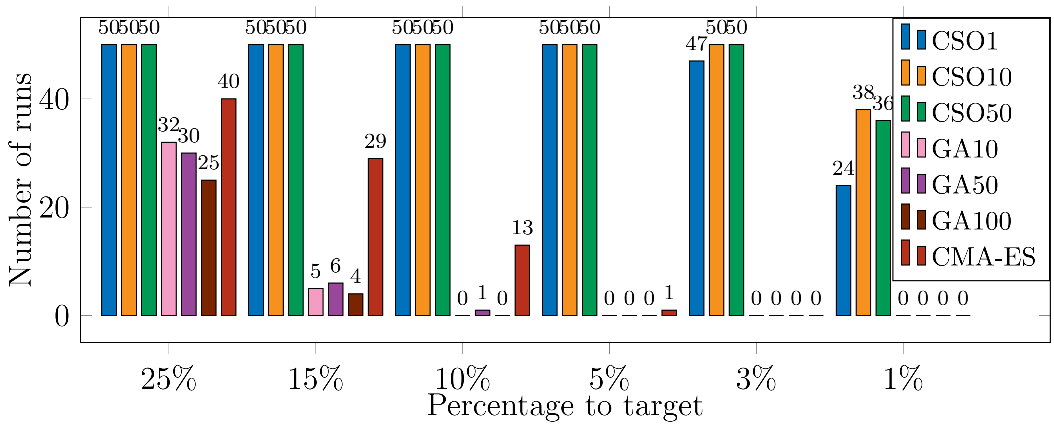
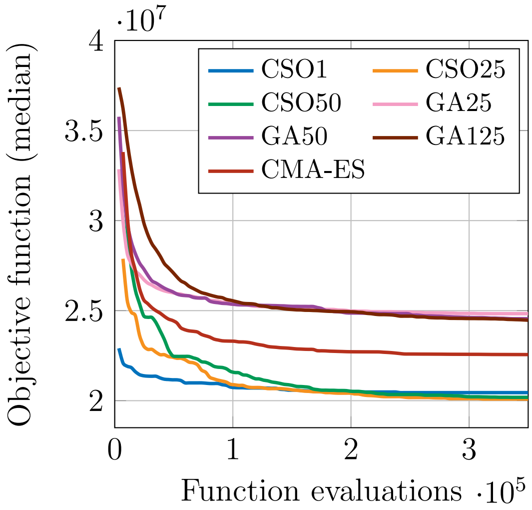
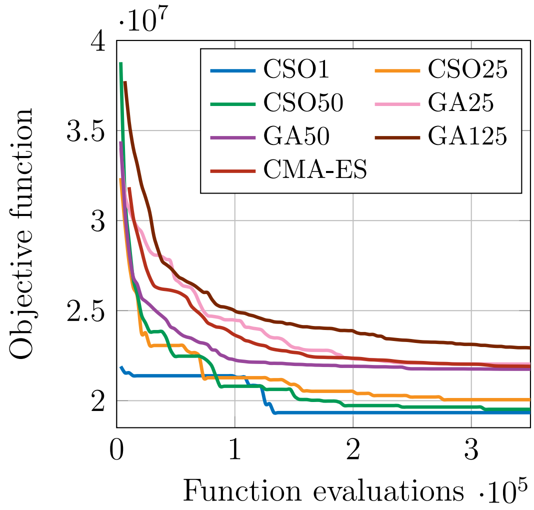


Figure 4.18: Success rate to percentages of the median target, $OR = 30\%$.



(a) Medians, $OR = 40\%$



(b) Best run, $OR = 40\%$

Figure 4.19: Median and best convergence curves for the 40% occupation rate.

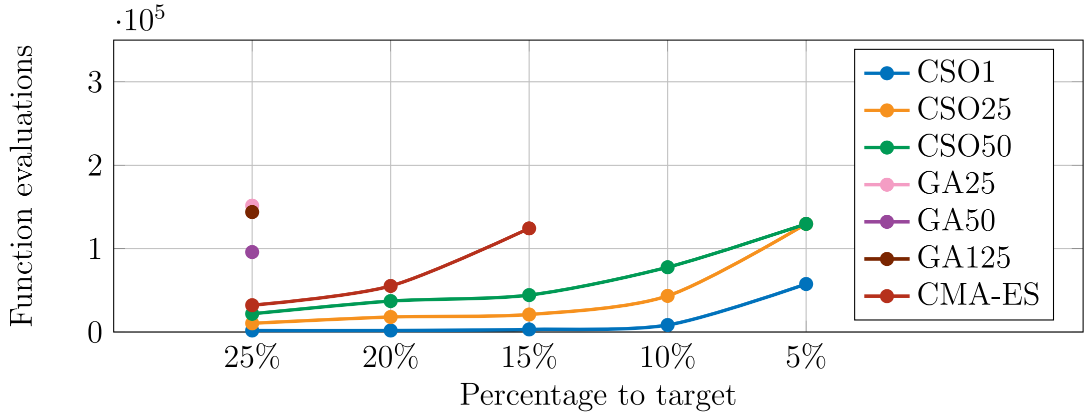


Figure 4.20: Function evaluations to percentages of the median target, $OR = 40\%$.

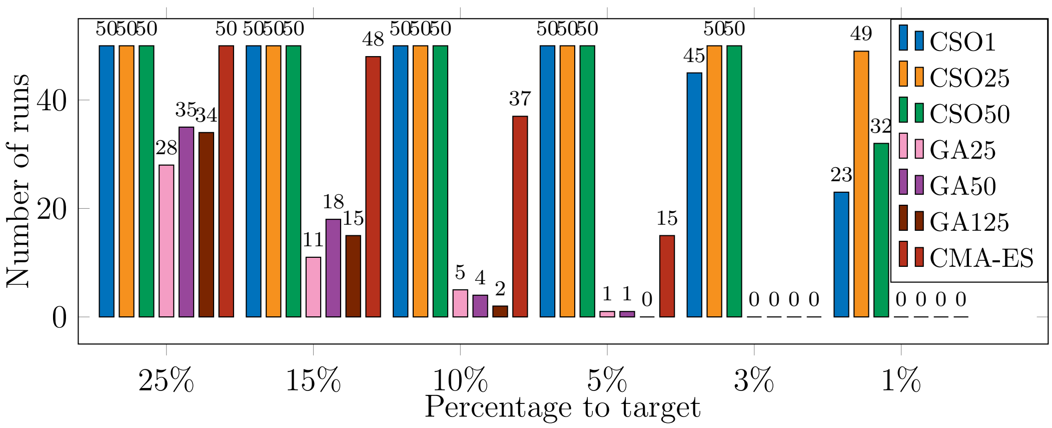
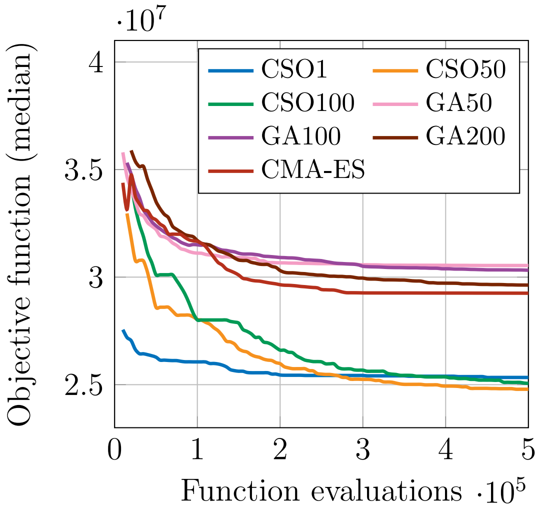
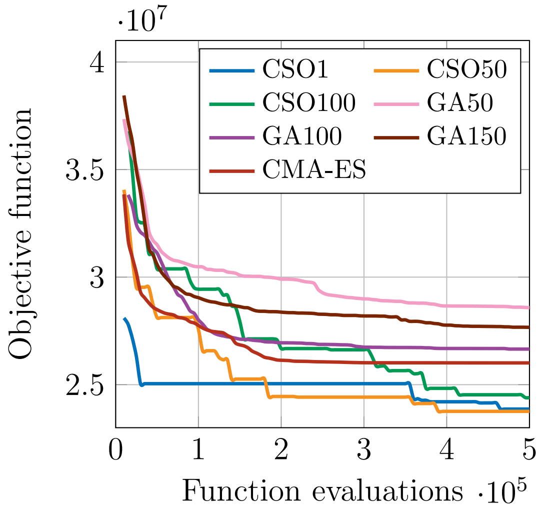


Figure 4.21: Success rate to percentages of the median target, $OR = 40\%$.



(a) Median, $OR = 50\%$



(b) Best run, $OR = 50\%$

Figure 4.22: Median and best convergence curves for the 50% occupation rates.

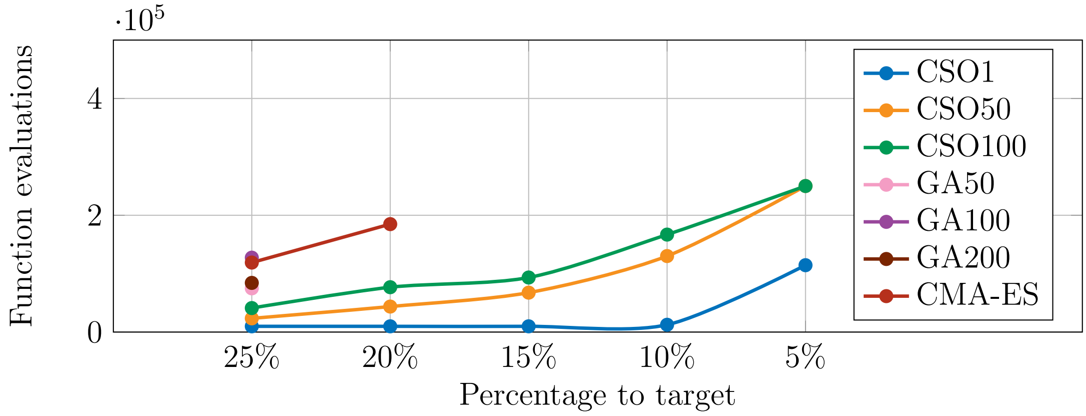


Figure 4.23: Function evaluations to percentages of the median target, $OR = 50\%$.

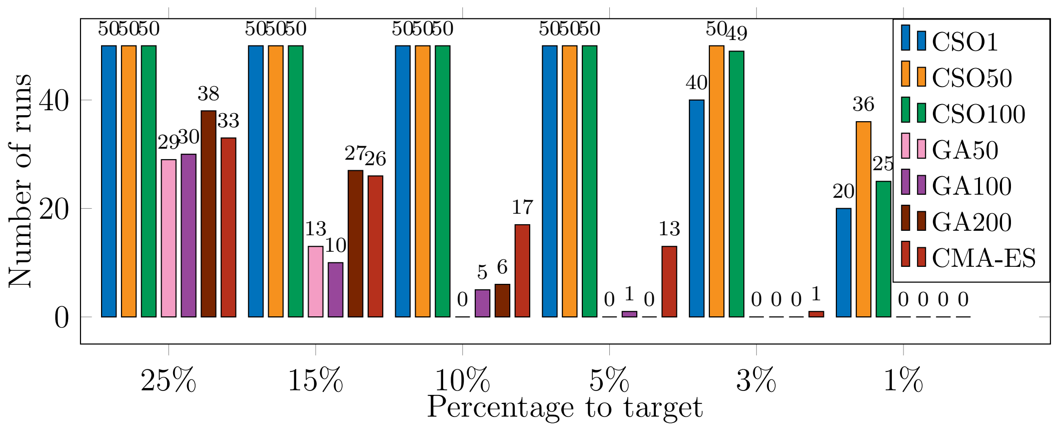
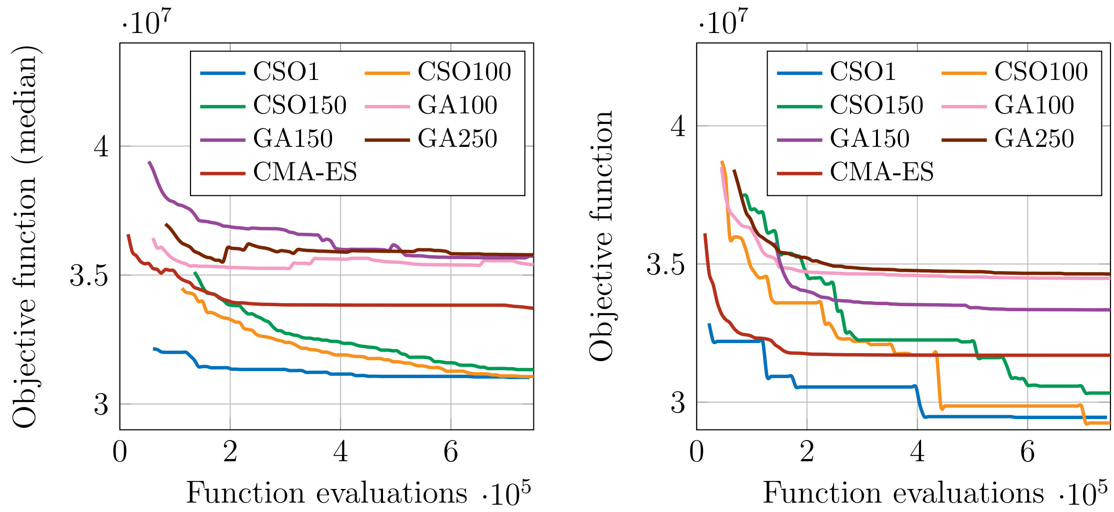


Figure 4.24: Success rate to percentages of the median target, $OR = 30\%$.



(a) Median, $OR = 60\%$

(b) Best run, $OR = 60\%$

Figure 4.25: Median and best convergence curves for the 60% occupation rate.

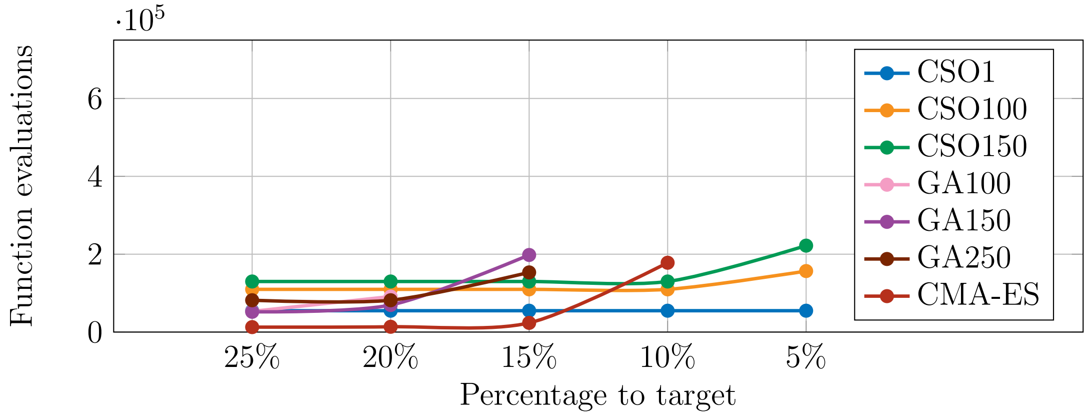


Figure 4.26: Function evaluations to percentages of the median target, $OR = 60\%$.

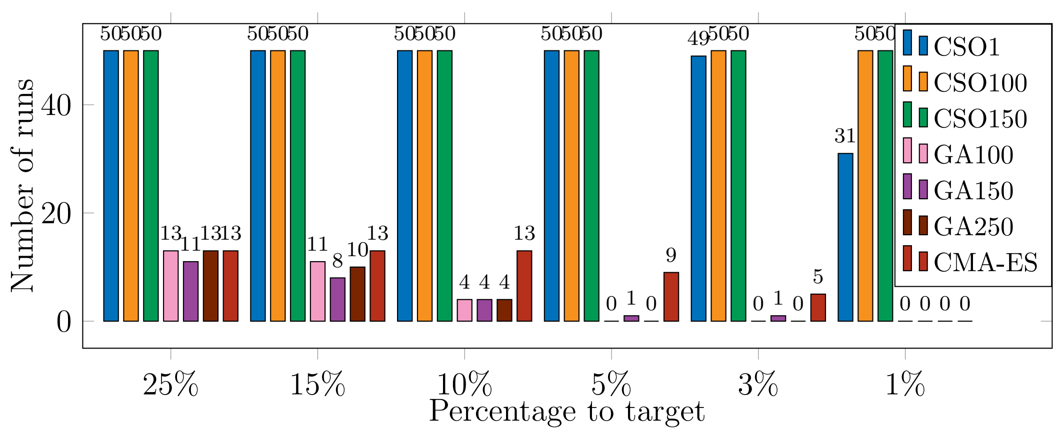
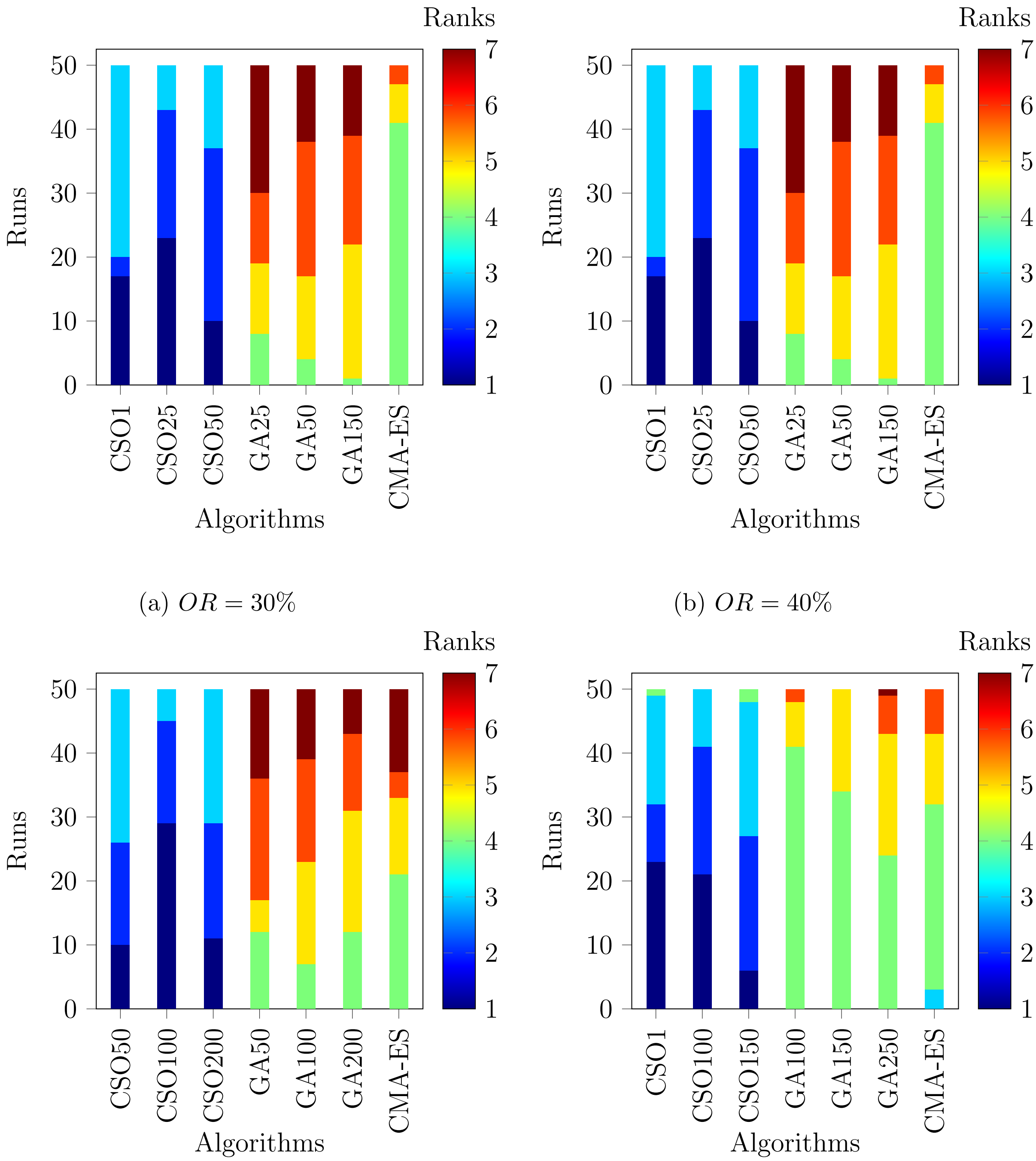


Figure 4.27: Success rate to percentages of the median target, $OR = 60\%$.

	OR	CSO-1	CSO-P2	CSO-P3	GA-P1	GA-P2	GA-P3	CMA-ES
Success rate	30%	**50**	**50**	**50**	**50**	**50**	**50**	**50**
	40%	**50**	**50**	**50**	46	49	49	**50**
	50%	**50**	**50**	**50**	44	44	43	41
	60%	**50**	**50**	**50**	15	13	15	21
Final objective (median)	30%	1.557e7	1.537e7	**1.516e7**	1.896e7	1.886e7	1.920e7	1.729e7
	40%	2.045e7	**2.008e7**	2.019e7	2.483e7	2.455e7	2.447e	2.256e7
	50%	2.534e7	**2.479e7**	2.506e7	3.050e7	3.030e7	2.961e7	2.925e7
	60%	3.103e7	**3.100e7**	3.133e7	3.539e7	3.573e7	3.579e7	3.350e7
Final IQR	30%	5.724e5	**4.574e5**	4.589e5	1.240e6	1.412e6	1.074e6	1.274e6
	40%	9.360e5	3.721e5	**2.746e5**	1.233e6	1.694e6	1.316e6	9.556e5
	50%	1.045e6	5.409e5	**4.232e5**	1.991e6	1.967e6	1.460e6	4.10e6
	60%	1.070e6	7.616e5	**6.433e5**	1.832e6	2.054e6	2.26e6	1.792e6
Best layout	30%	1.463e7	1.461e7	**1.446e7**	1.705e7	1.680e7	1.694e7	1.601e7
	40%	1.934e7	**1.928e7**	1.952e7	2.203e7	2.175e7	2.293e7	2.191e7
	50%	2.387e7	**2.376e7**	2.440e7	2.692e7	2.634e7	2.767e7	2.601e7
	60%	2.937e7	**2.925e7**	3.032e7	3.448e6	3.334e7	3.463e7	3.110e7
First feasible solution (mean)	30%	**190**	216	248	964	1787	3604	738
	40%	474	**299**	511	1851	2039	4140	7388
	50%	**1288**	1639	1348	20295	13727	24650	12240
	60%	9503	2908	**2338**	141882	240750	153470	37119

Table 4.7: Numerical results for the three algorithms, different sizes of population and the four occupation rates. Bold values indicate best algorithm.



(a) $OR = 30\%$

(b) $OR = 40\%$

(c) $OR = 50\%$

(d) $OR = 60\%$

Figure 4.28: Ranks of each algorithm over the 50 runs and the four occupation rates.

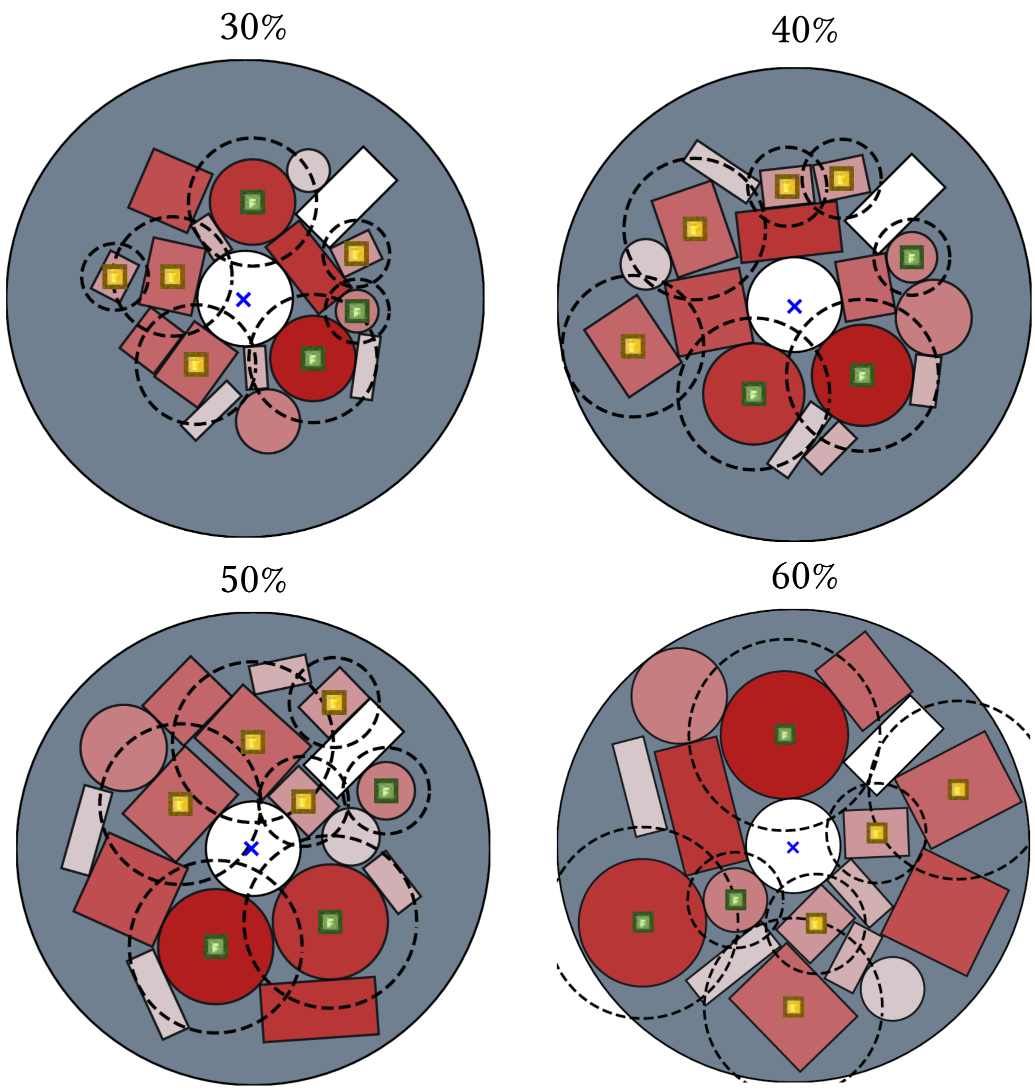


Figure 4.29: Best obtained layouts for the four occupation rates (all of them obtained by CSO-VF algorithm).

Success rate. Table 4.7 shows that all the CSO-VF algorithms have a success rate of 100% for all the occupation rates. In other words, for each occupation rate, all 50 trials provide a feasible solution. In contrast, the higher the occupation rate, the lower the success rate for the GAs and the CMA-ES algorithm, reaching between 30% and 40% of success rate for the 60% occupation rate.

Convergence accuracy. For all the occupation rates, the 3 configurations of the CSO-VF algorithm have a better final median than the GAs and CMA-ES. Indeed, the best CSO-VF algorithm allows to improve the final median by respectively 24.4%, 21.9%, 19.4%, 14.2% in comparison to the best GAs configuration for increasing occupation rates. They are improved respectively by 14%, 12.4%, 18%, 8.1% in comparison to the CMA-ES algorithm and increasing occupation rates. In terms of best run, *i.e.* the simulation providing the best final layout in terms of final objective function values, the trends are similar to the median observations. Indeed, all 3 CSO-VF algorithms provide the 3 best runs in terms of final objective function values and this for all the occupation rates.

Figure 4.28 ranks the values of the final objective function obtained by each algorithm over the 50 instances, for each occupation rates. It is observed that the CSO-VF algorithms practically always rank first to third while the GA and CMA-ES algorithms generally rank among the last ones. CMA-ES algorithms have generally better ranks than the GA algorithms but this trend is less pronounced for high occupation rates.

It is shown that the CSO-VF algorithm configuration employing multistart with the less starts but the more iterations, *i.e.* CSO-P2, performs better than the CSO-VF configuration with no multistart and than the CSO-VF configuration with more starts but less iterations (*i.e.*, CSO-P3) in terms of final median of the objective function and best layouts, from the 40% to the 60% occupation rates. Indeed, one initialization of the CSO-VF is not enough as the algorithm depends on the initialization and too many starts tend to reduce the number of iterations (as the number of objective function evaluations is fixed) and prevent each start from converging. This trend is also observed on Figure 4.28 showing that the CSO-P2 configuration always has the smallest ranks in comparison to the two other CSO-VF configurations. Consequently, a balance between the number of starts employed in the multistart and the number of iterations must be found.

The trends are more mixed for the GAs. Indeed, in terms of final median of the objective function no real trend can be drawn as the GAs configurations provide close values giving no advantage to one in particular. However, the intermediate configuration (*i.e* GA-P2) always provides the best run in terms of final objective function values. Thus, for the GAs a balance between the size of the population and the number of iterations should be considered.

Convergence speed. The third metric shows how many function evaluations are needed in order to reach certain percentages of the best final median obtained *i.e.* the final median of the CSO-VF algorithm with the lowest number of starts. It is observed that the GAs reach only 25% to 15% of this value according to the occupation rates. Moreover, the number of function evaluations needed to reach this value are globally higher than the three CSO-VF algorithms. CMA-ES performs better than the GAs as it reaches 15% to 10% of the target value.

The CSO-VF algorithm configuration with no multistart and more iterations *i.e.* CSO-1, provides the highest convergence speed for the three lower occupation rates. Indeed, the multistart configuration implies a multiplication of the number of evaluation functions at each iteration. For the 60% occupation rate, CMA-ES is faster until 15% of the target value and the CSO-P1, *i.e.* CSO-1 configuration is then faster because CMA-ES reached its convergence.

Robustness. For each occupation rate, Figures 4.18, 4.21, 4.24 and 4.27 show the number of runs among the 50 trials that reach a certain percentages of the best final median value *i.e.*, the target value. For all the occupation rates, all the 50 CSO-VF instances reach at least 5% of this target value. On the contrary, the GAs have at least 20% of the runs that do not reach 25% of the target values and only a very few runs reach 5% of the target value. The CMA-ES algorithm has better results than the GAs. Indeed, from 1 to 15 runs reach 5% of the target value depending on the occupation rate. It can be concluded that the CSO-VF algorithms have a better robustness with respect to the number of runs reaching the target value than the other GA and CMA-ES counterparts. This conclusion can be confirmed in Table 4.7 that reports the final interquartile range for each algorithm, each population size and each occupation rate. The best IQR obtained by the CSO-VF algorithms is respectively improved by 76%, 77.7%, 71%, 64.9% in comparison to the best IQR obtained by the GAs for increasing occupation rates. It is respectively improved by 79.8%, 71.3%, 89.7%, 64.1% in comparison to IQR obtained by the CMA-ES algorithm, for increasing occupation rates.

The final IQR values also show that the CSO-VF configurations that employ multistart have better robustness than the CSO-1 configurations that do not employ multistart. Indeed, the CSO-VF with multistart configurations improve the final IQR by respectively 20.1%, 70.7%, 59.5% and 39.9% in comparison to the CSO-1 configuration and the four occupation rates.

First feasible solution. Finally, the CSO-VF algorithms provide a first feasible solution in less function evaluations than the GAs and CMA-ES. The GAs and CMA-ES are population-based algorithms while the CSO-VF is based on multistarts. Thus, the number of function evaluations needed to reach a feasible solution for GAs and CMA-ES is calculated as the iteration at which the first feasible solution is observed times the size of the population. However, the first feasible solution obtained for one run (*i.e.*, with P multiple starting points) of the CSO-VF corresponds directly to the number of function evaluations as the starts of the CSO-VF are independent. It must be noted that for all algorithms, the number of function evaluations needed to provide a feasible solution increases with the occupation rate due to the increasing difficulty to solve the constraints.

Generally, the CSO-VF algorithm provides better global results in terms of success rate, convergence accuracy, robustness with respect to the final dispersion, convergence speed and ability to solve the constraints in comparison to the GA and CMA-ES algorithm counterparts. This is due to the fact that the CSO-VF algorithm provides dedicated operators to solve each of the constraints using the virtual-force system and the swap operator. Population-based algorithms dealing with the constraints with penalization or constraint-dominance have less ability to deal with the constraints. Indeed, during the optimization process, solutions with less constraint violations are favored in the population. However, the initialized layouts with larger initial constraint violation are not necessarily the most promising configuration in terms of objective function. On the other hand, CSO-VF optimizes all the initialized layouts which allows to identify more accurately and robustly the promising initializations using multistart. Moreover, the multistart allows CSO-VF to provide first feasible solutions in less function evaluations than population-based counterparts as the runs are independent. Finally, the swap operator allows to find new and better configurations by inducing a reconfiguration of the layout when one configuration has converged, while the studied population-based algorithms do not provide sufficiently effective reconfiguration operator when its population has converged.

The results also highlighted that the CSO-VF should be employed with multistart in order to improve the convergence accuracy and the robustness. However, a balance must be found between the number of starts and the number of iterations as too many starts lead to too few iterations and prevent the algorithm from converging. Moreover, it should be noted that for identical number of objective function evaluations, the CSO-VF algorithm employing multistarts contributes to reduce the computing time in comparison to the CSO-1 configuration as the starts can be run in parallel.

4.3 Multi-container Satellite Module Optimal Layout Problem

The multi-container configuration corresponds to the SMLP studied for instance in [CXW18; CZa19; Ten+09; ZTS08].

4.3.1 Problem formulation and configuration

4.3.1.1 Geometry of the container and the components

The container corresponds to two bearing plates defined by their radius R_{out} and their thickness H_p. They are disposed at the heights H_1 and H_2 in a cylindrical module of height H_3. Thus, four surfaces S_k, $k \in \{1, 2, 3, 4\}$ are laid out. A central column of radius R_{in} results in circular exclusion zones centered at the z axis on each surface. The whole module is characterized by its mass m_{sat} and its geometrical inertias along each axis: $I_{x,sat}, I_{y,sat}$, $I_{z,sat}$ resulting in a global inertia defined as $I_{sat} = I_{x,sat} + I_{y,sat} + I_{z,sat}$. The empty module has its center of gravity positioned in $(x_{x,sat}, x_{y,sat}, x_{z,sat})$. Figure 4.30 illustrates the multi-container configuration.

The $N = 60$ components to be positioned are N_{cub} cuboids and N_{cyl} cylinders and are initially proposed in [Ten+09]. They are listed in Table A.2 in Appendix A as well as the numerical values of the dimensions and dynamical features of the container.

4.3.1.2 Design variables

In the multi-container configuration, the components must be assigned to one of the four surfaces and be laid out on the corresponding surface, therefore the design variables are:

- For each component, the number of its assigned surface z_i, $i \in \{1, ..., N\}$ defined as unordered discrete variables;

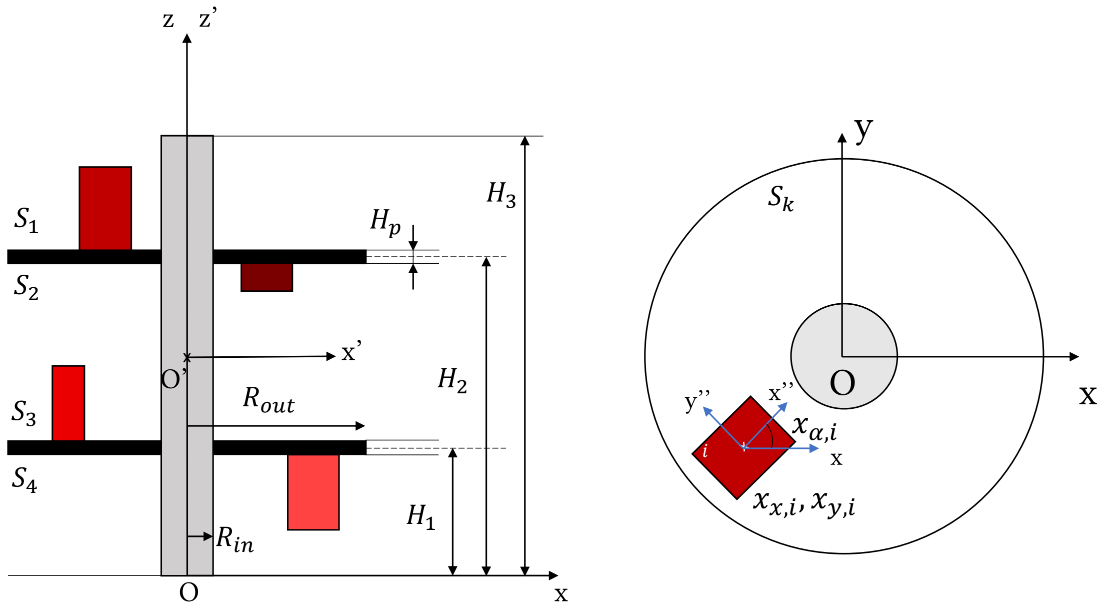


Figure 4.30: Geometrical definition of the multi-container configuration.

- The positions of the center of inertia of each component on the assigned surface considered as continuous variables $(x_{x,i}, x_{y,i})$, $i \in \{1, ..., N\}$;
- The orientations of cuboid components on the assigned surface are considered as discrete variables with values 0° or 90°, $z_{\alpha,i}$, $i \in \{1, ..., N_{cub}\}$.

Thus, the number of design variables n_{var} is defined as:

$$n_{var} = N + 2N_{cyl} + 3N_{cub} = 3N_{cyl} + 4N_{cub} \tag{4.16}$$

4.3.1.3 Objective function

As for the single-container configuration, the objective function to minimize is the total inertia of the module. Three systems of coordinates are considered:

- $Ox"y"z"$: the local coordinates' system related to each component. $O"$ corresponds to the center of inertia of the component and the axes are defined with the symmetry planes of the components;
- $O'x'y'z'$: the coordinates system related to the system of components. O' represents the current centroid of the system of components and the axes are defined with the symmetry planes of the module;
- $Oxyz$: the coordinates system related to the module. O is the geometric center of the container and the axes are defined using its symmetry planes.

The three systems of coordinates are illustrated in Figure 4.30.

The global inertia I_{tot} to minimize is calculated in the $O'x'y'z'$ system of coordinates as proposed in [CZa19; Ten+09; ZTS08] and defined as:

$$I_{tot}(\mathbf{x}, \mathbf{z}) = I_{x'x'}(\mathbf{x}, \mathbf{z}) + I_{y'y'}(\mathbf{x}, \mathbf{z}) + I_{z'z'}(\mathbf{x}, \mathbf{z}) \tag{4.17}$$

With the same formalism as the previous section, the solid inertia specific to each component with relation to their system of coordinates $O"x"y"z"$ are written $I_{x",i}$, $I_{y",i}$ and $I_{z",i}$.

The inertia of the module along each axis of the $Oxyz$ system of coordinates are written:

$$I_{xx}(\mathbf{x}, \mathbf{z}) = I_{x,sat} + \sum_{i=1}^{N} I_{x",i}\cos(z_{\alpha,i})^2 + I_{y",i}\sin(z_{\alpha,i})^2 + m_i(x_{y,i}^2 + x_{z,i}^2) \tag{4.18}$$

$$I_{yy}(\mathbf{x}, \mathbf{z}) = I_{y,sat} + \sum_{i=1}^{N} I_{y",i}\cos(z_{\alpha,i})^2 + I_{x",i}\sin(z_{\alpha,i})^2 + m_i(x_{x,i}^2 + x_{z,i}^2) \tag{4.19}$$

$$I_{zz}(\mathbf{x}, \mathbf{z}) = I_{z,sat} + \sum_{i=1}^{N} I_{z",i} + m_i(x_{x,i}^2 + x_{y,i}^2) \tag{4.20}$$

where $(x_{x,i}, x_{y,i}, x_{z,i})$ are the coordinates of component i in the $Oxyz$ system of coordinates. Then, the inertia are calculated along the axes of the $O'x'y'z'$ system of coordinates using the Huygens theorem:

$$I_{x'x'}(\mathbf{x}, \mathbf{z}) = I_{xx}(\mathbf{x}, \mathbf{z}) - (x_{y,c}^2 + x_{z,c}^2)m_{tot} \tag{4.21}$$

$$I_{y'y'}(\mathbf{x}, \mathbf{z}) = I_{yy}(\mathbf{x}, \mathbf{z}) - (x_{x,c}^2 + x_{z,c}^2)m_{tot} \tag{4.22}$$

$$I_{z'z'}(\mathbf{x}, \mathbf{z}) = I_{zz}(\mathbf{x}, \mathbf{z}) - (x_{x,c}^2 + x_{y,c}^2)m_{tot} \tag{4.23}$$

where $(x_{x,c}, x_{y,c}, x_{z,c})$ are the coordinates of the center of gravity of the module in the $Oxyz$ system of coordinates and $m_{tot} = (m_{sat} + \sum_{i=1}^{N} m_i)$ is the mass of the whole module.

4.3.1.4 Constraints functions

As for the single-container configuration, geometrical and functional constraints are considered:

- **Overlapping constraints between components:** no overlapping between components is allowed on each surface. As in [CZa19; Ten+09; ZTS08], the plates are supposed to be sufficiently spaced to avoid any overlapping between components positioned on the surfaces S_2 and S_3. The overlapping constraint between components is formulated as follows:

$$h_{overlap}^{C}(\mathbf{x}, \mathbf{z}) = \sum_{k=1}^{N_s} \sum_{i=1}^{N_k-1} \sum_{j=1}^{N_k} \Delta A_{ij,k}^{C}(\mathbf{x}, \mathbf{z}) \tag{4.24}$$

  where $N_s = 4$ is the number of surfaces, N_k is the number of components on surface k and $\Delta A_{ij,k}^{C}$ corresponds to the 2-dimensional projections (along the z axis) of components i and j on surface k and is function of the components centers of inertia and orientations.

- **Overlapping constraints between the components and the exclusion zones:** no overlapping between the components and the exclusion zones is allowed on each surface. The overlapping constraint is expressed as:

$$h^{E}_{overlap}(\mathbf{x},\mathbf{z}) = \sum_{k=1}^{N_s}\sum_{i=1}^{N} \Delta A^{E}_{i,k}(\mathbf{x},\mathbf{z}) \tag{4.25}$$

where $\Delta A^{E}_{i,k}$ is the are of intersection between the 2-dimensional projections (along the z axis) of a component and the exclusion zone on the surface k, and is function of the components centers of inertia and orientations (as well as the positions and shapes of the exclusion zones).

- **Balancing constraints:** the center of gravity (CG) of the whole laid out module must be positioned within a zone of tolerance located at the center of gravity of the empty module, along the x and y axes. Then, the balancing constraint can be formulated as follows:

$$g_{CG}(\mathbf{x},\mathbf{z}) = \sqrt{(x_{x,c} - x_{x,sat})^2 + (x_{y,c} - x_{y,sat})^2} - \delta_{CG} \tag{4.26}$$

where $(x_{x,c}, x_{y,c}, x_{z,c})$ are the coordinates of the current CG of the whole module also referred to as O' in the $Oxyz$ system of coordinates, $(x_{x,sat}, x_{y,sat}, x_{z,sat})$ are the coordinates of the empty satellite module in the $Oxyz$ system of coordinates and δ_{CG} represents a tolerance which corresponds to a sphere centered at the empty satellite module center of gravity and set to the numerical value of 3.0.

- **Angles-of-inertia constraints**: in the multi-container configuration, a geometrical constraint corresponding to the angles of inertia is added. The constraint relative to the angles of inertia is defined as follows:

$$g_{AI}(\mathbf{x},\mathbf{z}) = \sqrt{\theta_{x'y'}(\mathbf{x},\mathbf{z})^2 + \theta_{y'z'}(\mathbf{x},\mathbf{z})^2 + \theta_{y'z'}(\mathbf{x},\mathbf{z})^2} - \delta_{AI} \tag{4.27}$$

where $\theta_{x'y'}$, $\theta_{x'y'}$, $\theta_{y'z'}$ are the angles of inertia, defined as:

$$\theta_{x'y'}(\mathbf{x},\mathbf{z}) = \arctan\left(-\frac{2I_{x'y'}(\mathbf{x},\mathbf{z})}{(I_{y'y'}(\mathbf{x},\mathbf{z}) - I_{x'x'}(\mathbf{x},\mathbf{z}))}\right)\Big/2 \tag{4.28}$$

$$\theta_{x'z'}(\mathbf{x},\mathbf{z}) = \arctan\left(-\frac{2I_{x'z'}(\mathbf{x},\mathbf{z})}{(I_{x'x'}(\mathbf{x},\mathbf{z}) - I_{z'z'}(\mathbf{x},\mathbf{z}))}\right)\Big/2 \tag{4.29}$$

$$\theta_{y'z'}(\mathbf{x},\mathbf{z}) = \arctan\left(-\frac{2I_{y'z'}(\mathbf{x},\mathbf{z})}{(I_{y'y'}(\mathbf{x},\mathbf{z}) - I_{z'z'}(\mathbf{x},\mathbf{z}))}\right)\Big/2 \tag{4.30}$$

where the inertias $I_{x'y'}$, $I_{y'z'}$ and $I_{y'z'}$ are expressed as:

$$\begin{aligned} I_{x'y'}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{N} & \left[m_i x_{x,i} x_{y,i} + \frac{\begin{array}{c} I_{x"i} + m_i(x_{y,i}^2 + x_{z,i}^2) \\ - I_{y"i} - m_i(x_{x,i}^2 + x_{z,i}^2) \end{array}}{2} \sin(2x_{\alpha,i}) \right] \\ & - x_{x,c} x_{y,c} m_{tot} \end{aligned} \tag{4.31}$$

$$I_{x'z'}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{N} m_i x_{x,i} x_{z,i} - x_{x,c} x_{z,c} m_{tot} \tag{4.32}$$

$$I_{y'z'}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{N} m_i x_{y,i} x_{z,i} - x_{y,c} x_{z,c} m_{tot} \tag{4.33}$$

- **Functional constraints** are also defined in terms of incompatibility between several components. A heat as well as electromagnetic thresholds are defined such that some components responsible for heat or electromagnetic fields must be spaced away at given distances. More precisely, a set C_H of pairs of incompatible heat components $\{i,j\} \in C_H$ and a set C_E of pairs of incompatible electromagnetic components $\{i,j\} \in C_E$ are defined. Thus, the heat functional constraint is defined as:

$$g_H(\mathbf{x},\mathbf{z}) = \sum_{k=1}^{\mathrm{Card}(C_H)} d_H - d_k(\mathbf{x},\mathbf{z}) \tag{4.34}$$

  where $\mathrm{Card}(C_H)$ is the cardinal of the set C_H corresponding to the list of incompatible pairs of heat components, d_k is the distance between the two components of pair k if positioned on the same surface and d_H is the minimal distance between the two components of each pairs of C_H.

  With the same formalism, the electromagnetic functional constraint is defined as:

$$g_E(\mathbf{x},\mathbf{z}) = \sum_{k=1}^{\mathrm{Card}(C_E)} d_E - d_k(\mathbf{x},\mathbf{z}) \leq 0 \tag{4.35}$$

  where $\mathrm{Card}(C_E)$ is the cardinal of the set C_H corresponding to the list of incompatible pairs of electromagnetic components, d_k is the distance between the two components of pair k if positioned on the same surface and d_E is the minimal distance between the two components of each pairs of C_E. The list of heat and electromagnetic incompatible components is specified in Appendix A.

4.3.1.5 Mathematical formulation

Thus, fixed-size search space multi-module satellite optimal layout problems are mathematically defined as follows:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z}} \quad & I_{tot}(\mathbf{x},\mathbf{z}) \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x}, \mathbf{z} \in F_z \subseteq \mathbb{N}^{n_z} \\
\text{s.t.} \quad & \mathbf{h}^{C}_{overlap}(\mathbf{x},\mathbf{z}) = 0 \\
& \mathbf{h}^{E}_{overlap}(\mathbf{x},\mathbf{z}) = 0 \\
& g_{CG}(\mathbf{x},\mathbf{z}) \leq 0 \\
& g_{AI}(\mathbf{x},\mathbf{z}) \leq 0 \\
& \mathbf{g}_{H}(\mathbf{x},\mathbf{z}) \leq 0 \\
& \mathbf{g}_{E}(\mathbf{x},\mathbf{z}) \leq 0
\end{aligned} \tag{4.36}$$

Remarks on the inertia equations. In all the papers dealing with the SMLP [CXW18; CZa19; Liu12; Liu+18; Ten+09; ZTS08], the inertia are calculated along with the axes of the $O'x'y'z'$ system of coordinates corresponding to the system of coordinates linked to the current centroid of the whole module. It is obtained by calculating the inertia in the $Oxyz$ system of coordinates and then by applying the the Huygens theorem which leads to Equations 4.21, 4.22 and 4.23. However, as the objective function is to minimize the sum of the three aforementioned equations, it will also lead to a maximization of the three positive second terms:

$$(x_{y,c}^2 + x_{z,c}^2)m_{tot}$$
$$(x_{x,c}^2 + x_{z,c}^2)m_{tot}$$
$$(x_{x,c}^2 + x_{y,c}^2)m_{tot}$$

Thus, the formulation of the inertia also contributes to maximize the coordinates of the global center of gravity of the module. As the balancing constraint (Equation 4.25) bounds these coordinates to a tolerance zone, the optimization process will lead to a positioning of the center of gravity to the boundaries of the balancing constraint. However, the goal of the optimization process would rather be to minimize the inertia along with positioning the center of gravity as close as possible to a position of reference, which often corresponds to the geometrical center of the module or to the center of gravity of the empty module if different.

Moreover, the first terms of the inertia equations are calculated in the $Oxyz$ system of coordinates which is linked to the bottom of the satellite. Minimizing these quantities includes minimizing the terms $m_i x_{x,i}^2$, $m_i x_{y,i}^2$ but also $m_i x_{z,i}^2$. Then, the optimization process based on these inertia equations might overload the bottom surfaces S_3 and S_4 which leads to an unbalanced satellite module.

Consequently, in order to avoid the two aforementioned issues, the inertia equations should be written in the system of coordinates linked to the position of reference where the center of gravity has to be positioned, as developed in Appendix B. The equations have been corrected for the single-container configuration. However, for the present multi-container configuration, the equations employed in the literature *i.e.*, Equations 4.21, 4.22 and 4.23 are used in the following experiments in order to rigorously compare the results from the previous papers to the present study.

4.3.2 Algorithm configuration and experimental setups

The performance of the GA-assisted CSO-VF algorithm is compared with results obtained in previous papers [CZa19; Ten+09] regarding the problem detailed in last section. For this purpose, the experimental setups of the two stages of the proposed algorithm in Chapter 3 are detailed.

4.3.2.1 Upper stage settings

The upper stage solves the assignment task. In order to minimize the global inertia, the components should be distributed among the containers such that their center of gravity along the z axis is centered at the center of gravity of the empty module. Then, the assignment objective function consists in minimizing the terms of the global inertia equations depending on the $x_{z,i}$ variables while positioning the center of gravity of the components along the z axis in a tolerance zone centered at the empty module center of gravity and while not exceeding a 65% occupation rate per container. Indeed, higher occupation rates might lead to unfeasible layout configurations. The corresponding optimization problem is formulated as follows:

$$
\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{i=1}^{N} m_i (x_{z,i}(\mathbf{z}) - z_e)^2 \\
\text{w.r.t.} \quad & \mathbf{z} \in F_z \subseteq \{1,2,3,4\}^N \\
\text{s.t.} \quad & g_{CG}(\mathbf{z}) \leq 0 \\
& g_O^j(\mathbf{z}) \leq 0 \quad \text{for } j \in \{1,2,3,4\}
\end{aligned}
\tag{4.37}
$$

where $x_{z,i}$ are the positions of the center of inertia of the components of mass m_i depending on the assignment list $\mathbf{z}$. z_e is a position of reference which is taken as the center of gravity of the empty module. The inequality constraints are respectively:

- The balancing constraint:

$$
g_{CG}(\mathbf{z}) = \left| \frac{\sum_{i=1}^{N} m_i x_{z,i}}{\sum_{i=1}^{N} m_i} - z_e \right| - 3.0 \tag{4.38}
$$

- The occupation rate constraints defined for each surface j:

$$
g_O^j(\mathbf{z}) = \frac{A_j^{components}}{A_j^{container}} - 0.65 \tag{4.39}
$$

  where $A_j^{components}$ is the total area of the components assigned to container j and $A_j^{container}$ is the area of container j.

The operators of the GA are set using a parametric study and the hyperparameters are optimized using Bayesian Optimization following the procedure explained in Section 4.2.3.1.

4.3.2.2 Lower stage settings

The lower stage solving the layout task for each container corresponds to the CSO-VF algorithm. For each container, $P = 10$ independently initialized instances solves the corresponding single-container optimal layout problem. It is configured as follows:

- **Virtual-force system:** The defined forces and torques are: forces and torques to solve all overlapping constraints, gradient-based forces for the balancing constraint, gradient-based forces and torques for the angle of inertia constraints and forces and torques to solve the functional constraints.

- **Additional operators:** The LHS and heuristic rule initialization as well as the swap operator are used.

- **Number of iterations:** As highlighted during the analysis of the single-container optimal layout problems experiments, the number of iterations should be adapted along with the occupation rate of the container. Consequently, the maximum number of iterations allocated for each containers is a function of its occupation rate determined by the assignation of the components to the container. The number of iterations varies linearly with the occupation rate and is calculated as follows:

$$N_{it}(OR) = 15000 \cdot OR \tag{4.40}$$

  where N_{it} is the number of iterations and OR is the occupation rate from 0 to 1.

- **Hyperparameters:** The hyperparameters are also set as functions of the occupation rates. Indeed, the hyperparameters have been optimized using Bayesian Optimization for the 30%, 40%, 50% and 60% occupation rates during the settings of CSO-VF for the single-container SMLP. For each container, one of the four lists of hyperparameters is selected according to its occupation rate.

The CSO-VF instances dedicated to each container are launched in parallel. Then, all the possible combinations of laid out containers are compared in terms of feasibility and objective function values and the combination leading to the smaller global inertia corresponds to the output of the algorithm. In the specific case where no feasible combination is found, another assignment list is proposed and the containers layout are optimized once again. In this study, the maximum number of attempts is set to 1 *i.e.*, the algorithm must find a solution in one attempt. Figure 4.31 illustrates the proposed algorithm.

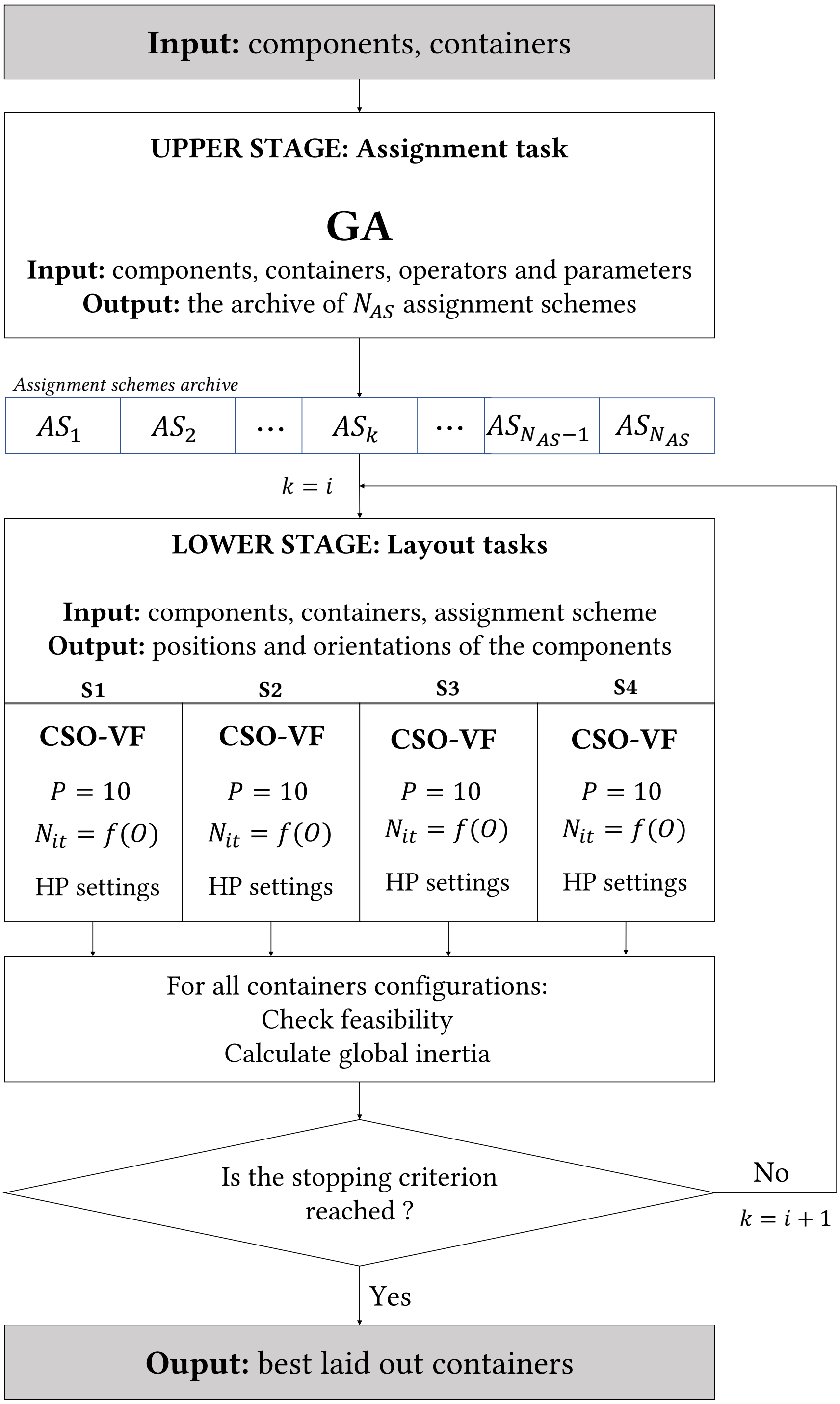


Figure 4.31: The GA assisted-CSO-VF algorithm for the multi-container SMLP.

4.3.3 Experimental results

The previous detailed algorithm is applied to the described multi-container SMLP and the results are compared to those published in:

- [Ten+09]: a dual-system cooperative co-evolutionary algorithm (called Oboe-CCEA) is developed for the multi-container SMLP based on both Potter's co-evolutionary framework [Pot+94] and the variable-grain model [ND02]. However, the assignment is an input of the algorithm *i.e.*, it is not optimized but taken from [ZTS08] and based upon human experience. It must be noted that this study does not consider the functional constraints $\mathbf{g}_H$ and $\mathbf{g}_E$. The rest of the mathematical formulation remains identical.

- [CZa19]: a two-stage algorithm called Dynamic FS is developed to solve both assignment and layout tasks of the multi-container SMLP. Heuristic rules are used to assign the components. They are mathematically translated into a multi-objective optimization problem and a NSGA-II algorithm is employed to solve it. The assignment list is then determined using a fuzzy decision making method. The layout task is performed using the NDCCDE/DPSO algorithm [CXW18] which corresponds a to dual-system cooperative co-evolutionary algorithm.

4.3.3.1 Analysis of the assignment algorithm

This section aims at analyzing the upper stage performance. Table 4.8 reports the total masses of each container for the fixed assignment list [Ten+09], the best assignment list obtained using the Heuristic+NSGA-II algorithm proposed in [CZa19] and the best assignment list obtained using the algorithm detailed in Section 4.3.2.1 of this manuscript. Table 4.9 reports the coordinate of the center of gravity along the z axis considering only the components or the whole module *i.e.*, the components in addition to the empty module, and for the three aforementioned assignment lists.

Reference	Surface 1	Surface 2	Surface 3	Surface 4
Fixed [Ten+09]	150.63	231.18	235.22	198.42
Heuristic+NSGA-II [CZa19]	88.33	314.20	317.29	95.63
GA (This manuscript)	85.95	289.02	326.67	113.78

Table 4.8: Masses distributions on surfaces.

z_{CG}	Fixed [Ten+09]	Heuristic + NSGA-II [CZa19]	This manuscript
Components	543.02	562.9	550.92
Whole module	547.38	559.03	552.01

Table 4.9: Center of gravity along z axis without and with the empty module. Coordinate along z axis of the center of gravity of the empty module: **553.56**.

For the three assignment schemes, it can be noticed that the surfaces 2 and 3 are always the heaviest. This contributes to minimize the global inertia along the z axis. Then, the bottom plate *i.e.*, surfaces S_3 and S_4, is more loaded than the top plate *i.e.*, surfaces S_1 and S_2. Indeed, the geometrical center between the two plates is situated at the coordinate of 565 along the z axis, but all the assignment schemes have centers of gravity located lower. The assignment scheme proposed in [CZa19] has the most balanced two plates. This is due to the fact that the heuristic assignment rules described in this paper impose that the two plates are both balanced (with a delta) in terms of mass and component area. This results in positioning the center of gravity of the components close to the geometrical center of the two plates *i.e.*, close to coordinate 565. However, as the two plates are not centered at the geometrical center of the empty module in the INTELSAT-III model, this results in pulling away the center of gravity of the components from the center of gravity of the empty module. Then, the assignment task resolution proposed in this thesis results in the positioning of the components on the two plates such that both center of gravity of the components and of the empty module coincide. This should provide a better global inertia in the end as well as more balanced satellite module.

Figure 4.32 shows the median and IQR of 50 independently initialized convergence curves of the upper level.

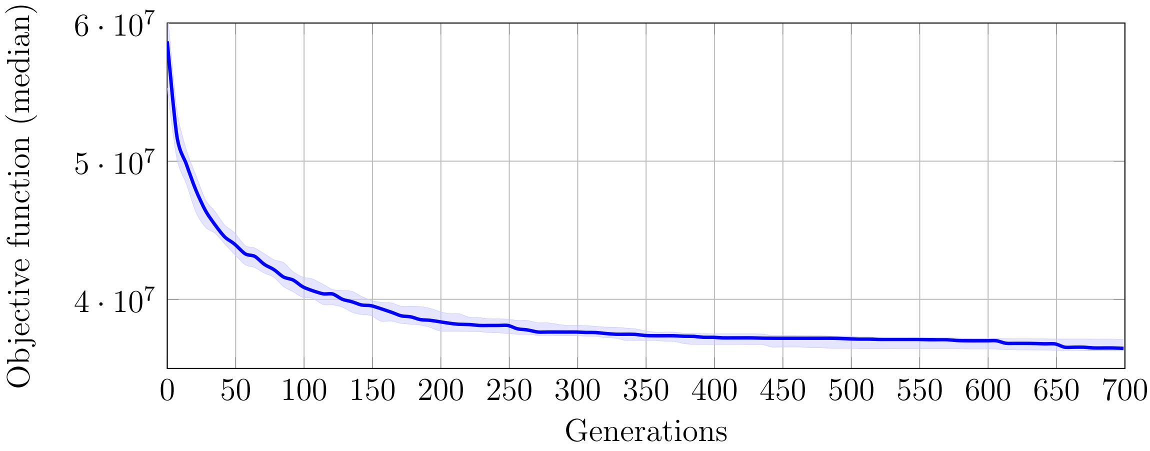


Figure 4.32: Median and IQR of convergence curves of the upper level for 50 independant runs.

It is notably observed that the IQR remains low during all the optimization process which characterizes a good robustness of the upper level with respect to the final interquartile range.

4.3.3.2 Analysis of the results of the two-stage algorithm

The algorithm detailed in Section 4.3.2 is run over 50 independently initialized instances. The obtained results are compared with those published in [CZa19] and [Ten+09] according the following indicators and reported in Table 4.11:

- The mean of the final obtained layout objective functions *i.e.* their global inertia;
- The standard deviation (STD) of the final obtained layout inertia;
- The best layout obtained in terms of inertia (among the 50 repetitions);

- The worst layout obtained in terms of inertia (among the 50 repetitions);
- The success rate *i.e.*, the percentage of runs leading to a feasible layout.

Table 4.10 reports the allocated budget in terms of number of objective function evaluations for the assignment and layout tasks and for the three compared algorithms. Figure 4.33 shows the best obtained layout using the Oboe-CCEA algorithm [CZa19] and the proposed algorithm. Figure 4.34 shows the best obtained layout using the Dynamic FS algorithm [CZa19] and the proposed algorithm. On both figures, the functionally incompatible components are highlighted in orange and blue for the heat constraints and orange for the electromagnetic constraint. Moreover, the redder the components, the heaviest their corresponding masses.

	Oboe-CCEA [Ten+09]	Dynamic FS [CZa19]	GA+CSO-VF
Assignment	x	1e6	1.75e5
Layout	1e6	1e6	2.7e5

Table 4.10: Allocated budget in terms of function evaluations for the assignment and layout tasks and the three compared algorithms. ”x” is written when the metric is irrelevant and ”-” is written when the data is unavailable.

Metrics	Oboe-CCEA [Ten+09]	Dynamic FS [CZa19]	GA+CSO-VF
Mean of final inertia	718.93	694.06	**682.96**
STD of final inertia	**2.64**	2.96	2.73
Best layout	712.99	689.00	**676.75**
Worst layout	726.59	700.37	**688.45**
Success rate	60%	80%	**100%**

Table 4.11: Numerical results for the multi-container SMLP obtained thanks to the three compared algorithms.

The proposed algorithm allows to improve the average final inertia by respectively 5% and 1.6% in comparison to the Oboe-CCEA algorithm with the fixed component assignment scheme based on human experience and to the Dynamic FS algorithm with optimized assignment schemes. It also improves the best obtained layout by respectively 5.08% and 1.78% in comparison with the two same algorithms. Moreover, it must be highlighted that the worst layout obtained using the GA assisted-CSO-VF algorithm remains lower *i.e.*, better than the best layouts obtained by both Oboe-CCEA and Dynamic FS algorithms. The success rate reaches 100% for the proposed algorithm against 80% for the Dynamic FS algorithm and 60% for the Oboe-CCEA algorithm. In other words, each of the 50 runs provides a feasible solution corresponding to one optimized assignment list of given by the upper stage of the proposed algorithm. However, the Oboe-CCEA has the smaller standard deviation which is increased by 12.1% by the Dynamic FS algorithm and 3.1% by the proposed algorithm.

Furthermore, Table 4.10 shows that the proposed algorithm has a lower computational budget in terms of number of objective function evaluations for both assignment and layout tasks than the two other algorithms. The routines are implemented in Python language, and when executed on a PC with an Intel Core i7, 16 GB of RAM and Windows operating system the upper level lasts 3 to 4 minutes and the lower level a dozen of minutes. Therefore, the computational time of one instance of the GA assisted-CSO-VF algorithm is approximately 15 minutes.

The proposed GA assisted-CSO-VF provides general better performance than the Oboe-CCEA and Dynamic FS algorithm regarding two aspects:

- The assignment scheme: The proposed assignment task resolution allows to position the components such that their center of gravity coincides with the center of gravity of the empty module which contributes to a better global inertia.

- The layout: As highlighted in the sections dedicated to single-container SMLP, the CSO-VF outperforms the population-based counterparts like GAs thanks to its ability to solve the constraints with dedicated operators. Indeed, Figure 4.34 shows that the CSO-VF algorithm systematically positions the heavier components closer to the center of the surfaces which contributes to minimize the global inertia of each container. On the contrary, it is observed that the Dynamic FS algorithm sometimes positions small and light components around the central bus.

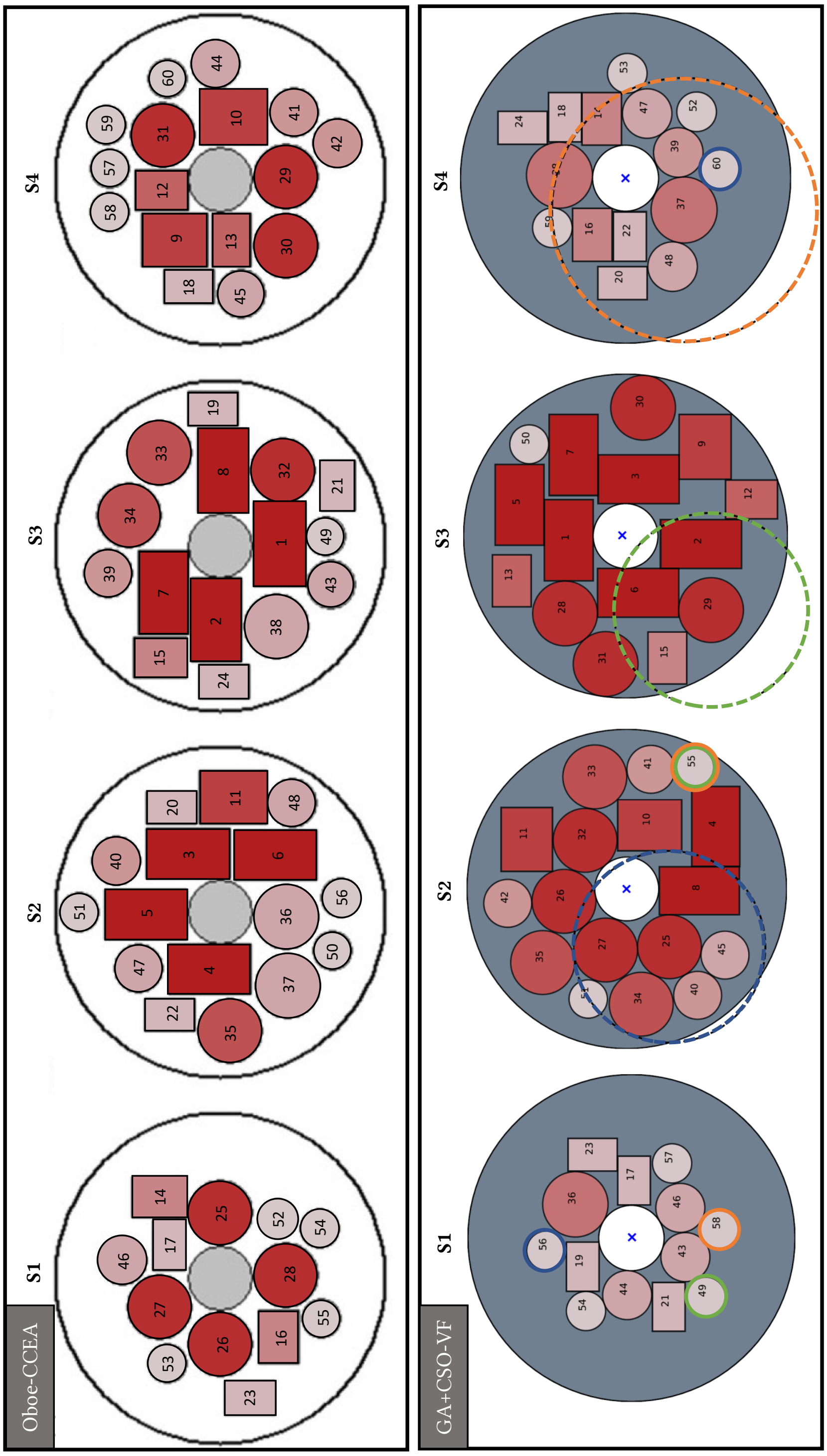


Figure 4.33: Best final layouts obtained with the Oboe-CCEA algorithm [Ten+09] and the proposed algorithm. The redder the components, the higher their corresponding masses.

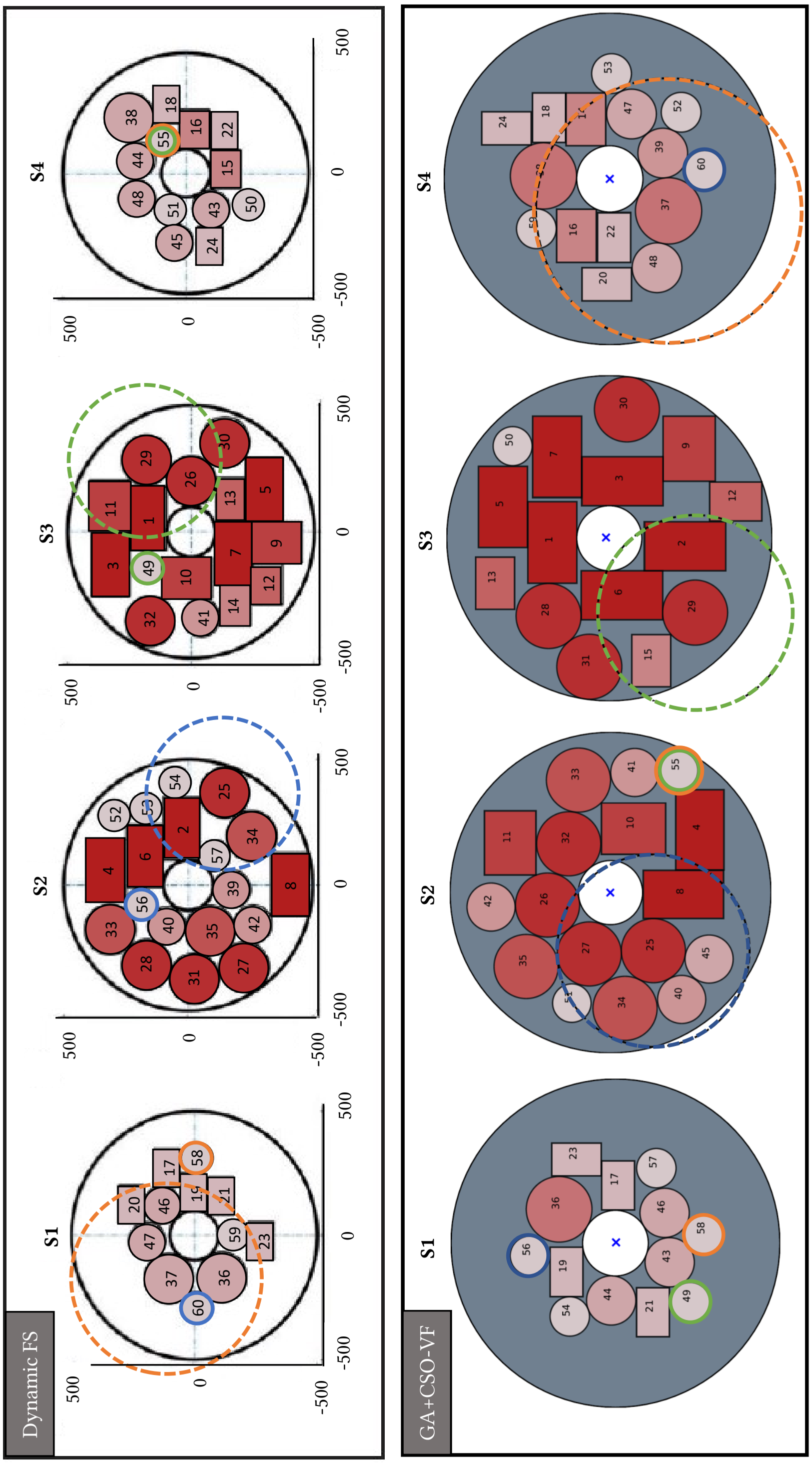


Figure 4.34: Best final layouts obtained with the Dynamic FS algorithm [CZa19] and the proposed algorithm. The redder the components, the higher their corresponding masses.

4.4 Advantages and limitations of the proposed algorithms

4.4.1 CSO-VF algorithm

The main advantages of the CSO-VF algorithm are summarized as follows:

- The constraints are handled in an inherent way thanks to dedicated operators like the virtual-force system, the swap operator and the initialization strategy.
- The CSO-VF algorithm performance is independent from the number of components *i.e.*, from the number of design variables, thanks to its inherent decentralized aspect.
- The CSO-VF algorithm only relies on the computation of the forces and torques which are simple mathematical operations and makes it compatible with GPU programming. Thus, it generally provides reasonable computing time.

Despite these strengths, some limitations are highlighted:

- The CSO-VF algorithm is a local search algorithm and locally improves the non convex objective function. Consequently, initialization of the layout has a critical impact on the final obtained layout and the convergence speed.
- The number of hyperparameters depends on the virtual-force system and especially on the number of gradient-based forces and torques stepsizes. Highly constrained problems can then lead to a high number of hyperparameters.
- The CSO-VF algorithm as described in Chapter 3 and Chapter 4 does not allow to tackle the multi-container configuration. On the one hand, it cannot deal with the assignment task. On the other hand, it cannot deal with constraints defined on several containers at the same time (*e.g.*, overlapping between components on surfaces facing each other, functional constraints defined in 3-dimensions, *etc.*). The virtual-force system should be extended to deal with these specificities.

4.4.2 Two-stage algorithm combining Genetic Algorithm and CSO-VF

The strengths of the two-stage algorithm are listed below:

- The two-stage (GA+CSO-VF) algorithm allows to tackle each of the assignment and layout tasks separately using dedicated methods. This facilitates the constraints solving as well as the exploration of the search spaces related to each task.
- As the upper stage provides an archive of assignment schemes, no sequential resolution is needed which tends to decrease the computational budget.

- The combination of the genetic algorithm and the CSO-VF algorithm allows to deal with a high number of components as both algorithms can deal with several hundreds of design variables.

Some limitations are nonetheless identified:

- The structure of the algorithm requires that the assignment and layout tasks can be addressed separately. This is not necessarily the case for all multi-container optimal layout problems.
- The upper stage also supposes that the assignment task can be mathematically formulated which might involve problem specific knowledge.
- In its current formulation, the algorithm is unable to deal with some constraints that involve several surfaces as their layouts are solved in independent optimization processes. For instance, overlapping constraints between components positioned on facing surfaces as well as functional constraints with 3-dimensional safety zones could not be handled.

4.5 Conclusions of the first part

This first part of the manuscript was devoted to fixed search space (FSS) optimal layout problems. In Chapter 2, a survey of existing methods for solving FSS optimal layout problems has been conducted. The advantages and limitations of the methods have been highlighted, which lead to particularly focus on quasi-physical approaches based on virtual-force systems. This range of methods seemed promising in order to address highly dimensional and highly constrained optimal layout problems. Virtual-force systems-based existing algorithms in the field of optimal layout problems were developed for solving circular packing problems. However, the formulation of such problems did not encompass all the geometrical configurations, all the constraints and all the objective and constraints functions that are considered in this thesis.

Consequently, Chapter 3 focused on formulating the optimal layout problems at hand and detailed a quasi-physical approach based on a virtual-force system. The Component Swarm Optimization algorithm based on a Virtual-Force system is designed for solving single-container optimal layout problems. It allows to deal with all shape of components and containers with the introduction of torques in its virtual-force system. It permits to deal with any constraint or objective functions differentiable with respect to the design variables using gradient-based forces and also allows to deal with the balancing constraints in the case were exclusion zones belong to the container and that no relaxation of this constraint is possible. A swap operator has been introduced which contributes to provide first feasible solutions in less iterations and to propose new configurations of the layout in case of stagnation of the convergence for example due to antagonist forces.

Then, the single-container optimal layout problems statement has been extended to multi-container configurations. Such problems require to assign each component to a container while optimizing the layout of each container and thus have a high combinatorial design space in addition to the high dimensional and constrained characteristics. A two-stage GA assisted-CSO-VF algorithm has been proposed for solving the assignement and layout tasks using two dedicated algorithms. The upper stage is a discrete Genetic Algorithm solving the assignation task and the CSO-VF algorithm is employed to solve the layout of each container in a lower stage.

Finally, Chapter 4 proposed application cases to evaluate the performance of the algorithms described in the previous chapter. Satellite Module Layout Problems were formulated as representative benchmarks of optimal layout problems. A single-container derived from the multi-container satellite module layout problem has been formulated and used for analyzing both the configuration of the CSO-VF algorithm and its global performance in comparison with GAs and a CMA-ES algorithm and for several occupation rates of the container. As a result, the CSO-VF algorithm provides better performance in terms of success rate, convergence accuracy, robustness, convergence speed and ability to solve the constraints in comparison to the GA and CMA-ES algorithm counterparts. Indeed, the dedicated operators for handling each of the constraints provided by the virtual-force system and the swap operator allow the CSO-VF algorithm to outperform population-based counterparts for the same number of evaluations of the objective function. It must also be noted that the CSO-VF algorithm relies on simple operations *i.e.*, the forces equations which is compatible with GPU programming and thus provides reasonable computational times.

The multi-container SMLP has been detailed in order to evaluate the performance of the two-stage GA assisted-CSO-VF algorithm in comparison with previous published results. The GA upper stage responsible for the assignation task has been configured using an analysis of the objective function equations. This same analysis allowed to identify some deficiencies in the present inertia equations which should be corrected in future experiments. The CSO-VF was employed as a lower stage to solve the layout of the components within the four assigned surfaces. The proposed algorithm outperforms the previously published results in terms of success rate, convergence accuracy and best layout obtained in less objective function evaluations. The analysis of the equations and the aforementioned strengths of the CSO-VF algorithm allowed to find both satisfying assignment scheme and layouts.

Finally, the CSO-VF algorithm has also been adapted and applied to balanced circular bin packing problems [He+13]. In Appendix E, the algorithm, its settings and the assessment of its performance using balanced circular bin packing benchmarks are detailed.

In this part of the manuscript, only optimal layout problems with fixed search space have been addressed. However, as described in the Introduction of the manuscript, optimal layout problems can more generally involve architectural choices that must be defined together with the optimal layout process. This can be translated by conditional search space optimal layout problems which have an additional task corresponding to the choice of the list of the components to integrate to the system in addition to the optimization of its layout. Thus, in the following part of the thesis, the fixed search space problems considered in this first part are extended to conditional search Space optimal layout problems.

Part II

Conditional Search Space Optimal Layout Problems

Chapter 5

Conditional Search Space Problems Statements and Related Works

Contents

Chapter goals

- Mathematical formulation of conditional search space problems.
- Extension of the mathematical formulation to generic conditional search space optimal layout problems.
- Survey of the existing approaches to handle the conditional search space problems.

5.1 Introduction

In the first part of the thesis, fixed search space optimal layout problems (OLPs) have been addressed. A fixed set of components was defined such that the number and type of design variables and constraint functions remained constant throughout the optimization process.

However, as mentioned in the introduction of this manuscript, some architecture choices cannot be made beforehand and must be defined together with the optimal layout process. For instance, considering the optimal layout of wind farms which consists in maximizing the power produced by wind turbines positioned in the farm as well as wake effect limitation [Sam13], the number of wind turbines has a strong influence on the power produced as well as on the wake effect and can thus be considered as a variable in the optimization process [Gon+11; Rye+17].

Considering a variable number of items to position makes the number and type of design variables vary from one subproblem to another, as well as the number and type of constraints. The choices related to the architecture definitions (*i.e.*, the choice of the components to be integrated to the system) can be represented with categorical or qualitative variables in the optimization problem formulation. These variables are named conditional variables [Swe+14] as they influence the number and type of continuous and discrete variables defining the layout, as well as the number and type of constraint functions. They are also called metavariables [Rye+17], dimensional variables [Pel+21], *etc.* Therefore, the resulting problems are referred to as conditional search space problems [LPS05]. They are also sometimes referred to as variable-size design space problems [NAO15; Pel+21].

The most naive approach to this type of problems could be to optimize the subproblems for each of the possible architecture definitions [Fra+18]. Considering the wind farm optimal layout problem, this approach would consist in optimizing the layout of the wind farms for all possible numbers of wind turbines and choose the best one with relation to their power production. However, depending on the complexity of the system at hand, several thousands of architecture definitions can sometimes be considered and this approach may be very time consuming and in some cases unfeasible. Consequently, the layout of the system must be optimized simultaneously with some of the design variables which define it. Allowing the number of components to vary during the optimization process may provide a better solution in the end.

Conditional search space (CSS) problems belong to the NP-hard class of problems and are therefore not solvable using classical optimization methods [WTS09b]. Dealing with CSS problems arises additional challenges notably because of the introduction of conditional variables and the ensuing conditional complexity (*i.e.*, the number of subproblems linked to the architecture choices defined by the conditional variables). Considering such variables requires to extend the approaches presented in the previous part.

The objectives of this chapter are to present the mathematical definition of CSS problems, to specify the formulation in the case of OLPs, to raise and discuss the issues related to both defined problems as well as to provide reviews of the existing techniques developed to handle them. In the first section of this chapter, CSS problems are mathematically formulated and specified in the case of OLPs. Then, the existing methods adapted to handle CSS problems are reviewed in the second section. These methods are then summarized and discussed in the two last sections of this chapter which allows to highlight the need for the methods developed in this second part of the manuscript.

It must be noted that, in the literature, very few OLPs integrate a conditional search space [Gon+11; Rye+17]. Thus, in this chapter, all application cases are considered in order to review the existing methods for solving CSS problems which could possibly be used in case of CSS OLPs.

5.2 Conditional search space problems formulation

The formulation of fixed search space (FSS) problems formulation presented in Chapter 1 is extended in order to add the conditional search space (CSS) aspect. In this section, CSS problems are defined and specified in the case of OLPs.

5.2.1 Generic mathematical formulation

A generic CSS optimization problem can be defined using three types of design variables, listed and detailed as follows:

- **Continuous variables: $\mathbf{x}$**
  As defined for FSS problems, continuous variables correspond to real numbers defined within given intervals.

- **Discrete variables: $\mathbf{z}$**
  As for FSS problems, discrete variables can be quantitative *i.e.* integers defined within given intervals, or qualitative *i.e.* non-relaxable variables defined within a given set of choices.

- **Conditional variables: $\mathbf{w}$**
  These variables are non-relaxable variables defined among a finite set of possibilities. According to their values, the number and type of continuous and discrete variables that have to be optimized in the problem may vary, the considered constraints as well. These variables, specific to the formulation of conditional search space problems, often correspond to different architectural choices. According to the number of conditional variables as well as their values, the number of FSS subproblems of the problem corresponding to each of the architecture definition varies. Each subproblem is defined by a set of continuous and discrete variables to optimize as well as constraint functions. Thus, the number and values of the design variables as well as the constraints vary during the optimization process. Those variables are also referred to as meta variables, as in [AHD23] in which a general mathematical framework to integrate meta and categorical variables is proposed. They can also be referred to as dimensional variables as in [Pel+21]. Finally, the term hierarchical variables can also be met [Bus+21].

As a result, a conditional variable vector $\mathbf{w}$ has a dimension of n_w while continuous and discrete variable vectors $\mathbf{x}$ and $\mathbf{z}$ have respectively dimensions of $n_x(\mathbf{w})$ and $n_z(\mathbf{w})$ depending on the values taken by the conditional variables. Therefore, continuous and discrete variables will sometimes be referred to as *depending variables* [AHD23]. Thus, similarly to what is proposed in [LP04; LPS05; Pel+21], a generic conditional search space problem can be formulated as follows:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{w}} \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}) \\
\text{where} \quad & \mathbf{x} \in F_x(\mathbf{w}) \subseteq \mathbb{R}^{n_x(\mathbf{w})}, \mathbf{z} \in F_z(\mathbf{w}) \subseteq \mathbb{N}^{n_z(\mathbf{w})}, \mathbf{w} \in F_w \\
\text{s.t.} \quad & \mathbf{h}(\mathbf{x},\mathbf{z},\mathbf{w}) = 0 \\
& \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0
\end{aligned} \tag{5.1}$$

where f is the objective function, and $\mathbf{h}$ and $\mathbf{g}$ correspond respectively to equality and inequality constraints. They take as arguments: continuous variables $\mathbf{x}$ which search space is named $F_x(\mathbf{w})$, discrete variables $\mathbf{z}$ which search space is $F_z(\mathbf{w})$ and conditional variables $\mathbf{w}$ which search space corresponds to F_w, as described previously.

5.2.2 Conditional search space optimal layout problems

Considering conditional search space (CSS) OLPs as representative CSS problems, the general mathematical formulation given in the previous section is specified in this case. In most of the OLPs tackled in the literature, the set of components to position in a container is fixed at the beginning of the optimization process. They correspond to fixed search space (FSS) problems described in Chapter 1.

In this thesis, in the case of CSS OLPs, a catalogue of components is defined at the beginning of the optimization process and thus, the main concern is to optimize the list of the components among the given catalogue as well as their layout in the container. Indeed, some design requirements can be met using several sets of components. For instance, a same amount of fuel can be loaded to a vehicle using one large tank of fuel but also two smaller ones. Another example would be to integrate a required battery power using different batteries with varying power.

Design Variables. The design variables of a CSS optimal layout problem are listed as follows:

- Continuous variables $\mathbf{x}$: As for FSS problems considered in the first part of the thesis, continuous variables can correspond to the centers of inertia and angular orientations of the components.

- Discrete variables $\mathbf{z}$: As for FSS problems considered in the first part of the thesis, discrete variables can correspond to one or several coordinates of the centers of inertia of the components as well as their orientations depending on the problem specifications [ST03]. They can also correspond to the geometry of the items (round, square, rectangle). In case of multi-container problems, discrete variables can also correspond to the the container index in which each component is positioned [CZa19].

- Conditional variables $\mathbf{w}$: The conditional variables stand for the choice of components to be included within the system at hand. When all items are similar, the conditional variables can merely correspond to the number of components to be included to the system. For instance, in the case of CSS wind farms OLPs, the wind turbines are identical. Then, only one conditional variable taking the value of the number of turbines can be used [Gon+11]. However, in case of various shapes, sizes and types of components, several conditional variables are needed to express the choice of components. In this thesis, conditional variables correspond to the different possible subdivisions of each family of components that are listed in the catalogue of components. For example, in the case of an aircraft design, a requirement might be to include a certain amount of fuel to the system. A family of components "Tanks of fuel" is thus needed and a catalogue with different subdivisions of tanks is given allowing to include this amount of fuel with several different components that can be chosen. The same goes for the battery. Figure 5.1 gives an example of the possible values of conditional variables for different numbers of hydrogen tanks of fuel integrated within an aircraft [Kha+13].

  Depending on the value of $\mathbf{w}$, the number of components to position in the container varies and consequently the number of continuous and discrete variables characterizing the layout vary as well.

If N components must be positioned in the container and each of them has n_i possible subdivisions ($i \in \{1, N\}$), then the number of possible architecture configurations (*i.e.*, the number of different possible lists of components) corresponds

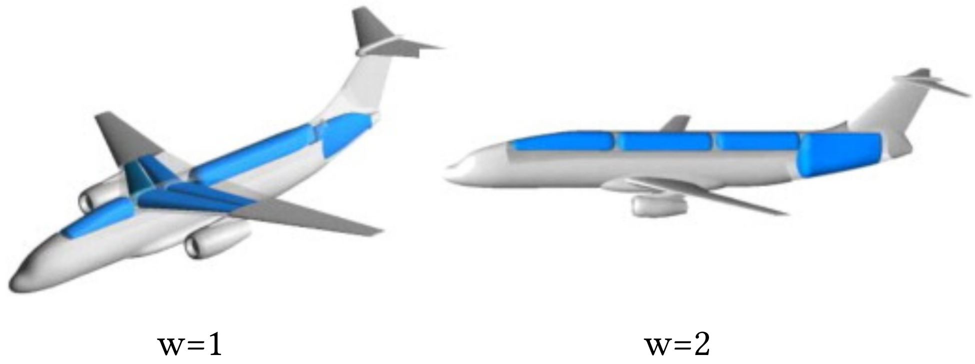


Figure 5.1: Example of conditional variables for two possible subdivisions of hydrogen tanks of fuel (7 tanks or 4 tanks) positioned within an aircraft [Kha+13].

to the cardinality of the set of achievable values of $\mathbf{w}$, F_w, and is mathematically defined as:

$$\mathrm{Card}(F_w) = \prod_{i=1}^{N} n_i \tag{5.2}$$

Objective Function and Constraints. The possible objective and constraint functions remain similar to those detailed in the case of CSS OLPs in Chapter 1.

The objective function to minimize can correspond to: the occupied volume of the components within the system, mechanical characteristics like for instance the global inertia of the system or the position of the center of gravity, any performance of the system as for example the power production, any costs related to the layout of the system, *etc.*

The possible constraint functions are also briefly summarized:

- **Geometrical constraints:** For instance, no overlapping between the components, no overlapping between the components and the container, balancing constraints (*i.e.*, the global center of mass must be positioned at the geometrical center of the container), *etc.*

- **Functional constraints:** For instance, proximity or minimal distance requirements between components for functional exigencies. This can be radiative threshold (*e.g.*, electromagnetic compatibility).

The number and type of constraints depend on the conditional variables $\mathbf{w}$. For instance, the number of overlapping constraints depend on the number of components chosen thanks to the conditional variables. In the same way, the number and type of functional constraints depend on the number and type of functional components integrated to the system.

5.3 Review of methods for conditional search space problems

As mentioned in Section 5.1, the aforementioned formulation of CSS optimization problems has only been applied very rarely to OLPs in the literature. Therefore, this section reviews and details the different existing approaches for CSS optimization problems in general, independently from their experimental case studies.

5.3.1 Exact exploration of the conditional search space

The first existing approach consists in optimizing independently the subproblems of the CSS problem at hand which correspond to each combination of the conditional variables. Therefore, the optimization is performed over the relevant depending continuous and discrete variables for every FSS subproblems defined by a unique combination of the fixed conditional variables. This approach allows to fully explore the conditional search space and can theoretically provide a consistent convergence towards the global optimum of the CSS problem in case of robust convergence towards global optimum of all FSS subproblems. Using the previous formalism, this approach can be mathematically formulated as follows [AHD23; Pel20]:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}_s) \quad \forall\ \mathbf{w}_s \in F_w \\ \text{where} \quad & \mathbf{x} \in F_x(\mathbf{w}_s), \mathbf{z} \in F_z(\mathbf{w}_s) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x},\mathbf{z},\mathbf{w}_s) = 0 \\ & \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}_s) \leq 0 \end{aligned} \tag{5.3}$$

An example of the approach is given in [Fra+18]. Frank *et al.* developed a methodology in order to explore large multi-architecture design space in the context of optimal design of aerospace vehicles. A systematic process was developed in order to generate architectures. Each of these architectures is subsequently optimized using a metaheuristic multi-objective optimization algorithm. Finally, the Pareto fronts obtained for each architecture are merged in order to perform a global analysis of their performance.

It must be noted that the ability of the approach to deal with mixed-variable problems and to handle the constraints depends on the chosen algorithm to optimize the subproblems. Nonetheless, even if this approach allows to fully explore the relevant search space, it can become inefficient or even unfeasible for complex CSS problems characterized either by a large number of combinations of conditional variables or by computationally expensive functions.

5.3.2 Direct search methods for conditional search space

In [AHD23], Audet *et al.* defined a generic mathematical framework to both formalize the subproblems definition and the exploration of such subproblems based on direct search techniques [AAD07; AD01; AD06; KAD01]. The framework used corresponds to Pattern Search (PS) algorithms, also referred to as mesh-based algorithms. PS algorithms are direct search methods that do not require any gradient information of the objective and constraints functions and can be thus applied to discontinuous or non-differentiable functions [Tor97]. In PS algorithms, the objective function is evaluated at a finite number of points on a mesh at any given iteration, in order to find one that yields a decrease in the objective function value. Each iteration begins with the incumbent solution point and the construction of a mesh which is explored in two sequential phases. First, a user-defined search phase explores many points and aims at improving the incumbent. If it fails, the poll phase is called corresponding to a potentially exhaustive local search of the mesh around the incumbent solution point. PS algorithms have been extended for mixed-variables problems and used in [Abr+09; AAD07; AD01; KAD01; LVM05].

In [AHD23], a new user-defined neighborhood is mathematically formulated for exploring the conditional search space. Indeed, conditional variables are often defined as qualitative discrete (*i.e.*, categorical variables), which do not possess intuitive neighborhoods nor directions of exploration. The user-defined neighborhood is then used in a direct search framework. The authors also proposed to solve each FSS subproblem with the MADS (Mesh Adaptive Direct Search) algorithm [AD06; ADT19] as it enables to treat simultaneously discrete and continuous variables with the constraint-handling technique proposed in [AD09]. It must be noted that any technique allowing to tackle FSS mixed-variables can be employed for solving the FSS subproblems of the global CSS problem.

Even if the mathematical framework proposed in [AHD23] formally models the conditional search space and its exploration, no computational experiments has yet been carried out with this mathematical framework [AHD23].

5.3.3 Extension of metaheuristic methods

In order to deal with conditional search space (CSS) problems, metaheuristic methods have been extended in order to take advantage from their global exploration abilities as well as the variety of generic frameworks that can be used.

5.3.3.1 Particle Swarm Optimization

In [EMS09], Escalante *et al.* used PSO with the original particle implementation to solve the Full Model Solving (FMS) problem. FMS consists in optimizing a combination of processing methods, one feature selection and one learning method as well as their hyperparameters in order to solve classification tasks. This problem is thus a CSS problem as the number and type of hyperparameters depend on the selected algorithms. The authors implemented the FMS with a classical particle mixed-variable representation. Variables are needed for:

- The choice of a combination of preprocessing method(s) and the corresponding hyperparameters settings;
- The choice of the feature selection algorithm and the corresponding hyperparameters settings;
- The choice of the classification algorithm and the corresponding hyperparameters settings.

The full-models are built using *The Challenge Learning Object Package* (CLOP) which is a machine learning toolbox [SG06]. Figure 5.2 sums up the different possible objects in the CLOP environment for each component (preprocessing, feature selection and learning algorithms) of the full-models.

Preprocessing objects

	Object name	Hyperparameters
1	Normalization	Center
2	Standardization	Center
3	Shift-scale	$take_{log}$

Feature selection objects

	Object name	Hyperparameters
1	Ftest	$f_{max}, w_{min}, p_{val}, fdr_{max}$
2	Ttest	$f_{max}, w_{min}, p_{val}, fdr_{max}$
3	aucfs	$f_{max}, w_{min}, p_{val}, fdr_{max}$
4	odds-ratio	$f_{max}, w_{min}, p_{val}, fdr_{max}$
5	relief	$f_{max}, w_{min}, k_{num}$
6	rffs	f_{max}, w_{min}
7	svcrfe	f_{max}
8	Pearson	$f_{max}, w_{min}, p_{val}, fdr_{max}$
9	Zfilter	f_{max}, w_{min}
10	gs	f_{max}
11	s2n	f_{max}, w_{min}
12	pc-extract	f_{max}

Learning objects

	Object name	Hyperparameters
1	zarbi	none
2	naive	none
3	klogistic	none
4	gkridke	none
5	logitboost	Units number, shrinkage,depth
6	neural	Units number, shrinkage, maxiter, balance
7	svc	Shrinkage, kernel parameters (coef0,degree,gamma)
8	kridge	Shrinkage, kernel parameters (coef0,degree,gamma
9	rf	Units number, balance, mtry
10	lssvm	Shrinkage, kernel parameters (coef0,degree,gamma)

Figure 5.2: Preprocessing, feature selection and learning objects available in the CLOP environment as well as their hyperparameters.

Then, for the FMS problem tackled in [EMS09] each particle is composed of 16 variables both continuous or discrete. To be evaluated, the particles are then decoded using the CLOP toolbox. If an object has less hyperparameters than the maximum number of possible hyperameters for its type of objects, then only a part of the hyperparameters variables are used to build the full-model.

Thus, this formulation allows to use classical metaheuristic frameworks as well as classical evolution operators. However, the depending variables corresponding to the hyperparameters of each component of the full-model are here grouped under shared variables. Indeed, for each component of the full-model (preprocessing, feature selection and learning algorithms), the number of variables dedicated to the hyperparameters settings corresponds to the maximum number of hyperparameters of such component. Therefore, if the hyperparameters of several algorithms share design variables but are of different types and take values from different ranges of the continuous search space, such an implementation will lead to suboptimal solutions as well as very low convergence rates. Moreover, this implementation is not always extendible to other application cases especially in case of complex conditional search space where the depending variables are very different from one subproblem to the other in terms of number, types and values.

To sum up, as the choice of conditional variables result in various numbers of depending continuous and discrete design variables, classical linear variable vector implementations can often not be used directly with metaheuristic frameworks and need to be adapted as well as the classical evolution operators. The following sections are dedicated to metaheuritics in which the encodings of the individuals of the population have been modified to allow a conditional search space.

5.3.3.2 Variable-length metaheuristic algorithms

In this approach, all the variables (sometimes referred to as metavariables) of the genotype are used in the phenotype *i.e.*, all the variables of each individual are used to compute the objective and constraint functions. Saraf *et al.* reviewed various metaheuristic algorithms with variable-length searching [SFT23]. Variable-length representations have mostly been included into GA frameworks [CML19; DKB18; LBE18] as well as PSO frameworks [KŠ18; Moh+21; TXZ18]. For the sake of conciseness, only variable-length GAs are detailed in this section. In [Rye+17], Ryerkerk *et al.* introduced metameric GA using a variable-length representation and adapted GA operators to deal with this representation. This variable-length approach is applied to 3 application cases: sensor coverage, wind farm layout and laminate stacking. Even if selection and replacement operators can remain the same as for usual GA chromosomes representation, crossover and mutation operators must be adapted.

In the following sections, in order to illustrate the evolutionary operators, the i^{th} gene of the j^{th} parent chromosome is denoted p_i^j. The children obtained from the parents chromosomes correspond to recombination and mutation of the aforementioned parents genes.

Crossover. First of all, the most used variable-length crossover operator is the cut-and-splice crossover which is a n-point crossover operator in which the crossover points do not have to match between the two parents. As a result, the parents and the offspring do not have necessarily the same length. Figure 5.3 illustrates the cut-and-splice crossover operator for two parents chromosomes with the same length ($i.e$, 7 variables p_i^j, $j \in \{1, 2\}$ and $i \in \{1, ..., 7\}$) and with fully independent variables.

Some additional challenges may arise when the design variables are not independent. For instance, in the case of OLPs, a component is located by its coordinates corresponding to two continuous design variables. Then, those two variables must be recombined together from parents to children. Consequently, crossover points must be chosen carefully. Other crossover operators for variable-length chromosomes have been developed like the spatial recombination [CP06] and the synapsing variable-length crossover [HW07].

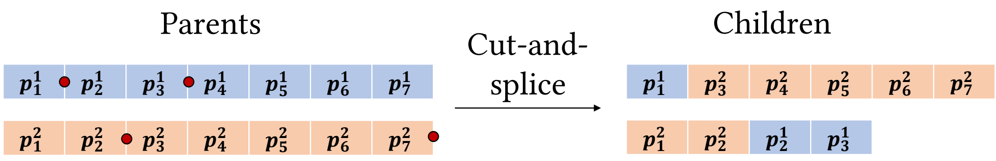


Figure 5.3: Cut-and-splice recombination operator on an example, for independent genes.

Mutation. Different kinds of mutations were introduced in addition to classical mutation operators occurring on the metavariables. For instance, the insertion mutator involves inserting a new sequence of genetic material into the genome at a random point while the deletion mutator removes a section of the genetic material at a random point [BM05]. Figure 5.4 illustrates the insertion and deletion mutators for the same parents chromosomes introduced in the previous paragraph: two parents with seven independent genes are considered.

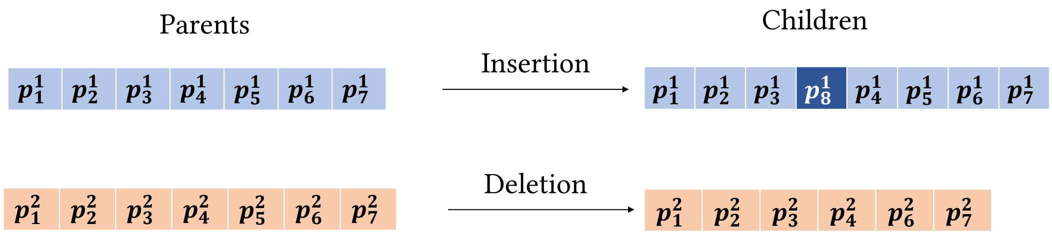


Figure 5.4: Insertion and deletion mutation operators for variable-length chromosomes representation on an example, for independent genes.

As for the variable-length crossover operators, difficulties arise when performing those mutations in order to preserve a coherent structure of the variable chromosomes (*i.e.* that the sequence of variables in a chromosome always corresponds to a combination of conditional variables). Other mutation operators exist like the swap or the parameter mutations [BM05].

The ability of this algorithm to handle the mixed-variables aspect and the constraints depends on the chosen metaheuristic algorithm as well as the chosen operators and constraint handling techniques.

5.3.3.3 Dynamic-Size Multiple Populations Genetic Algorithms

In [AG12], Abdelkhalik *et al.* described a Dynamic-Size Multiple Populations GA (DSMPGA) which follows a decomposition principle: initial subpopulations corresponding to sub-problems are generated. Thus, the members in each subpopulation will have the same chromosome length while different subpopulations will have different chromosome lengths. Subsequently, classical GA operators are applied to all subpopulations in parallel during a certain number of generations called a stage of generations. The individuals are then evaluated by the end of each stage and a selection operator is applied independently from the length of the individuals. As a result, the size of subpopulations with more interesting solutions increases while the size of less interesting solutions subpopulations decreases. Figure 5.5 illustrates the DSMPGA algorithm on an example with 4 initial subpopulations and 3 stages of generations. On the figure, for clarity purpose, the subpopulations are denoted with the letters A to D and are highlighted in blue, orange, yellow and green. In each subpopulation, the number of letters A to D corresponds to the size of the population. Thus, at the initial stage, all four subpopulations are the same size (*i.e.*, 5 individuals on the example). Subsequently, the sizes of each subpopulation vary throughout the stages of generations. Moreover, each subpopulation is assigned a gray level which characterizes the fitness value of the best individual in the subpopulation. The darker is the gray level, the higher (*i.e.*, the better) is the best value of the fitness in the subpopulation.

This DSMPGA algorithm has been successfully applied to the optimization of trajectory of interplanetary missions where conditional variables correspond to the number of swing-bys and the planets to swing by resulting to appearance and disappearance of variables describing the orbital maneuvers. 140 subpopulations were necessary to solve the problem resulting in 2800 individuals in the whole population. As the number of initial subpopulations depends on the number of combinations of conditional variables, this method may not be suitable for problems characterized by a large number of combinations of conditional variables as well as computationally expensive functions.

5.3.3.4 Structured Genetic Algorithm

Nyew *et al.* [NAO15] proposed a Structured GA with a more complex but theoretically more efficient chromosome implementation than previous adaptations. In the Structured GA, multilayer chromosomes are implemented depending on a hierarchical representation rather than a linear one. In this framework, the genes in the chromosomes are classified with dependency rules. Certain variables exist in the depending space only if some other variables (the conditional variables) take specific values. These dependencies are translated into the hierarchical representation. Figure 5.6 shows the multilayer chromosome representation [NAO15]. The same formalism as in Section 5.3.3.2 is employed to denote the genes of the chromosome. On Figure 5.6, a standard representation of a chromosome is shown, where genes are stacked together in one string. The genes are not linked and thus, the number of genes is independent from the value of any gene in the chromosome. The multilayer chromosome structure is also shown, where the genes are organized in different layers. In this representation, the value of gene p_2^1 determines how many genes exist in the second layer (two genes p_4^1 and p_5^1).

Classical crossover and mutation operators cannot be applied in case of a Structured GA due to the new representation of the chromosomes and are detailed in the following.

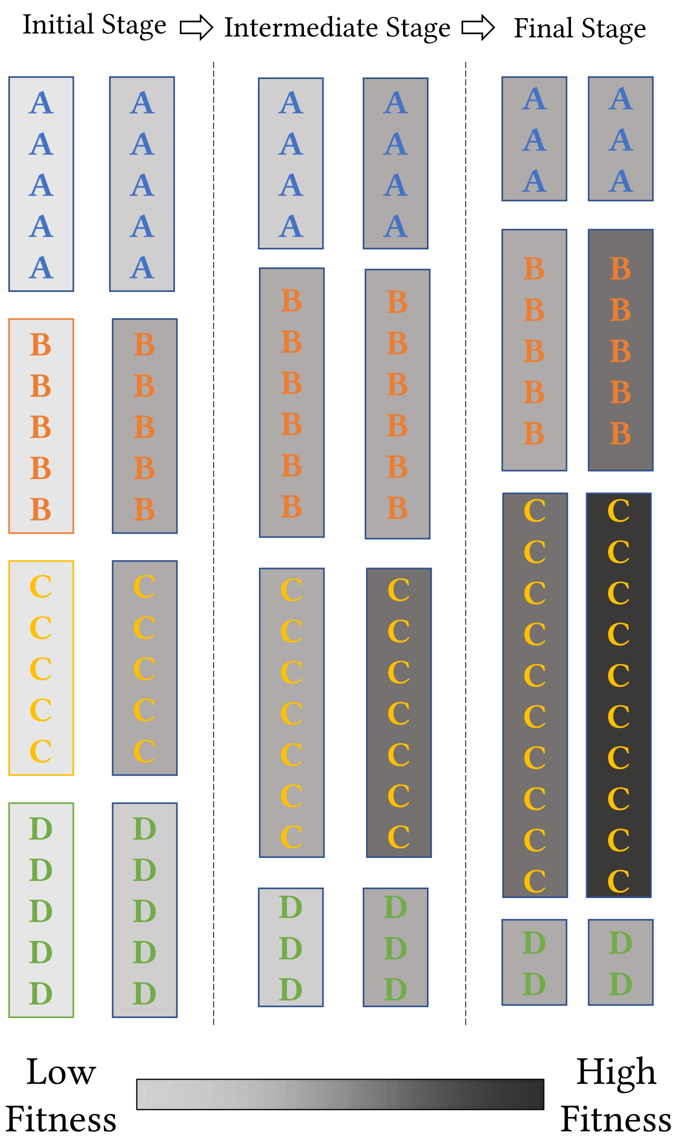


Figure 5.5: Illustration of the Dynamic-Size Multiple Populations GA on an example [AG12].

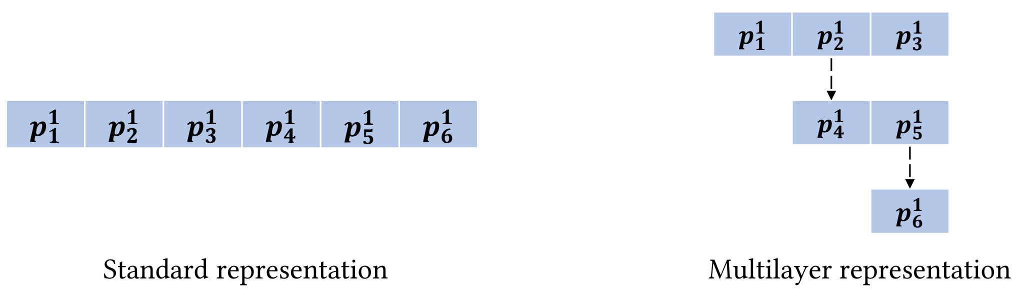


Figure 5.6: Structured GA multilayer chromosome representation compared to the standard representation [NAO15].

Crossover. In [NAO15], the proposed crossover operator exchanges genes which are of identical types. Indeed, the recombination operators must guarantee a semantically correct crossover and provide meaningful children. Figure 5.7 illustrates the crossover operator. At the initial stage, the first parent chromosome contains 6 genes from ν_1 to ν_6 to and the second parent chromosome contains 4 genes ν'_1 to ν'_4. The genes that have the same letter have the same type. A certain number of exchanges, for instance 3, then occurs between the genes of the same type of the two parents genetic material, layer from layer. In case of exchange between genes having dependent variables, all the dependent variables of the genes are swapped. They are still candidate for other exchanges when it will be their layer's turn of exchanges.

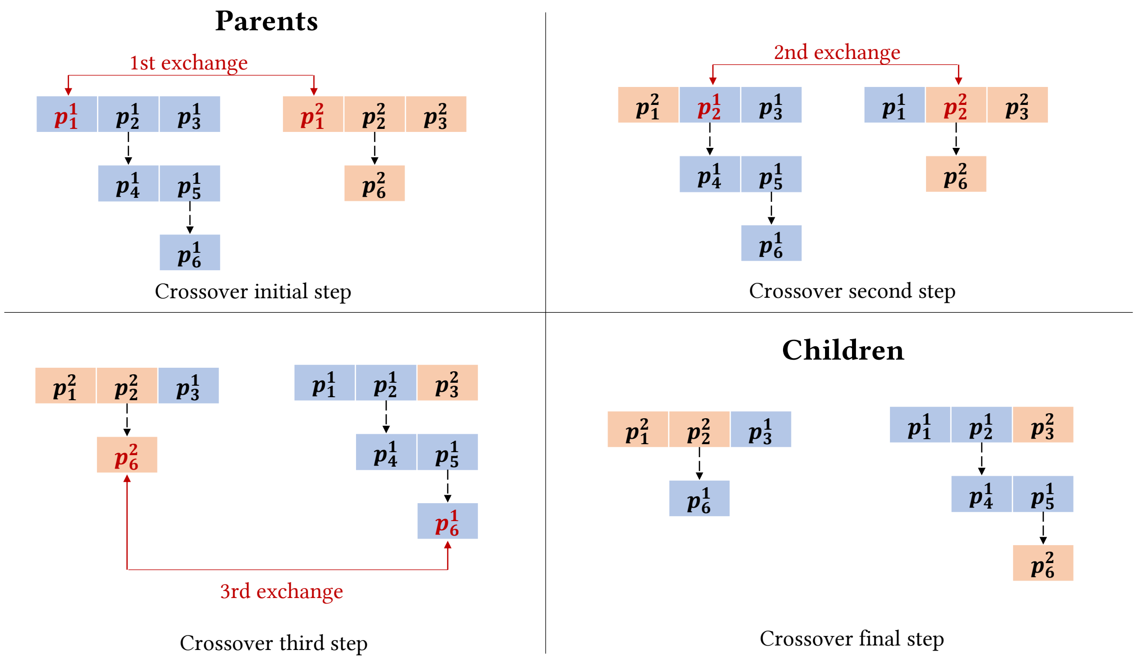


Figure 5.7: Crossover operator for Structured GA on an example [NAO15].

Mutation. The mutation operation allows a gene to take a new value. Each gene has a predefined mutation probability and when a gene is selected for mutation, the algorithm makes sure that the randomly generated new value is consistent with the gene's type (*e.g.*, continuous, discrete or categorical). The generated value must fall between the lower and upper bounds provided for the gene. The genes are mutated from the highest layer to the lowest layers. Therefore, if the mutated gene has dependent genes, then the dependent genes are created or deleted in a way that ensures semantic correctness.

Transformation for Structured DE. In [NAO15], the authors also developed a Structured DE algorithm and defined a new transformation operator encompassing crossover and mutation operations. Transformation starts by selecting a target chromosome $\mathbf{p_{target}} = \{p_j^{target}\}$, for $j \in \{1, ..., N_{DV}^{target}\}$ and three other chromosomes ($\mathbf{p_1} = \{p_j^1\}$, for $j \in \{1, ..., N_{DV}^1\}$, $\mathbf{p_2} = \{p_j^2\}$, for $j \in \{1, ..., N_{DV}^2\}$ and $\mathbf{p_3} = \{p_j^3\}$, for $j \in \{1, ..., N_{DV}^3\}$), and it uses two steps to generate a child. The first step is to create a temporary chromosome $\mathbf{p_{temp}} = \{p_j^{temp}\}$, for $j \in \{1, ..., N_{DV}^{temp}\}$ as follows:

$$\mathbf{p_{temp}} = \mathbf{p_1} + \gamma(\mathbf{p_2} - \mathbf{p_3}) \tag{5.4}$$

where γ is the differential weight that adjusts the influence of $(\mathbf{p_2} - \mathbf{p_3})$ in the new chromosome. The final step is to generate the offspring by crossover from the target chromosome $\mathbf{p_{target}}$ and the newly generated chromosome $\mathbf{p_{temp}}$ [NAO15].

The algorithm was efficiently applied to the optimization of trajectories of interplanetary missions and in [Gen+19], the same implementation was used to solve satellite tracking problems.

Such a method proved to be efficient to represent and implement complex optimization problems as multilayer chromosomes. However, this encoding is more complex than all the other chromosome implementations described in this section. Moreover, during the mutation operations, the mutations of conditional variables may lead to the appearance of additional dependent variables and it is necessary to ensure that a correct structure of the chromosome is preserved. Finally, the encoding of multilayers chromosome may often requires problem-specific knowledge in order to be efficiently implemented.

5.3.3.5 Hidden-Genes Genetic Algorithms

Abdelkhalik *et al.* [GA11] developed a Hidden-Genes GA (HGGA). In this algorithm, the chromosomes encompass all the genes of all subproblems corresponding to the choice of all the possible combinations of conditional variables. Thus, all chromosomes have the same length. However, not all the genes are taken into account when computing the values of objective and constraint functions. In order to choose which genes are considered or not, activation genes directly related to the conditional variables are introduced. One way to implement the activation genes is to attach a tag vector to each chromosome of the population such that a tag value is assigned to each gene of the chromosome. If the value of the tag is 0, then the corresponding gene will not participate in the objective function evaluation and reciprocally. Figure 5.8 illustrates the chromosome encoding with the tag vectors.

This representation has the advantage of allowing classical reproduction operators to be applied on design variables vectors as well as on tag vectors. Similar or different crossover and mutation operators can be used for both the chromosome and its associated tag vector. This encoding has the advantage of being intuitive and easily implementable and generic to different problems. However, as all the possibly present variables are encoded within all the chromosomes, the number of genes may become considerably large and therefore inefficient memory-wise in case of complex problems characterized by either a large number of combinations of conditional variables or a large number of depending continuous and discrete variables in each subproblem. It has been mostly applied to the optimization of interplanetary mission trajectories [Abd13; DA18; EAE22] but also for coverage and wind

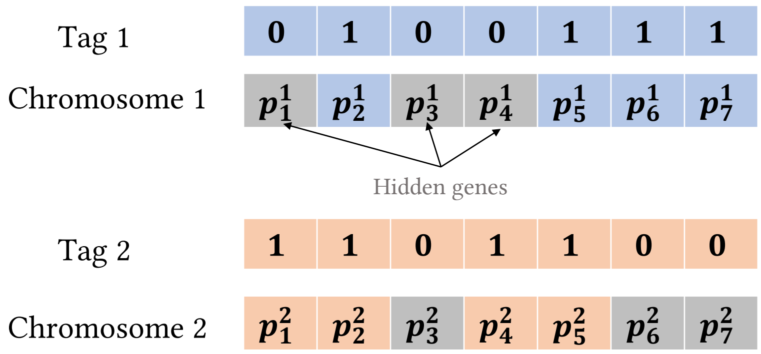


Figure 5.8: Chromosome representation in HGGA.

farm optimal layout [Rye+17].

In this section, different population-based algorithms are revisited in order to deal with conditional search space problems have been reviewed. Mostly evolutionary algorithms as well as PSO have been studied. They often consist in modifying the implementation of the design variables in order to take into account the additional conditional variables and their connections to depending variables. Other techniques than metaheuristics have been adapted for conditional search space problems as for instance, Bayesian Optimization introduced in the next section.

5.3.4 Bayesian Optimization

A machine learning technique that can be used for solving conditional search space problems is Bayesian Optimization (BO) using Gaussian Processes (GP) which has recently been applied for optimal design of various devices and systems especially in case of black-box and computationally expensive objective functions [Sha+15]. GP surrogate modeling is a technique for modeling complex and expensive-to-evaluate systems using GP regression. It involves a GP to approximate the relationship between inputs and outputs of a system (*e.g.*, a black-box function), based on a limited set of observations of the system. The GP model is then used as a surrogate for the system, allowing for predictions at new input points without the need to run the expensive simulation or evaluation of the black-box function. GP also provides uncertainty models along with the prediction model. Figure 5.9 illustrates the GP prediction and uncertainty models on an example [RCM20]. On the figure, a true function (*i.e.*, the black-box studied function) is shown by an orange dotted line. Five observed values are illustrated in blue dots and are used by the GP in order to build a mean predicted function illustrated by a blue line, as well as a confidence interval shown with a light blue zone.

GP is then enriched during the optimization process in order to converge to the optimum. The enrichment process involves an auxiliary optimization process of an infill criterion in order to find the best valuable candidate to be evaluated at the lower value of the criterion. Most of the surrogate modeling techniques as well as Bayesian Optimization algorithm frameworks have been developed to model and optimize continuous problems [JMW98]. However, they have lately been adapted to deal with mixed-variable problems. The most naive approach consists in relaxing discrete variables into continuous variables and rounded them afterward [GH20]. More advanced mixed continuous/discrete surrogate modeling techniques have been developed and are reviewed in [BZ17], [Mec+01], [Pel+20] and [Swi+14]. Lately, BO frameworks have also been adapted to deal with CSS optimization problems [AHD23; Pel20]. In the following subsections, the BO frameworks using GP for mixed-variables and then for CSS problems are detailed.

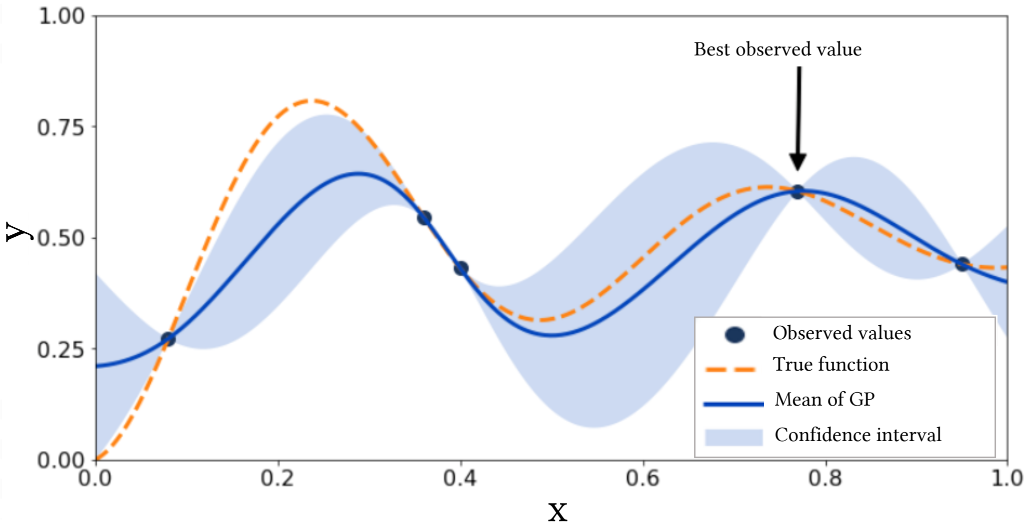


Figure 5.9: Illustration of the Gaussian Process surrogate prediction and uncertainty models on an example [RCM20].

5.3.4.1 Bayesian Optimization for mixed-variables problems

Gaussian Process Surrogate Modelling. A surrogate model is created by generating a training data set $\mathcal{T}$ of n samples defined by their continuous and discrete variables $\mathbf{x}_i$ and $\mathbf{z}_i$ as well as its corresponding objective function evaluation y_i: $\{\mathbf{x}_i, \mathbf{z}_i, y_i\}$ with $i \in \{1, n\}$. $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \{\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_{N_{DoE}}\} \in F_x, \mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_{N_{DoE}}\} \in F_z,\ \mathbf{y} = \{y_1, ..., y_{N_{DoE}}\} \in F_y\} \tag{5.5}$$

where $\mathbf{X}$ is the matrix containing the N_{DoE} vectors of continuous variables, $\mathbf{Z}$ is the matrix containing the N_{DoE} vectors of discrete variables both constituting the training data set (with definition domain F_x and F_y), and $\mathbf{y}$ is the vector containing the corresponding functions evaluations *i.e.*, objective and constraints functions (with definition domain F_y). A generic Gaussian Process $\mathcal{Y}(\mathbf{x}, \mathbf{z})$ is defined by its mean function μ and its covariance function also called kernel function Cov between two samples $\{\mathbf{x}, \mathbf{z}\}$ and $\{\mathbf{x}', \mathbf{z}'\}$:

$$\mu(\mathbf{x}, \mathbf{z}) = \mathbb{E}[\mathcal{Y}(\mathbf{x}, \mathbf{z})] \tag{5.6}$$

$$Cov(\mathcal{Y}(\mathbf{x}, \mathbf{z}), \mathcal{Y}(\mathbf{x}', \mathbf{z}')) = \mathbb{E}[(\mathcal{Y}(\mathbf{x}, \mathbf{z})) - \mu(\mathbf{x}, \mathbf{z}))(\mathcal{Y}(\mathbf{x}', \mathbf{z}') - \mu(\mathbf{x}', \mathbf{y}'))] \tag{5.7}$$

Thus, the GP describing the generic black-box function $f(\cdot)$ is defined as:

$$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), Cov(\cdot)) \tag{5.8}$$

It must be noted that the covariance function is often written $k(\cdot)$ such that:

$$Cov(\mathcal{Y}(\mathbf{x}, \mathbf{z}), \mathcal{Y}(\mathbf{x}', \mathbf{z}')) = k(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) \tag{5.9}$$

The GP predicts the function value $\hat{y}$ at position $\{\mathbf{x}^*, \mathbf{z}^*\}$ using a Gaussian distribution described by the mean and kernel functions, and conditioned by the training data set $\mathcal{T}$. In that respect, the kernel function is a core component of the GP surrogate modeling and is defined as an input space dependent parameterized function. The predicted value f^* of this function at an unmapped location $\{x^*, z^*\}$ is then computed under the form of a Gaussian distribution conditioned on the data set [RW06]:

$$f^* | \mathbf{x}^*, \mathbf{z}^*, \mathbf{X}, \mathbf{Z}, \mathbf{y} \sim \mathcal{N}(\hat{y}(\mathbf{x}^*, \mathbf{z}^*), \hat{s}(\mathbf{x}^*, \mathbf{z}^*)) \tag{5.10}$$

where $\hat{y}(\mathbf{x}^*, \mathbf{z}^*)$ and $\hat{s}(\mathbf{x}^*, \mathbf{z}^*)$ are respectively the mean and standard deviation of the function prediction. Expressions can be found in [RW06].

Kernels. Generally speaking, the kernel is a mathematical function that defines the similarity between pairs of input points following the Reproducible Kernel Hilbert Space (RKHS) formalism [RW06]. Consequently, it has a significant impact on the performance of the GP (*e.g.*, its prediction capabilities). In order for a function $k(\cdot)$ to represent a valid covariance, there are two main requirements: $k(\cdot)$ must be symmetric and positive semi-definite [Aro50]. To deal with mixed-variables optimization problems, a popular approach to define valid kernels is to combine independent kernels defined in the continuous conditional space and kernels defined in the discrete conditional space, as is proposed by Roustant *et al.* [Rou+20]:

$$k(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = k_x(\mathbf{x}, \mathbf{x}') \times k_z(\mathbf{z}, \mathbf{z}') \tag{5.11}$$

where $k_x(\cdot)$ is a kernel defined with respect to the continuous variables and $k_z(\cdot)$ is a kernel defined with respect to the discrete variables. Examples of continuous kernels can be found in [RW06]. Popular ones comprise Squared Exponential and Matèrn.

Some discrete kernels have been developed in the literature. The most simple one is the Compound Symmetry (CS) characterized by a single covariance value for any non-identical pair of inputs [Rou+20]. A particular case of the CS kernel is obtained by considering the covariance in the mixed continuous discrete search space to be spatially dependent as a function of the so-called Gower distance, as is proposed by Halstrup [Hal16]. In the Gower distance [Gow71], the coordinates of the two samples that are being considered are compared dimension-wise. For the continuous dimensions, the distance is proportional to the Euclidean distance, while for the discrete dimensions the distance is the Hamming distance d_H [ASN14].

Another noteworthy discrete kernel is the Latent Variable (LV) kernel proposed by Zhang *et al.* [Zha+19]. Other discrete kernels exist such as the Hypersphere decomposition kernel [ZQZ11] or the Coregionalization kernel [ARL12]. Recently, Saves *et al.* developed a class of kernels for GP models that extends the exponential continuous kernels to the mixed-categorical setting. The authors showed that the class of kernels generalizes Gower distance and continuous relaxation based kernels [PSa+23].

Bayesian Optimization Algorithm. Each iteration of the Bayesian Optimization process can be subdivided into two main phases: the first phase consists in creating a GP based surrogate model of the objective and constraint functions from the data set $\mathcal{T}$. During the second phase, the algorithm minimizes or maximizes an acquisition function whose aim is to determine the next promising location in the design search space. The most employed acquisition function is the Expected Improvement (EI) function [JSW98]. The EI function represents the expected value of the improvement in terms of the objective function value and with respect to the data set. It notably depends on the minimum of the objective value within the data set at the given iteration and the mean and variance of the objective function prediction. The EI acquisition function for mixed-variable formulations is expressed as follows:

$$\mathbb{E}[I(\mathbf{x}^*, \mathbf{z}^*)] = \mathbb{E}[\max(y_{min} - \mathcal{Y}(\mathbf{x}^*, \mathbf{z}^*), 0)] \tag{5.12}$$

$$\mathbb{E}[I(\mathbf{x}^*, \mathbf{z}^*)] = (y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*))\Phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}(\mathbf{x}^*, \mathbf{z}^*)}\right) + \hat{s}(\mathbf{x}^*, \mathbf{z}^*)\phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}(\mathbf{x}^*, \mathbf{z}^*)}\right) \tag{5.13}$$

where y_{min} is the minimum feasible value present within the data set at the given BO iteration, $\hat{y}(\mathbf{x}^*, \mathbf{z}^*)$ and $\hat{s}(\mathbf{x}^*, \mathbf{z}^*)$ are the mean and standard deviation of the objective function prediction $\mathcal{Y}(\mathbf{x}^*, \mathbf{z}^*)$, $\Phi(\cdot)$ is the cumulative distribution function of the Normal distribution and $\phi(\cdot)$ is the probability function of the Normal distribution. The EI function thus provides a trade-off between the exploitation of the search space, driven by the first member of Equation 5.13 and the exploration of the search space, driven by the second member of the equation.

Other acquisition functions exist and are for instance reviewed in [PWG13]. Among other notable acquisition functions, the Upper Confidence Bound (UCB) acquisition function [Sri+09] is expressed as follows:

$$UCB(\mathbf{x}^*, \mathbf{z}^*, \lambda) = \hat{y}(\mathbf{x}^*, \mathbf{z}^*) + \lambda\hat{s}(\mathbf{x}^*, \mathbf{z}^*) \tag{5.14}$$

With UCB, the trade-off between exploration is tuned via the parameter λ. Indeed, UCB is a weighted sum of the expected value captured by $\hat{y}$, and of the uncertainty $\hat{s}$, captured by the standard deviation of the GP. When λ is small, BO will favor solutions that are expected to be high-performing. On the contrary, when λ is large, BO rewards the exploration of currently uncharted areas in the search space.

For the chosen acquisition function, an optimization process is needed to determine the most promising point (*i.e.*, the point maximizing the acquisition function) at each iteration. The acquisition function is expected to have a large number of local optima between the locations of the data set samples. The search space of the acquisition function corresponds to the search space of the optimization problem at hand. Then challenges arise when dealing with mixed-variables search space due to the introduction of discrete or categorical variables. A possible approach consists in optimizing the acquisition function in each category of the mixed-variables problem (*i.e.*, optimize the continuous variables for each value of the discrete variables). The optimizations are then purely continuous and can be performed using exact or global optimization techniques. This approach tends to become computationally inefficient when it comes to large numbers of categories. Another approach then consists in optimizing the acquisition function with the help of a mixed-variables optimization methods like for example adapted GA [Pel+21; SNB98]. It must be noted that those optimizations can be computationally expensive as they are performed on the metamodels.

Then, at each iteration, the most promising point (*i.e.*, the point maximizing the acquisition function) is evaluated and added to the data set. At the next step, the GP surrogate model is updated with this new added point. This process is repeated until a stopping criterion is met. Figure 5.10 schematically describes the BO framework. For clarity purpose, figures are integrated to the framework in order to illustrate each step. They correspond to the first iteration of the process and to a continuous search space.

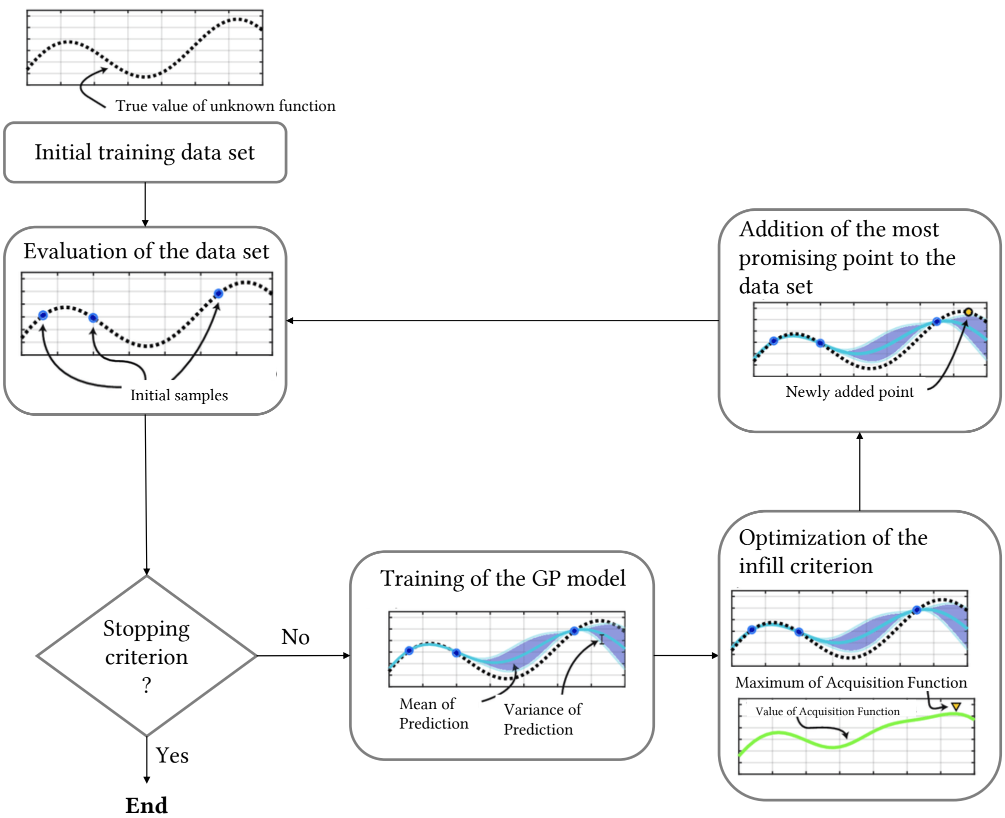


Figure 5.10: Description of the Bayesian Optimization framework. The figures are shown for the first iteration of the process and the illustrations correspond to a continuous function.

Constraint-handling in Bayesian Optimization framework. Different techniques have been developed in order to handle the constrained optimization problems within BO frameworks as discussed in [DMB16], [PBD19] and [Sas02]. Among them, the Probability of Feasibility (PoF) [SWJ98] and the Expected Violation (EV) [Aud+00] approaches are the most employed.

The PoF approach consists in computing the acquisition function as the product of the EI function formulation and the probability of feasibility which is defined as the probability that all constraints of the problem are satisfied at the unmapped location $\{\mathbf{x}^*, \mathbf{z}^*\}$:

$$PoF(\mathbf{x}^*, \mathbf{z}^*) = \prod_{i=1}^{N_g} \mathbb{P}(G_i(\mathbf{x}^*, \mathbf{z}^*) \leq 0) \tag{5.15}$$

where $G_i(\mathbf{x}^*, \mathbf{z}^*)$ is the GP prediction of the inequality constraint i (among N_g inequality constraints) at the location $\{\mathbf{x}^*, \mathbf{z}^*\}$. Under the assumption that the GP prediction follows a normal distribution of mean $\hat{g}_i$ and standard deviation $\hat{s}_{g_i}$ *i.e.*, $G_i(\mathbf{x}^*, \mathbf{z}^*) \sim \mathcal{N}(\hat{g}_i(\mathbf{x}^*, \mathbf{z}^*), \hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*))$, then Equation 5.15 can be written:

$$PoF(\mathbf{x}^*, \mathbf{z}^*) = \prod_{i=1}^{N_g} \mathbf{\Phi}\left(\frac{0 - \hat{g}_i(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*)}\right) \tag{5.16}$$

The new infill criterion can thus be computed as:

$$IC(\mathbf{x}^*, \mathbf{z}^*) = \mathbb{E}[I(\mathbf{x}^*, \mathbf{z}^*)] PoF(\mathbf{x}^*, \mathbf{z}^*) \tag{5.17}$$

Finally, the location of the new infill point to be evaluated at a given BO iteration is determined through the optimization of the following unconstrained problem:

$$\begin{aligned} \max_{\mathbf{x},\mathbf{z}} \quad & IC(\mathbf{x}, \mathbf{z}) \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subset \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z \subset \mathbb{N}^{n_z} \end{aligned} \tag{5.18}$$

Alternatively, the EV criterion can be used to handle the constraints in the BO framework. The EV represents the expected value of the violation of a given constraint *i.e.*, the difference between the predicted value and the maximum acceptable value usually set to 0:

$$V_i = \max(G_i(\mathbf{x}^*, \mathbf{z}^*) - 0, 0) \tag{5.19}$$

Then, the EV for a given inequality constraint g_i is defined in a similar way to the EI acquisition function:

$$\begin{aligned} \mathbb{E}[V_i(\mathbf{x}^*, \mathbf{z}^*)] &= \mathbb{E}[\max(G_i(\mathbf{x}^*, \mathbf{z}^*) - 0, 0)] \\ &= (\hat{g}_i(\mathbf{x}^*, \mathbf{z}^*) - 0)\mathbf{\Phi}\left(\frac{\hat{g}_i(\mathbf{x}^*, \mathbf{z}^*) - 0}{\hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*)}\right) + \hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*)\phi\left(\frac{\hat{g}_i(\mathbf{x}^*, \mathbf{z}^*) - 0}{\hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*)}\right) \end{aligned} \tag{5.20}$$

Then, considering the EI function as the infill criterion, the location of the new infill point to be evaluated at a given BO iteration is determined through the optimization of the following constrained problem:

$$\begin{aligned} \max_{\mathbf{x},\mathbf{z}} \quad & \mathbb{E}[I(\mathbf{x},\mathbf{z})] \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subset \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z \subset \mathbb{N}^{n_z} \\ \text{s.t.} \quad & \mathbb{E}[V_i(\mathbf{x},\mathbf{z})] \leq \delta_i \quad for \quad i \in \{1,...,N_g\} \end{aligned} \tag{5.21}$$

where δ_i is the maximum accepted violation for the constraint i.

Then, it must be noted that the two approaches lead to either an infill criterion unconstrained optimization problem in the case of the PoF approach or to an infill criterion constrained optimization problem in the case of the EV approach. Then, it has to be taken into consideration when choosing an appropriate algorithm to handle them [Pel+21].

5.3.4.2 Bayesian Optimization for conditional search space

Mixed-variable Bayesian Optimization frameworks were again extended to tackle CSS problems. In [AHD23], the authors proposed a general mathematical framework to model constrained mixed-variable optimization problems in a blackbox optimization context including metavariables and categorical variables. Their notation supports main blackbox mixed-variable optimization approaches including surrogate model-based techniques. As a result, specific kernels are required to handle CSS as: the Arc [HO13a], Indefinite Conditional [ZH18b], Imputation [ZH18b], Wedge [Hor+19], Budget Allocation Strategy [Pel+21], Sub-Problem-Wise (SPW) Decomposition [Pel+21] and Dimensional Variable-Wise (DVW) Decomposition [Pel+21] kernels. They are detailed in the following paragraphs.

Extended Square Exponential Kernels. A standard GP model does not incorporate information about a variable's activity state (*i.e.*, if a variable is relevant in the conditional category at hand). The idea proposed in [HO13b; Swe+14; ZH18a] is to integrate that knowledge into the kernel function. Thus, the Arc, Indefinite Conditional, Imputation and Wedge kernels are based on the square exponential kernel in which the distance has been modified following the principles of Hutter et Osborne [HO13b] regarding the requirements to be met by kernels designed for CSS problems.

Budget Allocation Strategy. The Budget Allocation Strategy is based on the decomposition of the CSS problem into mixed-variable fixed search space subproblems [Pel+21]. The underlying idea is to rely on the information provided by the surrogate models of each subproblem objective and constraint functions along the optimization process in order to determine which subproblems are more likely to contain the global optimum. At each iteration of the process, this information provided by the various subproblems surrogate models is exploited in order to allocate a different computational budget to each subproblem (in terms of infilled data samples) proportionally to how promising it is. Moreover, in order to further optimize the usage of the overall computational budget, the subproblems least likely to contain the global optimum can also discarded during the process.

Subproblem-Wise Decomposition Kernel. An alternative way of defining a kernel in a conditional search space consists in separately computing the within-subproblem covariance (*e.g.*, covariance between samples belonging to the same subproblem) and the between-subproblems covariance (*e.g.*, covariance between data samples belonging to different subproblems [Pel+21]). By definition, the within-subproblem covariance can be represented through a mixed-variable kernel. The between-subproblems covariance, instead, can be represented as a discrete kernel defined on the search space of the (combinatorial) conditional variables $\mathbf{w}$. The global variable-size design space kernel can then be computed as the sum between the within-subproblem and between-subproblems covariances.

Dimensional Variable-Wise Decomposition Kernel. Another alternative variable-size design space kernel can be defined by considering a separate and independent kernel for each conditional variable (*e.g.*, subproblem) and the continuous and discrete variables which depend on it [Pel+21]. In fact, it can be noticed that each conditional variable related to a number of possibly active continuous and discrete design variables depending on its value is equivalent to a CSS problem characterized by a single conditional variable, and can therefore be modeled through the SPW decomposition kernel described in the previous paragraph. Thus, the DVW Decomposition kernel is then computed as a product of n_w SPW kernel (*e.g.*, one for each conditional variable).

The previous sections described the Bayesian Optimization process and its extensions to mixed-variable and conditional search space problems which mostly occur during the kernels functions definitions. Generally speaking, Bayesian Optimization can often converge in less objective function evaluations than other algorithm counterparts like metaheuristic-based algorithms, making this approach highly competitive particularly for expensive-to-evaluate functions problems. However, Bayesian Optimization can originally only handle a few dozens of variables and may be thus limited in case of complex formulation of real-world optimization problems. Research studies are currently conducted in order to deal with large-scale BO also referred to as high-dimensional BO [LI+18; Ran+17; Sav+22].

Among the benchmarks used in order to assess the Bayesian Optimization for conditional search space problems, the Combined Algorithm Selection and Hyperparameter optimization problem is one of the most employed. Therefore, the next section aims at briefly describing this problem and the BO-based methods used for solving it.

5.3.4.3 The Combined Algorithm Selection and Hyperparameter optimization problem

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem, sometimes also called Full Problem or Model Selection [EMS09; SPM12], refers to the task of jointly determining the best machine learning algorithm and its corresponding hyperparameter settings for a given dataset and learning task. It involves simultaneously selecting the most suitable algorithms from a set of candidates and optimizing the hyperparameters of the chosen algorithms. It has been introduced in [Tho+13] and belongs to the category of CSS problems. Indeed, the settings of the hyperparameters directly rely on the choices of the algorithm and some algorithms may have categorical hyperparameters (*i.e.*, some parameters need only be specified if another parameter is active or has a certain value).

In order to deal with the CASH (or FMS) problem, Section 5.3.3.1 reviewed the use of a classical PSO algorithm [EMS09]. Alternatively, mostly BO-based methods have been proposed. Considering BO based on GPs surrogate modelling, the aforementioned kernels can be used to deal with the CASH problem. For instance, in [Swe+14] proposed the Arc-kernel in order to optimize the structure of neural networks with unknown numbers of layers. However, Feurer *et al.*, in [FH19], stated that alternative models for BO would be more suitable in case of large, categorical and conditional configuration spaces than standard GPs, like for instance random forests [Bre01; HHK11; Jen+17] in which the probabilistic regression model is an ensemble of regression trees and which, by their structure, allow the use of conditional variables. Thus, the Sequential Model-based Algorithm Configuration (SMAC) method is introduced in [HHK11] and corresponds to a random forests-based Bayesian Optimization framework. It is used in the automated machine learning (AutoML) tool designed to simplify the process of algorithm selection and hyperparameter optimization which is called Auto-WEKA [Tho+13]. Auto-WEKA is built on top of the machine learning software WEKA (Waikato Environment for Knowledge Analysis) and aims to provide a user-friendly and automated approach of the CASH problem. An alternative method to the GPs and random forests model is the Tree Parzen Estimator (TPE) [Ber+11; Spa+15; ZL18]. TPE uses Parzen estimators organized in trees in order to model two distributions depending on whether the objective function values are below or above specified quantile. Other techniques than Bayesian Optimization have nevertheless been investigated like for instance a deterministic RBF Surrogate [Ili+17], the hyperband algorithm [Li+18], a spectral approach [HKY17].

All the previous described techniques optimize the conditional and depending variables in an unique optimization process. A last technique discussed in this chapter corresponds to bilevel approaches that can be used in order to deal with conditional and depending variables in two nested optimization processes.

5.3.5 Bilevel approaches

5.3.5.1 General overview

Bilevel approaches aim at solving complex problems described by two optimization problems which rely on each other in nested structures [Bar13; Dem02; VC94]. More in details, a bilevel problem is a mathematical optimization problem where the set of all design variables are partitioned between two vectors $\mathbf{x}_1, \mathbf{x}_2$ (with respective search space X_1 and X_2) which are optimized within two leveled optimization problems.

Initially proposed by Stackelberg in the context of game theory [Sin+13; Sta52], a bilevel problem can be mathematically formulated as follows [Bar13; Dem02; MPV13]:

$$
\begin{aligned}
\min_{\mathbf{x}_1 \in X_1} \quad & f_{up}(\mathbf{x}_1, \mathbf{x}_2) \\
\text{s.t.} \quad & \mathbf{h}_{up}(\mathbf{x}_1, \mathbf{x}_2) = 0 \\
& \mathbf{g}_{up}(\mathbf{x}_1, \mathbf{x}_2) \leq 0 \\
& \min_{\mathbf{x}_2 \in X_2} f_{low}(\mathbf{x}_1, \mathbf{x}_2) \\
& \text{s.t.} \qquad \mathbf{h}_{low}(\mathbf{x}_1, \mathbf{x}_2) = 0 \\
& \qquad\qquad \mathbf{g}_{low}(\mathbf{x}_1, \mathbf{x}_2) \leq 0
\end{aligned}
\tag{5.22}
$$

where f_{up}, $\mathbf{h}_{up}$ and $\mathbf{g}_{up}$ are respectively the objective function and constraint functions of the **upper level** and f_{low}, $\mathbf{h}_{low}$ and $\mathbf{g}_{low}$ are respectively the objective function and constraint functions of the **lower level**.

As stated in [Bar13], the terminology surrounding the field of bilevel programming may be ambiguous. Thus, as in [Bar13], a relaxed view is adopted in this thesis and the terms of bilevel problems, optimization and algorithms are used to refer to nested problems described by Equation 5.22. In a general context, bilevel or two-level problems refer to various kinds of problems exhibiting a hierarchical structure. Then, the upper level determines the optimal values of its design variables so as to minimize its objective. Those upper optimal values subsequently become parameter values for the lower level which minimizes its objective function taking as arguments a second set of depending design variables and the aforementioned parameter values.

5.3.5.2 Hierarchical approaches for conditional search space problems

Bilevel optimization can be used in order to handle separately conditional and depending design variables. This approach is composed of two loops: the outer loop which specifies the values of the conditional variables and the inner loop which optimize the depending continuous and discrete variables. The bilevel approach applied to conditional search space (CSS) problems can be mathematically formulated as follows [Pel20; SMD17]:

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & f(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{w} \in F_w \\
& \{\mathbf{x}^*, \mathbf{z}^*\} = \text{argmin } f(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \\
& \text{w.r.t.} \quad \mathbf{x} \in F_x(\mathbf{w}), \mathbf{z} \in F_z(\mathbf{w}) \\
& \text{s.t.} \qquad \mathbf{h}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) = 0 \\
& \qquad\qquad \mathbf{g}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \leq 0
\end{aligned}
\tag{5.23}
$$

A bilevel optimization algorithm framework in the case of CSS problems can be illustrated as shown on Figure 5.11.

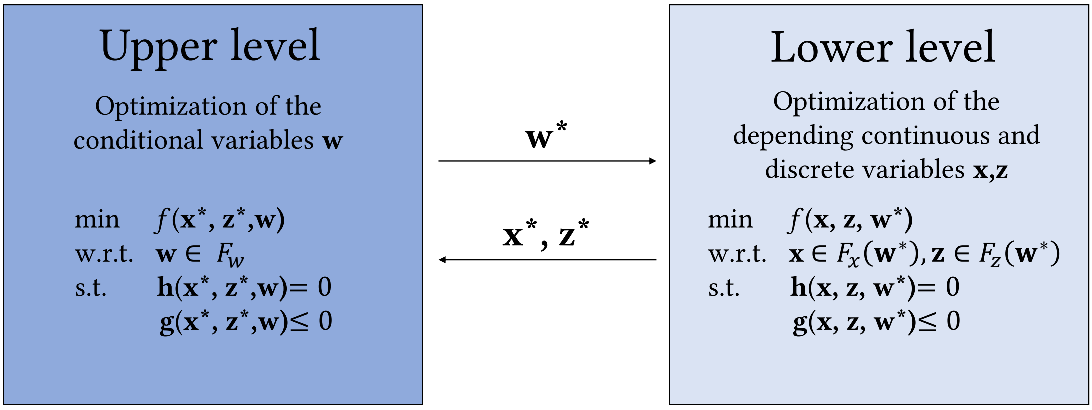


Figure 5.11: Nested algorithm frameworks schematically formulated.

The above formulations can be generalized including continuous and discrete variables in the upper level. Moreover, the upper objective function and the lower objective function can eventually differ. Many nested algorithms structures have been developed in the literature exploring various techniques as upper and lower levels which correspond to CSS optimization problems.

Upper level. The upper level is responsible for optimizing the conditional variables. Exact methods can be employed if the combinations of the conditional variables can be explored in an exhaustive way [HL04]. Then, any metaheuristic (or specific heuristic) can be used to solve the upper level problems. Among them, GA [LR21; MPA94; MPA00], PSO [HM06; ZW19], DE [AKB13; Koh07; ZYW06], *etc.* have been used. Lately, other techniques have been used as upper levels as BO [DP22; Kie+21; Yin+22]. BO often provides better convergence speed in fewer iterations than metaheuristic methods and thus constitutes a relevant approach in order to reduce the computational cost of bilevel algorithms especially in case of computationally expensive functions problems.

Lower level. As for the upper level, various techniques can be embedded as lower levels depending on the difficulty of the lower problem. Exact algorithms like for instance branch and bound techniques [KAA10; Yin+22] can be used in case of low complexity. Otherwise, metaheuristic methods are often preferred. They can be identical to the solution selected à for the upper level in homogeneous nested approaches [LR21; ZW19].

Table 5.1 reviews contributions using different techniques for both levels and validating their approach with various application cases.

Table 5.1: Some bilevel algorithms, their structure and application cases. *SI: Swarm Intelligence.

Reference	Upper level	Lower level	Application cases
[DTS08]	GA	Path Enumeration	Road network design
[Zuo+10]	GA	Exact method	Road costing problems
[Kap+19]	GA	Heuristic	Production system design
[LR21]	GA	GA	Assembly line design
[Ma+22]	GA	GA	Energy storage systems sizing
[SPM12]	GA	PSO	Full-model selection problem
[HM06]	PSO	PSO	Chemical systems optimization
[ZW19]	PSO	PSO	Benchmark functions
[LZW18]	VNS	SS	Infrastructure location strategy
[Tho+21]	SI*	SI	Energy storage systems sizing
[KAA10]	TS	Branch and Bound	Facility Location
[Koh07]	DE	Gradient descent	Network design
[PC11]	SA	Fast gradient descent	Highway network design
[SS17]	ACO	GA	Traffic signal optimization
[Kie+21]	BO	SQP	Benchmark functions
[Yin+22]	BO	Branch and bound	Network design

5.4 Literature summary

Table 5.2 sums up all the methods which have been reviewed for solving both CSS problems formulated in Section 5.2. When possible, for each approach, the very first reference or at least a representative reference is given as well as the main references from the present review.

Table 5.2: Summary of the literature reviews.

Approaches	Algorithms	Definition	Main ref.
Exact exploration		[Fra+18]	[Fra+18]
Direct search		[AHD23]	[AHD23; AD06; ADT19]
Metaheuristic	PSO	[EMS09]	[EMS09]
	Variable-length	[SFT23]	[CML19; Rye+17; TXZ18]
	DSMPGA	[AG12]	[AG12]
	Structured GA	[NAO15]	[NAO15; Gen+19]
	HGGA	[GA11]	[Abd13; DA18; EAE22]
Bayesian Optimization		[Sha+15]	[AHD23; HO13a; Pel+21]
Bilevel		[Bar13; Dem02]	[Ma+22; SPM12; Yin+22]

5.5 Literature review synthesis

In the previous sections, different algorithms allowing to solve conditional search space problems are presented and discussed. It can be noticed that the CSS problems formulation has only rarely been applied to OLPs. Such problems often present a large discrete and conditional search space as well as being often highly constrained. A lot of methods previously discussed tend to rely on the specific nature of the problem and it may be challenging to generalize their functioning to any conditional search space (CSS) problem without including problem specific knowledge. Moreover, several approaches depend on the definition of a distance metric in order to define the neighborhood of a candidate solution like BO or PS algorithms [AHD23].

Furthermore, CSS problems can often be high dimensional and require methods allowing to deal with several hundreds of variables and constraints as well as a complex conditional search space. Many methods such as exact methods or BO are not suitable for high dimensional problems [Pel+21]. Therefore, most of the previous methods used to directly tackle CSS problems with hundreds of dimensions and constraints rely on metaheuristic frameworks. Thus, a GA-based algorithm will be developed as a first approach for solving CSS OLPs in this thesis.

A last limit which can be identified is the computational cost of the optimization. Exact methods, heuristic, metaheuristic and direct search techniques tend to require a large number of function evaluations in order to converge due to their necessity of exploring a large part of the design space. Furthermore, it has been highlighted in [KS05; MC11; Sal09] that pure metaheuristic techniques may present some weaknesses for solving problems with complex combinatorial spaces or high number of constraints. In order to overcome these issues, hybridizations and particularly bilevel approaches can be used in order to take advantage of several methods. Moreover, Bayesian Optimization is known to converge in less function evaluations than most of the other technique discussed [Pel+21]. For this reason, bilevel approaches using Bayesian Optimization will be considered in this thesis as a second approach for dealing with CSS OLPs.

5.6 Conclusion

In this chapter, the mathematical formulation of conditional search space optimal layout problems (CSS OLPs) is presented as well as the existing methods allowing to handle them. In more detail, the optimization problem resulting from the inclusion of technological and architectural choices within the system design framework is presented. Those technological choices are included in the mathematical formulation under the form of conditional variables which can modify the search space as a function of their values, in terms of number and type of design variables the problem functions depend on. In the same way, they can also influence the number and type of constraints, and by consequence the feasibility domain. Then, the existing optimization algorithms allowing to optimize the resulting CSS problems are described and discussed. The literature review about CSS optimization problems first shows that the CSS formulation has rarely been applied to OLPs. Moreover, both literature reviews about FSS OLPs and CSS problems show that most of the techniques do not seem to simultaneously provide a generic framework able to deal with hundreds of variables, with a sufficiently efficient constraints-handling technique (to guarantee a trade-off between exploration of the feasible design space and exploitation toward the optimum without loss in precision), as well as a sufficiently fast convergence (in terms of function evaluations) toward the global optimum. They are thus not viable solutions when it comes to computationally intensive combinatorial and constrained system design problems.

For this reason, the objective of rest of this part of the manuscript is to develop optimization methods for CSS OLPs relying both on GA frameworks and on nested approaches combining different techniques as Bayesian Optimization and the CSO-VF algorithm (*i.e.*, a virtual-force system-based method) presented in the previous part of the manuscript. The objective is thus to provide algorithm frameworks for high-dimensional and constrained CSS OLPs which can improve the constraints handling as well as reduce the amount of function evaluations required to find the problem optimum.

The rest of the second part of the manuscript is organized into three chapters. Chapter 5 develops both proposed algorithms for CSS OLPs. The first one is based on a hidden-genes GA framework and the second one is based on a bilevel structure combining Bayesian Optimization and the CSO-VF algorithm proposed in Chapter 2. In Chapter 6, both algorithms are applied to benchmark problems composed of single-container satellite module OLPs and their global performance are compared. Finally, Chapter 7 aims at addressing CSS OLPs enhanced with an assignment task in order to solve multi-container CSS OLPs. A tri-level approach combining Bayesian Optimization, GA and CSO-VF is proposed and applied to a multi-container satellite module layout problem.

Chapter 6

Algorithms for Conditional Search Space Optimal Layout Problems

Contents

Chapter contributions

- Extension of fixed search space optimal layout problems to conditional search space optimal layout problems.
- Development of two different approaches: a hidden-variables mechanism for Genetic Algorithm and a surrogate assisted-Component Swarm Optimization algorithm based on a Virtual-force system approach.

6.1 Introduction

In this chapter, conditional search space optimal layout problems (CSS OLPs) are addressed. As stated in Chapter 5, few CSS OLPs have been defined in the literature. In [Gon+11; Rye+17], wind farms and sensor coverage CSS optimal layouts have been studied. In those papers, all the components to position *i.e.*, the wind turbines or the sensors are identical and thus, the conditional search space is defined by a unique conditional variable corresponding to the number of components to position. The number of subproblems directly corresponds to the number of values that can be reached by this conditional variable and is usually less than a dozen. However, in many real-world engineering design problems, the components to position are not identical. For instance, considering the design of an aerospace concept, a possible choice would either be to place a single tank or to split it into two smaller tanks. Another possible choice would either be to place in the container two small batteries or a big one for the same power generation requirement. Then, architecture choices have to be made for each type of components. Consequently, the purpose is to provide a catalogue of all the components which can possibly be chosen to be part of the system layout and let an optimization algorithm find both the best combinations of components and their positions in the container. Allowing the number of components to vary during the optimization process may provide a better solution at the end.

In Chapter 5, two approaches have been identified as promising in order to address the aforementioned CSS OLPs. The first one is a hidden-variable mechanism for GA while the second one corresponds to the hybridization of the CSO-VF algorithm developed and assessed in Chapter 3 and Chapter 4 of the manuscript with Bayesian Optimization.

Then, the rest of the chapter is organized as follows: in the first section, generic conditional search space OLPs are described and formulated. In Section 2, the hidden-variable mechanism for GAs is described while Section 3 is devoted to the surrogate assisted-CSO-VF algorithm.

6.2 Formulation of conditional search space optimal layout problems

In the present thesis, in the case of CSS OLPs, a catalogue of components is defined at the beginning of the optimization process and the main concern is to optimize the list of the components among the given catalogue as well as their layout in the container. Indeed, some design requirements can be met using several sets of components.

6.2.1 Design variables

The design variables of a CSS OLP are listed as follows:

- Continuous variables $\mathbf{x}$: As for fixed search space (FSS) problems considered in Part I, continuous variables can correspond to the positions of the centers of inertia and angular orientations of the non-cylinder components.

- Discrete variables $\mathbf{z}$: Similarly to [ST03], the components could also be positioned on a grid and their orientations could be restricted to take on a set of specific values (*e.g.*, 0° and 90°). In this particular case, the design variables are discrete. In the context of multi-container problems, discrete variables can also be used to represent the index of a given container within which the component should be located [CZa19].

- Conditional variables $\mathbf{w}$: The conditional variables are used to describe the choice of components that must be included within the system. When all items are identical, the conditional variables can merely correspond to the number of components to be included in the system. For instance, in the case of CSS wind farms layout problems, the wind turbines are identical. Then, only one conditional variable taking the value of the number of turbines can be used [Gon+11]. However, in the case of various shapes, sizes and types of components, several conditional variables are needed to express the choice of components. In the present work, conditional variables are used to define all possible subdivisions of all the components listed in the catalogue. For instance, in aircraft design, a given amount of fuel should be carried. A tank is thus required to accommodate the volume of fuel that is necessary for the aircraft to perform its mission. A number of tank architectures would thus be specified as alternative design options as a means to distribute the fuel over different regions of the aircraft such as its wing or in the rearmost part of the aircraft. Figure 6.1 gives an example of the possible values of conditional variables for one cylindrical component that can be subdivided into either two smaller cylinders or four even smaller ones.

Depending on the value of $\mathbf{w}$, the number of components to position in the container varies and consequently the number of continuous and discrete variables characterizing the layout vary as well. Mathematically, these variables are qualified as categorical.

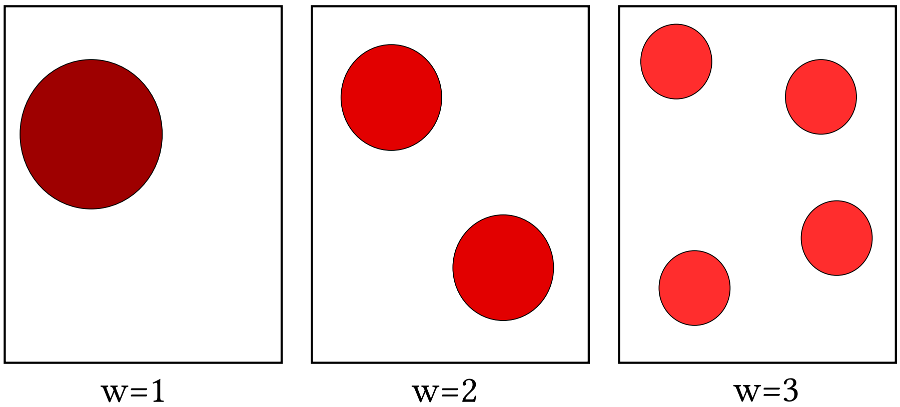


Figure 6.1: Example of conditional variables for three possible subdivisions of one cylindrical component positioned in a rectangular container. The redder the cylinders, the higher their corresponding masses.

Thus, if N components must be positioned in the container and each of them has n_i possible subdivisions ($i \in \{1, ..., N\}$), then the number of possible architecture configurations (*i.e.*, the number of different possible lists of components) corresponds to the cardinality of the set of achievable values of $\mathbf{w}$, F_w, and is mathematically defined as:

$$\text{Card}(F_w) = \prod_{i=1}^{N} n_i \tag{6.1}$$

6.2.2 Objective function

The possible objective and constraints functions remain similar to those detailed in the case of FSS OLPs in Chapter 2. The objective functions that can be optimized are for instance:

- Any dynamical requirements (*e.g.*, overall mass, inertia or stability of the system);

- Any costs related to the layout (*e.g.*, manufacturing or handling costs);
- Any performance of the layout (*e.g.*, power consumption or emission).

6.2.3 Constraint functions

The possible constraint functions are split into two categories:

Constraints relative to the choice of components. Some constraints are specific to the choice of the components. They can involve for instance a maximum value of total mass or a maximum value of occupation rate.

The layout constraints functions. The layout constraints functions rely on the layout of the chosen components. They are similar to those defined for the FSS OLPs and are briefly summarized as follows:

- **Geometrical constraints:** These constraints translate geometric specifications on the layout which only depend on geometric and mass considerations such as no overlapping between the components, the full inclusion components in the container, balancing constraints (*i.e.*, the global center of mass must be positioned at the geometrical center of the container), *etc.*;
- **Functional constraints:** These constraints do not only depend on geometric considerations, but also on system-level specifications. One example is for instance proximity or minimal distance requirements between components for functional requirements (*e.g.*, radiative constraints).

To sum up, as for the multi-container FSS OLPs, the single-container CSS OLPs involve two tasks:

- The "choice of components" task: selecting a set of components from a catalogue;
- The layout task: positioning the selected components into the container.

Figure 6.2 illustrates the CSS single-container OLPs.

6.2.4 Mathematical formulation of the problem

The problem described in the previous sections is mathematically formulated as follows:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{w}} \quad & f_{obj}(\mathbf{x},\mathbf{z},\mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x(\mathbf{w})}, \mathbf{z} \in F_z \subseteq \mathbb{N}^{n_z(\mathbf{w})}, \mathbf{w} \in F_w \subseteq \mathbb{N}^{n_w} \\
\text{s.t.} \quad & \mathbf{h}_{component}(\mathbf{x},\mathbf{z},\mathbf{w}) = 0 \\
& \mathbf{g}_{component}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \\
& \mathbf{h}_{layout}(\mathbf{x},\mathbf{z},\mathbf{w}) = 0 \\
& \mathbf{g}_{layout}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0
\end{aligned} \tag{6.2}$$

where f_{obj} is the objective function which can correspond to FSS OLPs objective functions. $\mathbf{h}_{component}$ and $\mathbf{g}_{component}$ correspond respectively to equality and inequality constraint functions only related to the choice of the components. $\mathbf{h}_{layout}$ and $\mathbf{g}_{layout}$ correspond respectively to equality and inequality constraint functions related to the layout of the set of components.

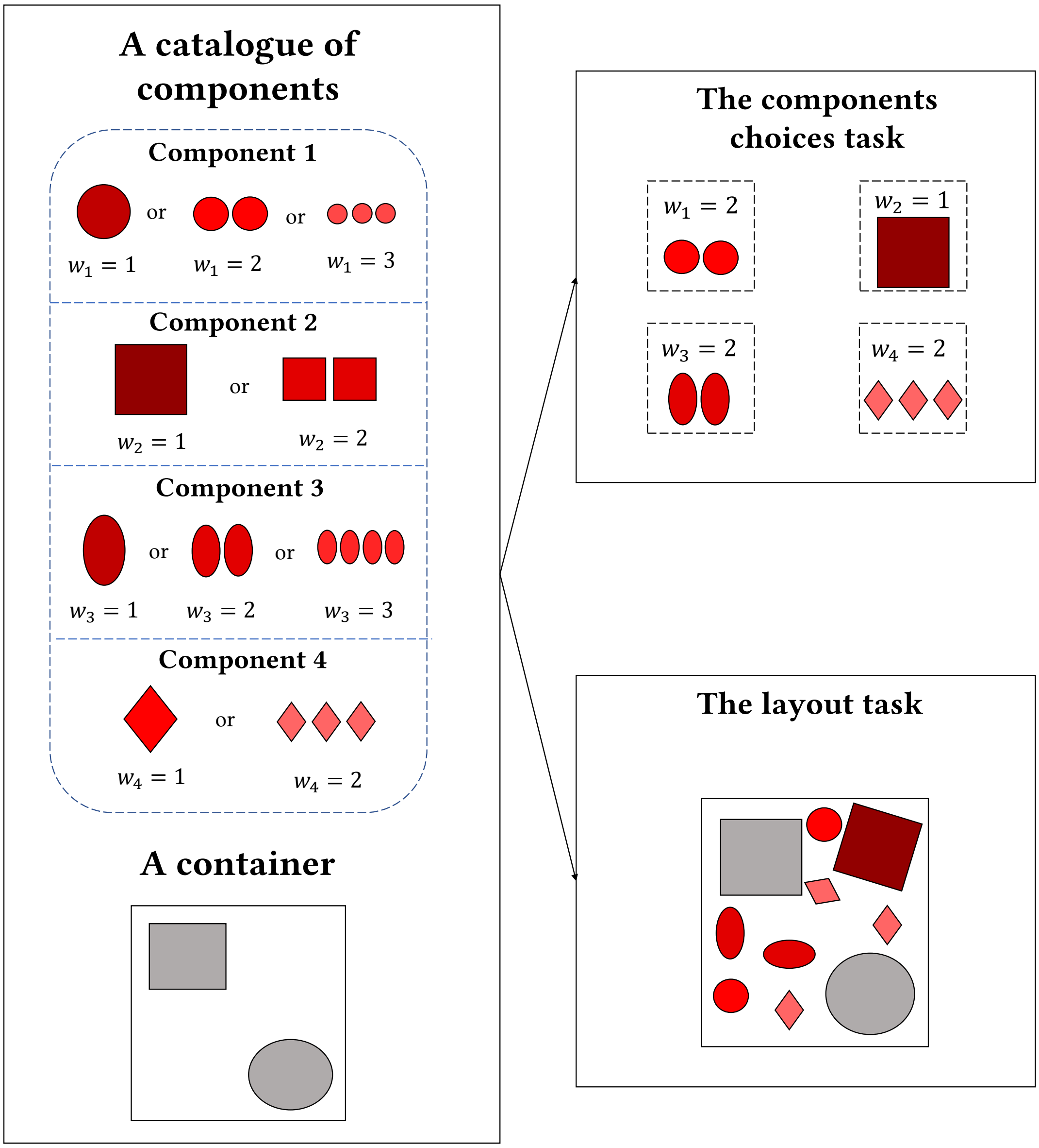


Figure 6.2: Illustration of the CSS single-container OLPs. The redder the components, the heavier their corresponding masses.

6.3 First approach: derivation of hidden-variables mechanism for Genetic Algorithms

6.3.1 Overall description of the proposed mechanism

In order to solve OLPs as formulated above, an adaptation of an hidden-variables mechanism initially described in [GA11] is proposed in this thesis and is integrated in a Genetic Algorithm (GA). This mechanism consists in modifying the encoding of the design variables in comparison to the implementation of classical GA chromosomes. Indeed, all the design variables are implemented so that all the individuals (*e.g.*, the chromosomes) in the population have the same length. However, some of the variables are hidden, meaning that they will not take part in the objective and constraints evaluations. For the aforementioned OLP, the hidden variables represent the design variables $\mathbf{x}$ and $\mathbf{z}$ for the components of the subdivisions that are not selected by the conditional variables $\mathbf{w}$ to be part of the current layout. On the contrary, the expressed design variables correspond to the variables relative to the components selected to be in the container.

Thus, this method allows to tackle the CSS aspect of optimization problems while ensuring that all the chromosomes (*i.e.*, the individuals of a population) have the same length which enables the use of many existing evolutionary operators. This method should thus be suitable for the problems at hand. Indeed, it is well suited for the large number of design variables unlike Bayesian Optimization (if used to deal with all the variables of the problem) or the Dynamic-Size Multiple Population Genetic Algorithm [AG12], especially in case of high combinatorial conditional search space. It is a flexible and generic mechanism limiting the introduction of expert knowledge during the implementation process unlike the Structured GA solution [NAO15] and relatively simple to be implemented. In addition, it allows to keep an entire freedom of choice for the evolutionary operators unlike some techniques which do not guarantee the same number of design variables for all individuals of the population [Rye+17].

Two methods are derived in order to hide variables and those methods will be applied to a GA. Depending on the method used to hide the variables, some of the evolutionary operators must nevertheless be adapted. Those methods as well as the resulting adaptations are detailed in the following section.

6.3.2 Implementations of the hidden-variables mechanism with a genetic algorithm

Different methods exist for the implementation of the hidden-variables mechanism. Two of them are described in the following sections.

6.3.2.1 First encoding method: additional design variables

The first possibility to hide variables is to implement conditional variables as additional design variables. More specifically, conditional variables responsible for the choice of hidden or expressed variables are added to the set of design variables. Before evaluating the objective function and the constraints, the conditional variables are read for each candidate solution and their values indicate which design variables of the candidate solution must be taken into account (and by extension, which must be ignored) to calculate both the objective and constraints. Figure 6.3 provides an illustration of this method on an example. In the example, two alternative subdivisions of a component can be chosen between one cylindrical component or two smaller ones. As only two configurations of the layout are considered, a unique conditional variable taking binary values is enough to describe this feature. If the conditional variable value is 0 then the first subdivision is chosen and the corresponding dependent design variables are expressed. On the contrary, if the conditional variable value is set to 1, then the second subdivision is chosen and the other set of design variables are accounted for. It is worth noting that there exists various ways to define conditional variables in order to describe the possible configurations.

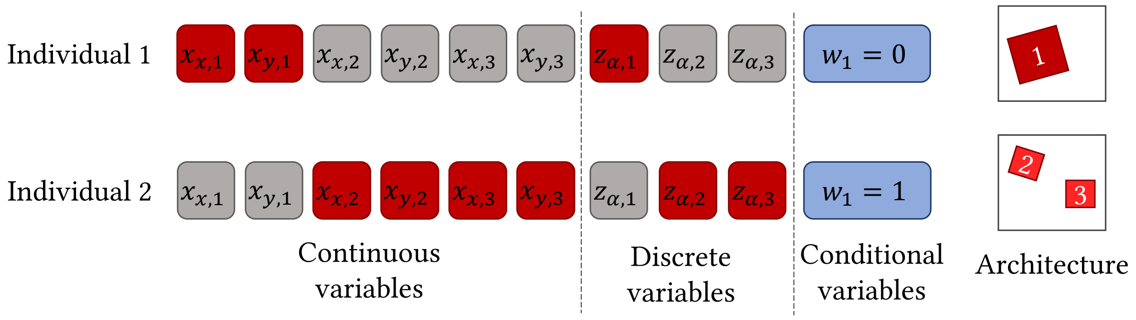


Figure 6.3: Illustration of the implementation of the hidden-variables mechanism using conditional variables. Red variables are the expressed variables while grey ones are the hidden ones.

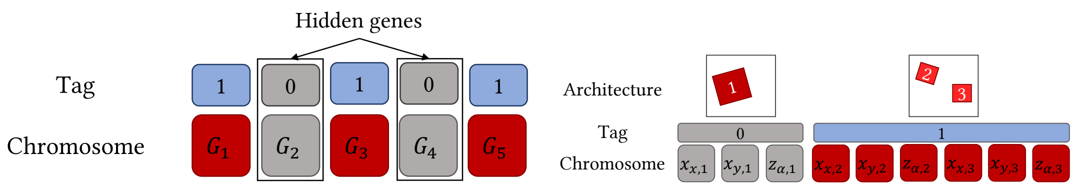
If a catalogue is composed of N generic subdivisible components for a total of N_{tot} available components (amongst which N_{cyl} cylinders), the chromosome length L_{chr} is given by:

$$L_{chr} = N + 2N_{cyl} + 3(N_{tot} - N_{cyl}) \tag{6.3}$$

For this chromosome implementation, classical evolutionary operators can be used without any adaptation.

6.3.2.2 Second method: tags

Another possible way to deal with conditional search spaces is to introduce tags as proposed in [GA11]. In this method, a tag vector is attached to each chromosome of the population such that a tag value is assigned to each gene of the chromosome. If the value of the tag is 0, then the corresponding gene will not participate in the objective and constraints functions evaluations and reciprocally. This mechanism is shown in Figure 6.4a. The tags are a direct translation of the conditional variables: their role is to choose which variables will be expressed or not during the fitness evaluation as well as the corresponding constraints to activate. Figure 6.4b illustrates the adaptation on the same example as in the previous section. If the second subdivision of the component is chosen to be part of the layout, the tag vector will activate all the design variables (coordinates of the center of mass and orientation if needed) to be part of the objective function and constraints evaluations.



(a) Definition of the tags. (b) Adaptation of the tag method.

Figure 6.4: Illustration of the tag mechanism.

With the same formalism as before, the chromosomes length L_{chr} as well as the tags length L_{tag} are computed as follows:

$$L_{chr} = N + 2N_{cyl} + 3(N - N_{cyl}) \tag{6.4}$$

$$L_{tag} = N \tag{6.5}$$

As the chromosomes are extended by tags, the operators of the GA must be compatible with the tags which will take part in the evolution process. The chromosomes and tags will undergo independent operators of crossover and mutation as proposed in [GA11]. Crossover and mutation operators are detailed in the following paragraphs.

Crossover operator for tags

For the chromosomes that encompass continuous and discrete variables related to the layout, the crossover operator can be chosen among all the available operators used in genetic algorithms [Tal09]. The operators considered in this thesis are detailed in Section 6.3.3.1. For the tags, a n-points crossover operator is chosen, without any loss of generality [Tal09]. This mechanism must be adapted as not all the points of the tag vector can be crossover points because of the fact that the design variables are not independent from each other. The crossover operator on the tags must provide a feasible configuration of subdivisions. Figure 6.5 illustrates the principle of the n-points crossover on the tags adapted for the subdivisions of the components. On the example, two components able to subdivide themselves into two smaller components are considered so that there are four possible configurations of components in the container. For each component, only one of the two subdivisions can be present in the container and so in the tag vector, among the three points that may be considered as crossover points only one (indicated in orange) guarantees the fact that the crossover operation leads to feasible individuals.

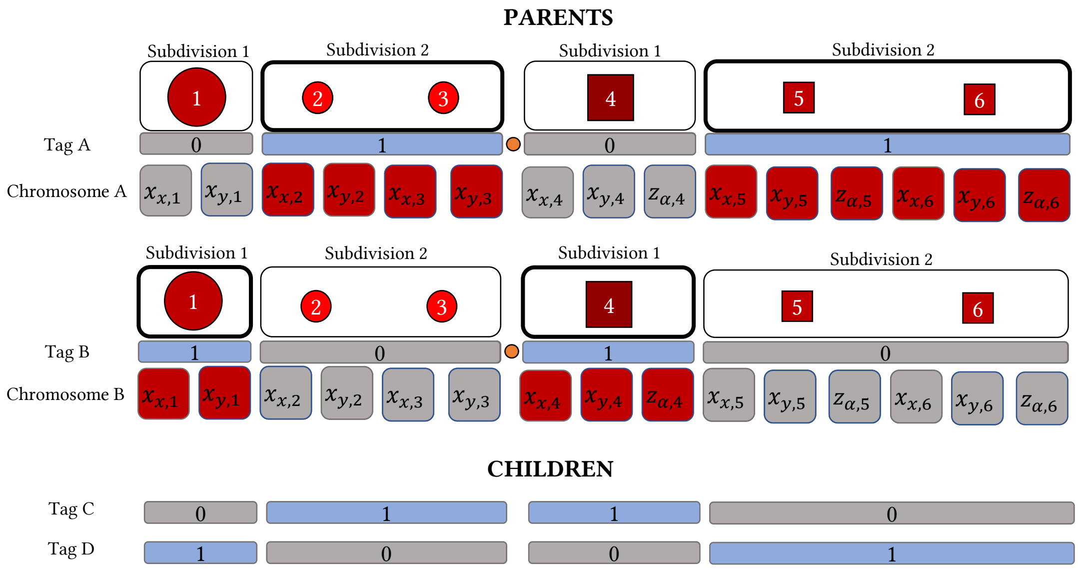


Figure 6.5: Crossover operator for the tags on an example. The orange dots correspond to the crossover location.

Mutation operator for tags

For the mutation operator, again the mechanism differs according to the chromosomes or tags. For the chromosomes, the mutation operator can be chosen amongst all the operators available from "classical" genetic algorithms. The mutation operators studied in this thesis are detailed in Section 6.3.3.2. For the tags, a bit flip operator can be applied [Tal09]. Like the crossover operator, it must be adapted for the subdivision case to provide feasible solutions. The bit flip consists in fact in flipping the subdivision chosen. Hence the tags cannot be flipped randomly as for each component only one tag must be equal to one in order to select a unique subdivision among the possible ones. Figure 6.6 illustrates the mutation operator designed for the tags. Only a few mutations are possible to guarantee the feasibility of the solution.

6.3.3 Evolutionary operators

The conditional search space aspect of the problems at hand induces a more complex and wide search space than fixed search space problems and thus, evolutionary operators should be carefully chosen in order to promote diversity in the population along with the generations in order to avoid premature convergence towards local optima.

In the following paragraphs, the parents and children involved in the crossover operators are respectively denoted p and c. In the case of mutation operator, the mutated gene obtained from a child gene c is denoted c'. More specifically, the i^{th} gene of th j^{th} parent (resp. child) is denoted p_i^j (resp. c_i^j). The same formalism is employed for the mutated genes.

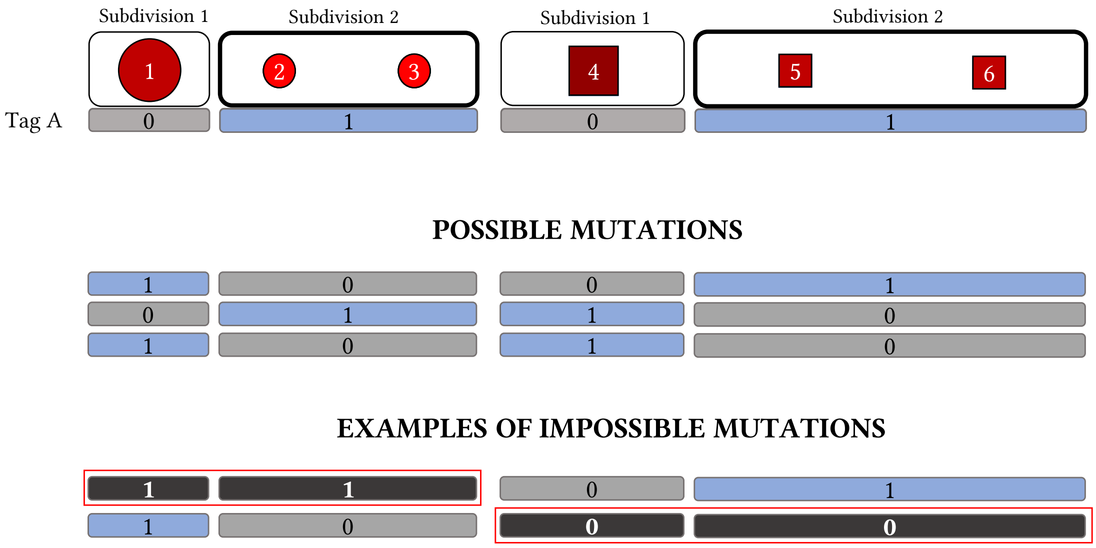


Figure 6.6: Mutation operator for the tags on an example.

6.3.3.1 Crossover operators

Two crossover operators are considered:

- **Simulated-Binary Crossover** (SBX) [DA95]: The SBX operator is one of the most employed crossover operators for the recombination of real-coded chromosomes. It simulates the operation of a single-point binary crossover directly on real variables. The i^{th} genes of the two children involved in the crossover operator, c_i^1 and c_i^2, are obtained from the corresponding i^{th} genes of the two parents, p_i^1 and p_i^2, as follows:

$$c_i^1 = \bar{p_i} - \frac{1}{2}\beta_i(p_i^2 - p_i^1) \tag{6.6}$$

$$c_i^2 = \bar{p_i} + \frac{1}{2}\beta_i(p_i^2 - p_i^1) \tag{6.7}$$

where $\bar{p_i} = \frac{1}{2}(p_i^1 + p_i^2)$ and β_i is obtained thanks to the probability distribution:

$$P(\beta) = \begin{cases} 0.5(\eta+1)\beta_i^{\eta}, & \text{if } \beta_i \leq 1, \\ 0.5(\eta+1)\beta_i^{\frac{1}{\eta+2}}, & \text{otherwise.} \end{cases} \tag{6.8}$$

where η is the distribution index.

- **Uniform Crossover** (UX) [Dan+16]: For each gene, the UX operator swaps the parents genetic material with a certain probability α. In other words, for each of the N_{DV} genes, a uniformly distributed random variable in the range [0, 1] r is generated. The genetic materials are exchanged if r is greater than the given probability α as represented on Figure 6.7.

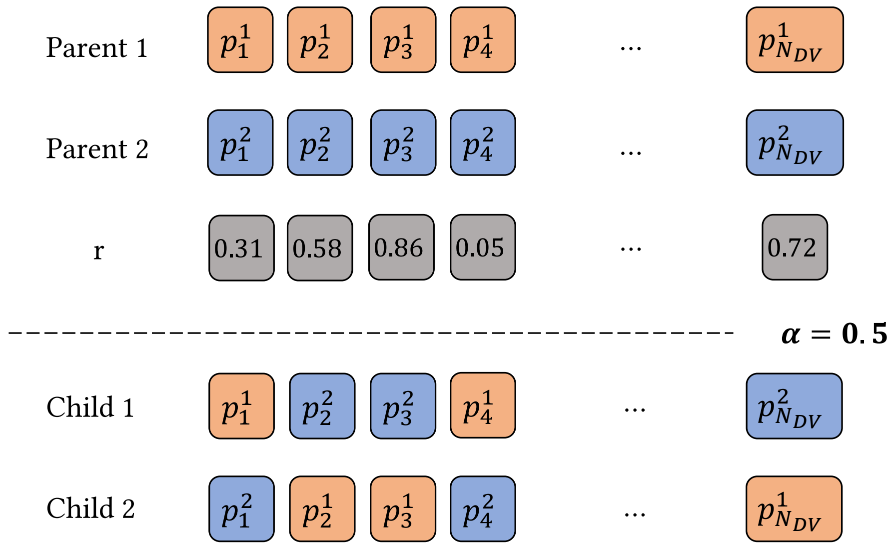


Figure 6.7: Illustration of the UX crossover operator on an example.

6.3.3.2 Mutation operators

Two mutation operators are considered:

- **Polynomial Mutation** (PM) [Deb01]: The genes of the mutated child c_i' are computed thanks to the genes of the original child c_i as follows:

$$c_i' = c_i + \delta_i(c_i^U - c_i^L) \tag{6.9}$$

where c_i^U and c_i^L are respectively the upper and lower bounds for c_i and δ_i is obtained thanks to the probability distribution:

$$P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|^{\eta_m}) \tag{6.10}$$

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m}+1} - 1, & \text{if } r_i \leq 0.5, \\ 1 - (2(1 - r_i))^{\frac{1}{\eta_m+1}}, & \text{otherwise.} \end{cases} \tag{6.11}$$

where η_m is the distribution index and r_i is a uniformly distributed random variable in the range [0, 1].

- **Uniform random Mutation** (UM) [Tal09]: The UM operator goes through the genes and replaces the value uniformly (randomly) with one between the upper and lower values of the corresponding gene.

6.3.3.3 Selection operators

Two selection operators are compared in this thesis. For both operators, the i^{th} selected individual is denoted s_i.

- **Selection based on Tournament and Constraint-dominance** [Deb00]: In the current population, k individuals are chosen randomly. The winner of the tournament is determined thanks to Constraint-dominance rules [Deb00]. The rules are based on the comparison of the individuals objective function values and/or constraints violation values. The constraints violation value of a selected individual s_i is denoted CV_i. In case of N_{cst}^g inequality constraints denoted $\mathbf{g} = \{g_k\}_{k\in\{1,\dots,N_{cst}^g\}}$ and N_{cst}^h equality constraints denoted $\mathbf{h} = \{h_k\}_{k\in\{1,\dots,N_{cst}^h\}}$, the constraints violation CV_i of the individual s_i is computed as:

$$CV_i = \sum_{k=1}^{N_{cst}^g} \max(g_k(s_i), 0) + \sum_{k=1}^{N_{cst}^h} |h_k(s_i)| \tag{6.12}$$

  Algorithm 6.1: describes the routine used to compare two individuals with Constraint Dominance.

Algorithm 6.1: Constraint Dominance

Input: Individual 1: objective function $f_{obj,1}$, constraint violation CV_1 ; Individual 2: objective function $f_{obj,2}$, constraint violation CV_2.
Output: Dominating solution: Individual 1 or 2.

```
if CV1 < CV2 then
  Individual 1 dominates.
else if CV2 < CV1 then
  Individual 2 dominates.
else
  if f_obj,1 < f_obj,2 then
    Individual 1 dominates.
  else if f_obj,2 < f_obj,1 then
    Individual 2 dominates.
  else
    Choose one individual randomly.
  end if
end if
```

- **Selection based on Tournament and Stochastic Ranking** (SR) [RY00]: The Stochastic Ranking is based on bubble sort and gives infeasible solutions or solutions with higher constraints violations a chance to survive during the selection operator. This operator should thus lead to a more diverse population throughout the generations than more elitist operators. The SR operator is as follows: 1) the population is ranked according to dominance rules following the routine described by Algorithm 6.2:, 2) the ranked population is truncated such that m individuals are considered for reproduction, 3) the Tournament operator is applied on the truncated ranked population to select the parents according to their ranks.

  The stochastic ranking of the population detailed in Algorithm 6.2: works as follows: a number of sweeps N_S is defined as well as a SR probability P_{SR}. The individuals of the population are ranked by comparing adjacent individuals in at least N_S sweeps following the procedure: if both solutions are feasible, they are ranked according to their objective values. However, if at least one solution is infeasible, there is a probability P_{SR} that they are also ranked according to their objective function so that infeasible solution or solutions with higher constraints violations have a chance to survive and be part of the crossover and mutation operations and thus increase the diversity of the population.

Algorithm 6.2: Stochastic Ranking

Input: Population of N individuals, SR probability P_{SR}, N_S number of sweeps.
Output: Ranked population.

```
for i = 0 to N_S do
  for j = i to N − 1 do
    Sample r uniformly in the range [0,1]
    if CV_j = CV_{j+1} = 0 or r < P_SR then
      if o_j > o_{j+1} then
        Swap the ranks of individuals j and j + 1 in the population.
      end if
    else
      if CV_j > CV_{j+1} then
        Swap the ranks of individuals j and j + 1 in the population.
      end if
    end if
  end for
  if no swap done then
    break
  end if
end for
```

6.3.3.4 Replacement operators

Two replacement operators are studied:

- **Non-dominated truncating** (NDT) [Deb+02]: The current population is expanded with the generated children population, sorted according to their objective function values and truncated at the population size.
- **Replacement based on Deterministic Crowding** (DC) [Mah92; Mah94]: in order to increase diversity during the replacement phase for the next generation the Deterministic Crowding method compares the parents to their children according to the distance between their genotypes. The distance is defined as a mean between the Euclidean distance between tags and the euclidean distance between the genes. If the genotypes are close, either the parent or the child is chosen to integrate the next population based on the dominance rules chosen for the selection operator [Mah92; Mah94]. Algorithm 6.3: gives the pseudo-code of the DC routine.

Algorithm 6.3: Deterministic Crowding

Input: 2 parents p_1, p_2 and their 2 children c_1, c_2, a distance measure d, a dominance method.
Output: 2 individuals chosen to be part of the next population.
if $d(p_1, c_1) + d(p_2, c_2) \leq d(p_1, c_2) + d(p_2, c_1)$ **then**
  if c_1 dominates p_1 **then**
    Replace p_1 with c_1
  end if
  if c_2 dominates p_2 **then**
    Replace p_2 with c_2
  end if
else
  if c_2 dominates p_1 **then**
    Replace p_1 with c_2
  end if
  if c_1 dominates p_2 **then**
    Replace p_2 with c_1
  end if
end if

In Chapter 7, a Taguchi's plan of experiments compares the aforementioned operators in order to find the best combination of evolutionary operators.

6.3.4 Algorithm framework

Figure 6.8 illustrates the framework of the resulting algorithm, for both implementations of the hidden-variable mechanism. The steps specific to the hidden-variables mechanisms are highlighted in orange. In this thesis, the stopping criterion corresponds to a given number of objective function evaluations.

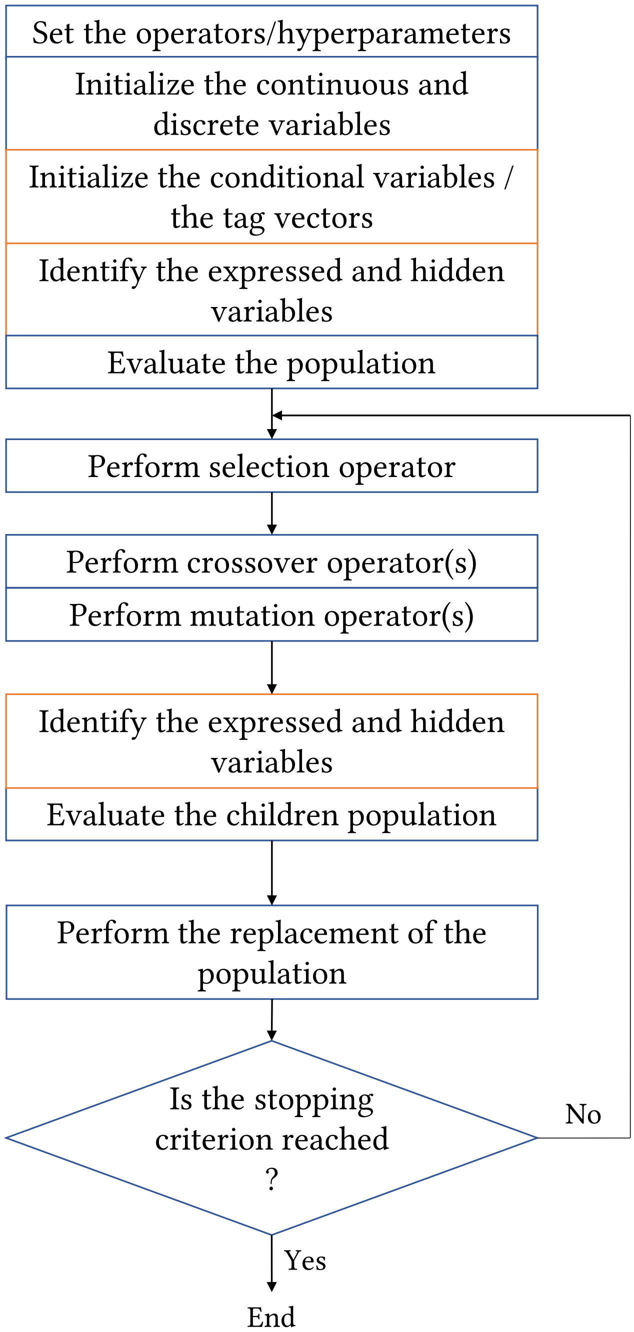


Figure 6.8: GA enhanced with hidden-variable mechanism framework. Steps related specifically to the hidden-variables mechanism are highlighted in orange.

6.4 Second approach: surrogate assisted-CSO-VF algorithm

The second proposed approach to the conditional search space (CSS) single-container OLPs is a bilevel algorithm combining the strength of the CSO-VF algorithm and a discrete Bayesian Optimization algorithm purposely adapted to tackle the dimensional aspect of this problem.

6.4.1 Mathematical formulation of the problem

The mathematical formulation defined in Section 6.2.4 by Equations 6.2 is reformulated in this Section. Indeed, the conditional search space problems involve conditional and dependent variables that allow to write the problem as a nested mathematical formulation, without discarding any solutions. The nested mathematical formulation of the CSS single-container OLP is written as follows:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & f_{obj}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \\
\text{s.t.} \quad & \mathbf{h}_{component}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) = 0 \\
& \mathbf{g}_{component}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \leq 0 \\
& \mathbf{h}^*_{layout}(\mathbf{x}, \mathbf{z}, \mathbf{w}) = 0 \\
& \mathbf{g}^*_{layout}(\mathbf{x}, \mathbf{z}, \mathbf{w}) \leq 0 \\
\text{w.r.t.} \quad & \mathbf{w} \in F_z \subseteq \mathbb{N}^{n_w} \\
& \{\mathbf{x}^*, \mathbf{z}^*\} = \text{argmin } f_{obj}(\mathbf{x}, \mathbf{z}, \mathbf{w}) \\
& \text{w.r.t.} \quad \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x(\mathbf{w})}, \mathbf{z} \in F_z \subseteq \mathbb{R}^{n_z(\mathbf{w})}
\end{aligned} \tag{6.13}$$

where f_{obj} is the objective function which can correspond to fixed search space OLPs objective functions. $\mathbf{h}_{component}$ and $\mathbf{g}_{component}$ correspond respectively to equality and inequality constraint functions only related to the choice of the components. The design variables denoted with a star exponent (*i.e.*, $\mathbf{x}^*$ and $\mathbf{z}^*$) are referred to as the design variables optimized in the lower level and which are thus fixed in the upper level. In the same way, the constraints denoted with a star exponent (*i.e.*, $\mathbf{h}^*_{layout}(\cdot)$ and $\mathbf{g}^*_{layout}(\cdot)$) are referred to as the equality and inequality constraints handled in the lower level but which must ensure the feasibility of the layout for the list of components defined by $\mathbf{w}$ at the upper level.

6.4.2 General structure of the algorithm

The proposed algorithm consists in solving the nested aforementioned problem formulation which leads to optimizing the conditional variables $\mathbf{w}$ separately from the continuous and discrete variables $\mathbf{x}, \mathbf{z}$. More specifically, the upper stage of the algorithm is responsible for the choice of the components to be integrated into the system, while the lower stage is responsible for the optimization of the layout of the chosen set of components. The two optimization processes can thus be handled using specific methods. The methodology embedded within each of the two stages of the algorithm is introduced below:

- **Lower level:** Component Swarm Optimization based on a Virtual-force system (CSO-VF) algorithm described in Chapter 3 and illustrated in Chapter 4. This algorithm is designed to solve fixed search space OLPs;
- **Upper level:** Bayesian Optimization (BO) used to deal with the conditional variables [Pel+21].

Figure 6.9 illustrates the structure of the bilevel algorithm and both levels are detailed in the following sections.

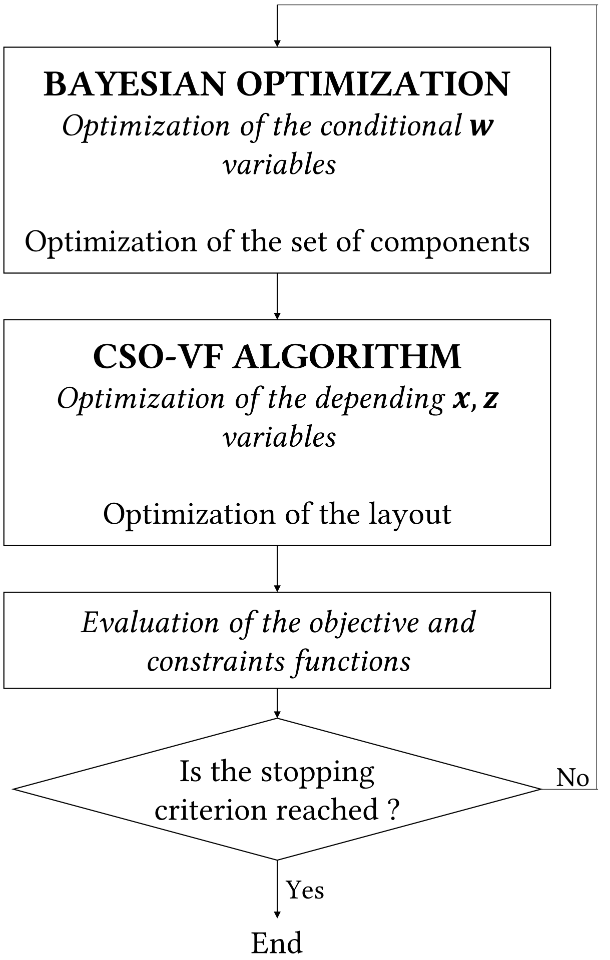


Figure 6.9: Structure of the surrogate assisted-CSO-VF algorithm.

6.4.3 Lower level: CSO-VF algorithm

The lower level optimizes the layout of the set of components that is given by the upper level. The design variables optimized by this level consequently correspond to the $\mathbf{x}$ and $\mathbf{z}$ continuous or discrete design variables vectors defining a layout *i.e.*, the positions of the centers of inertias of the components as well as their orientations for non-cylindrical components. To do so, the CSO-VF algorithm described in Chapter 3 is employed. More precisely, N_{start} instances of the CSO-VF algorithm are run in parallel over N_{it} iterations for a given set of components corresponding to the input of the lower level. The best objective function obtained using the N_{start} CSO-VF instances is the output of this level.

6.4.4 Upper level: Bayesian Optimization for categorical variables

6.4.4.1 General description of the upper level

The goal of this level is to optimize the list of the components chosen among the given catalogue. More precisely, the design variables optimized by this level correspond to the $\mathbf{w}$ vector of conditional categorical design variables which characterizes the choices of components among the possible subsets. Then, if N generic components must be positioned and each of them have n_i possible subsets (*i.e.*, possible subdivisions), then the vector $\mathbf{w}$ of categorical variables is expressed as:

$$\mathbf{w} = \{w_i\}_{i \in \{0,...,N\}} \quad \text{where} \quad w_i \in \{1, ..., n_i\} \tag{6.14}$$

The w_i corresponds to categorical variables as there is no relation of order between the possible choices of subdivisions. In other word, considering the ith component which can be divided among n_i subdivisions, a subdivision k is not greater nor lower than another subdivision $k+1$, for $k \in \{1, ..., n_i - 1\}$.

The goal of the upper level is then to find the best sequence of components that lead to the best possible layout in terms of the objective function considered in the lower level. In order to solve this problem, Bayesian Optimization based on Gaussian Process (GP) surrogate modeling is employed. Indeed, Bayesian Optimization is particularly efficient when it comes to optimizing black-box expensive-to-evaluate fitness functions as described in Chapter 5. In this chapter, a limited budget of function evaluations is considered. Moreover, the fitness function corresponds to the lower level and is expensive-to-evaluate. Bayesian Optimization should hence be an appropriate method [BCf10; Fra18; WHD18].

6.4.4.2 Gaussian Process surrogate modeling

The Bayesian Optimization process involves using a GP to approximate the relationship between inputs and outputs of the black-box function, based on a limited set of observations of the system. In the case of conditional search space OLPs, the goal is thus to approximate the relationship between the set of components and the quality of the optimized layout obtained with the lower level.

Training data set. A training data set $\mathcal{T}$ of N_{DoE} samples $\{\mathbf{w}_j, y_j\}$ with $j \in \{1, ..., N_{DoE}\}$. $\mathcal{T}$ is generated and defined as follows:

$$\mathcal{T} = \{\mathbf{W} = \{\mathbf{w}_1, ..., \mathbf{w}_{N_{DoE}}\} \in F_w, \quad \mathbf{y} = \{y_1, ..., y_{N_{DoE}}\} \in F_y\} \tag{6.15}$$

where $\mathbf{W}$ is the matrix containing the N_{DoE} vectors of conditional variables constituting the training data set (with definition domain F_w), and $\mathbf{y}$ is the vector containing the corresponding function evaluations resulting from CSO-VF algorithm (with definition domain F_y).

Depending on the definition of the catalogue of N components and thus to the available values n_i for the $\mathbf{w}$ vector defined by Equation 6.14 in the problem at hand, thousands of possible components lists might be considered. The problem that is solved by the upper level can thus be highly combinatorial and consequently, the choices of the training data sets' points to evaluate have a critical impact on the GP modeling and hence on the Bayesian Optimization process global performance.

Therefore, the training data set is generated using discrete Latin Hypercube Sampling (LHS) [Tra+22]. Indeed, the LHS technique allows to distribute the values of the N_{DoE} $\mathbf{w}$ vectors over the available search space and might provide better GP process inputs than, for instance, random techniques.

GP surrogate modeling definition. A generic Gaussian Process $\mathcal{Y}(\mathbf{w})$ is defined by its mean function μ and its covariance function also called kernel function k:

$$\mu(\mathbf{w}) = E[\mathcal{Y}(\mathbf{w})] \tag{6.16}$$
$$k(\mathbf{w}, \mathbf{w}') = E[(\mathcal{Y}(\mathbf{w}) - \mu(\mathbf{w}))(\mathcal{Y}(\mathbf{w}') - \mu(\mathbf{w}'))] \tag{6.17}$$

Thus, the GP describing the function $f(\mathbf{w})$ is defined as:

$$f(\mathbf{w}) \sim \mathcal{GP}(\mu(\mathbf{w}), k(\mathbf{w}, \mathbf{w}')) \tag{6.18}$$

In the present algorithm, the GP predicts the objective function value of the lower level y^* for a given components set $\mathbf{w}^*$ using a Gaussian distribution described by the mean and kernel functions, and conditioned by the aforementioned training data set $\mathcal{T}$. In that respect, the kernel function is a core component of the GP surrogate modeling and is defined as an input space dependent parameterized function. In order to model functions taking as argument categorical variables, continuous kernels must be adapted or new kernels must be developed.

6.4.4.3 Kernels for categorical variables

Generally speaking, the kernel is a mathematical function that defines the similarity between pairs of input points following the Reproducible Kernel Hilbert Space (RKHS) formalism. Consequently, it has a significant impact on the performance of the GP (*e.g.*, its prediction capabilities). A large number of kernels can be used for handling categorical variables. In this manuscript, following the recommendations given in [Pel+21], two kernels are considered:

- **Compound Symmetry (CS) kernel** [Rou+20]. The CS kernel function can be defined as follows:

  $$k(w, w') = \sigma_w \exp(-\theta d(w, w')) \tag{6.19}$$

  where σ_w and θ are hyperparameters of the CS kernel and d stands for a meaningful distance in the search space between the two dimensional variables w and w'. Most of the time the Euclidean distance is considered (the corresponding kernel is the well-known Squared Exponential Kernel [RW06]). However, as the considered categorical variables do not have any direct relationship of order, other distances must be used. Two alternative distances are considered:

  - Hamming Distance [ASN14]: the Hamming distance counts the mismatches between two categorical variables vectors. Let $\delta(i, j)$ be the Kronecker delta function:

    $$\delta(i, j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{6.20}$$

    The Hamming distance between two conditional variables w and w' is computed as follows:

    $$d_H(w, w') = 1 - \delta(w, w') \tag{6.21}$$

The corresponding kernel has been used with the Bayesian Optimization based on GP surrogate modeling framework to deal with discrete or categorical variables [HO13a; Zae18; ZH18b]. Considering the above detailed OLP, the Hamming distance computed on two distinct lists of components characterizes the dissimilarity between the chosen subdivisions.

– Distance based on the number of components: another possible metric that can be used in the precise case of the OLP is based on the number of components resulting from the chosen subdivisions. Indeed, the number of components to position in the container characterizes the difficulty to solve the constraints and thus, a distance can be computed from this specificity. If $C(w)$ corresponds to the number of components resulting from a chosen subdivision of a component, then the components distance is computed as:

$$d_C(w, w') = (C(w) - C(w'))^2 \tag{6.22}$$

- **Latent Variable (LV) kernel** [Zha+20]. In the LV kernel, the categorical variables are first transformed into a lower-dimensional space, where they are modeled as continuous latent variables. In other words, the categorical variables levels (*i.e.*, the l reachable values for each categorical variable) are mapped onto a 2-dimensional Euclidean space. This transformation is mathematically defined as:

$$\begin{aligned} \phi(w) : F_w^l &\longrightarrow \mathbb{R}^2 \\ \phi(w = w_m) &= [\theta_{m,1}, \theta_{m,2}]^T, m \in \{1, l\} \end{aligned} \tag{6.23}$$

where $\theta_{1,m}$ and $\theta_{2,m}$ are the hyperparameters corresponding to the continuous coordinates in the 2-dimensional Euclidean space onto which the level m of the categorical variable w is mapped. Then, an equivalent continuous kernel $\tilde{k}$ can be used in order to define a valid kernel on the categorical search space:

$$k(w, w') = \tilde{k}(\phi(w), \phi(w')) = \sigma_w^2 \exp(\|\phi(w) - \phi(w')\|_2^2) \tag{6.24}$$

It must be noted that as the CSO-VF algorithm depends on the initialization, a noisy data modeling of the kernels functions is considered, translated by the definition of a nugget [RW06].

6.4.4.4 Category-wise and level-wise categorical kernels

In the previous paragraphs, the kernels are formulated for one-dimensional categorical vectors. However, nearly all of the considered CSS OLPs depend on multiple categorical variables in order to describe the choice of multiple components subdivisions. In this case, two different approaches can be considered [Pel+20]: level-wise kernels and category-wise kernels. The first approach treats each dimension independantly while the second one defines the kernels on the combinatorial categorical search space. Thus, the appropriate approach must be chosen according to the characteristics of the categorical variables of the problem at hand. In the case of the defined CSS OLPs and without loss of generality, the number of categorical variables as well as the number of available values for each categorical variable might result in thousands of categories *i.e.*, the possible lists of components, which would make the category-wise kernels inoperable. Moreover, the assumption can reasonably be made that the components' subdivisions can be chosen independently of each other. Consequently, the level-wise kernels approach might be generally more suitable for the CSS OLPs considered in this manuscript. The kernels are then defined as:

$$k(\mathbf{w}, \mathbf{w}') = \prod_{d=1}^{N_d} k(w_d, w'_d) \tag{6.25}$$

where N_d is the dimensionality of the problem.

6.4.4.5 Optimization of the kernels hyperparameters

The kernels detailed in Section 6.4.4.3 rely on a certain number of hyperparameters which could influence the value returned by the kernel function, independently from the input data samples. Thus, to ensure that the GP delivers the most precise prediction of the underlying modeled function along with a valid associated error model, it is essential to optimize the value for each hyperparameter of the aforementioned kernels. This procedure is commonly referred to as GP training. Several methods and criteria can be used for the training of the GP: the cross-validation [CW78], the marginal likelihood optimization [RW06] or the restricted likelihood optimization [DO91]. In this work, the marginal likelihood optimization technique is employed. Therefore, the set of kernels hyperparameters referred to as $\boldsymbol{\theta}$ is optimized by maximizing the log marginal likelihood as follows:

$$\max_{\boldsymbol{\theta}} \quad \log p(\mathcal{Y}|\mathbf{W}, \boldsymbol{\theta}) \tag{6.26}$$

This optimization problem is solved in this thesis using the Adam Optimizer [KB]. The Adam Optimizer algorithm is particularly known for its computational efficiency, its little memory requirement and its ability to handle large problems in terms of parameters [HA21; KB; YAJ20].

6.4.4.6 Bayesian Optimization process

The GP model defined in the previous sections is then used as a surrogate for the system, allowing for predictions at new input points (*i.e.*, new components lists) without the need to run the expensive simulation or evaluation of the CSO-VF algorithm. GP is then enriched during the optimization process in order to converge to the optimum. The enrichment process involves an auxiliary optimization process of an infill criterion in order to find the best valuable candidate (*i.e.*, components list) to be evaluated using the CSO-VF algorithm at the lower value of the criterion. Most of the surrogate modeling techniques as well as Bayesian Optimization algorithm frameworks have been developed to model and optimize continuous problems [JMW98]. However, those techniques have recently been adapted to handle mixed continuous-discrete problems. They are reviewed in [BZ17; Swi+14; Mec+01]. At each iteration, the most promising point (*i.e.*, the $\mathbf{w}$ vector minimizing the infill criterion) is evaluated and added to the data set. At the next step, the GP surrogate model is updated with this new added point. This process is repeated until a stopping criterion is met.

Infill Criterion. At each iteration of the Bayesian Optimization process, the algorithm minimizes an acquisition function whose aim is to determine the next promising location in the design search space. Several infill criteria can be chosen [CK20]. Among them, the Expected Improvement (EI) function [JSW98] is one of the most used infill criteria and notably depends on the minimum of the objective value within the data set at the given iteration and the mean and variance of the objective function prediction. EI provides a good balance between exploration and exploitation [JSW98] and is hence used in this thesis.

Feasibility oriented infill criterion. To consider the existence of constraints in the CSS problem at hand and, as a result, generate feasible final designs, the objective function-oriented infill criterion mentioned in the preceding section needs to be integrated with an auxiliary criterion.

Considering the CSS OLPs at hand in this thesis, the main constraint which has to be considered during the BO process is the feasibility or non-feasibility of the layout regarding the chosen input set of components. In other words, the Bayesian Optimization process has to predict if a set of components may or may not lead to a feasible layout in addition to the prediction of the quality of the feasible layout in terms of objective function (*i.e.*, the outputs of the CSO-VF algorithm) in order to infill *a priori* feasible points.

As detailed in Chapter 5, two main ways are used in order to deal with the constraints during the Bayesian Optimization process: the Probability of Feasibility and the Expected Violation techniques. Both techniques rely on the GP prediction of the constraint at unmapped locations. Thus, as the GP must fit heaviside functions with probable flat regions, both techniques might encounter troubles to predict the layout feasibility or unfeasibility of the unmapped sets of components.

Therefore, another technique is employed in this thesis. First of all, the value 1 is assigned to the lists of components which lead to infeasible layouts while the value 0 is assigned to the lists of components leading to feasible layouts. The prediction of the layout feasibility or unfeasibility is then performed using a neural network classifier trained on the data set. The employed neural network classifier is a Multi-layer Perceptron (MLP) algorithm that trains using backpropagation [GB10; He+15; RHW86]. Moreover, the constraint of layout feasibility is integrated to the optimization process of the infill criterion. Thus, the next set of components *i.e.*, at which the the actual objective and constraint functions of the considered problem are evaluated at a given Bayesian Optimization iteration is determined through an auxiliary optimization process mathematically formulated as follows:

$$\begin{aligned} \min_{\mathbf{w}} \quad & EI(\mathbf{w}) \\ \text{w.r.t.} \quad & \mathbf{w} \in F_w \\ \text{s.t.} \quad & \mathbf{h}_{predict,NN}(\mathbf{w}) = 0 \end{aligned} \tag{6.27}$$

where EI is the chosen infill criteria which depends on the objective function as follows:

$$\mathbb{E}[I(\mathbf{w})] = (f_{obj}^{min} - \hat{f}_{obj}(\mathbf{w}))\Phi\left(\frac{f_{obj}^{min} - \hat{f}_{obj}(\mathbf{w})}{\hat{s}((\mathbf{w})}\right) + \hat{s}(\mathbf{w})\phi\left(\frac{f_{obj}^{min} - \hat{f}_{obj}(\mathbf{w})}{\hat{s}((\mathbf{w})}\right) \tag{6.28}$$

In Equation 6.28, f_{obj}^{min} is the minimum feasible value present within the data set at the given Bayesian Optimization iteration, $\hat{f}_{obj}(\mathbf{w})$ and $\hat{s}(\mathbf{w})$ are the mean and standard deviation of the objective function prediction, $\Phi(\cdot)$ is the cumulative distribution function of a normal distribution and $\phi(\cdot)$ is the probability function of a normal distribution.

Moreover, in Equation 6.27, $\mathbf{h}_{predict,NN}$ is the equality constraint corresponding to the prediction of the neural network classifier regarding the feasibility of the unmapped set of components $\mathbf{w}$, and trained on the current mapped locations. It is related to the layout constraints defined in Equations 6.13 as follows:

$$\mathbf{h}_{predict,NN}(\mathbf{w}) = \begin{cases} 0, & \text{if } \mathbf{h}^*_{layout}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) = 0 \text{ and } \mathbf{g}^*_{layout}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \leq 0 \\ 1, & \text{otherwise} \end{cases} \tag{6.29}$$

with the same formalism as in Equations 6.13, the star exponent characterizes the design variables and constraints functions handled in the lower level. The layout constraints $\mathbf{h}^*_{layout}(\cdot)$ and $\mathbf{g}^*_{layout}(\cdot)$ ensure the feasibility of the layout for the list of components $\mathbf{w}$ using the predictions of the neural network. Therefore, the formulation of the Bayesian Optimization problem written with Equations 6.27 is a translation of the upper stage of the nested formulation (*i.e.*, Equations 6.13), specified with the Bayesian Optimization features.

Infill criterion optimization. To determine the location for infilling new data samples in each iteration of the Bayesian optimization algorithm, an auxiliary optimization process is needed, as formulated in the previous section. It is worth noting that the evaluation of acquisition functions may require significantly less time when compared to the computation of the main problem's objective and constraint functions. Therefore, common optimization algorithms can be employed for this purpose.

In the case of CSS OLPs, the acquisition function must be optimized in a categorical search space. The chosen technique must be able to deal with such category of design variables.

In this work, the infill criterion is optimized using a Genetic Algorithm (GA) able to deal with categorical variables as well as employing constraint dominance selection criterion, similar to the one described by Stelmack [SNB98]. The choice of specific parameters for this GA, such as mutation and crossover probabilities, as well as the number of generations and population size, varies depending on the test cases and is adapted to the size and characteristics of the specific problem being addressed.

6.4.5 Algorithm framework

Then, the surrogate assisted-CSO-VF algorithm process is as follows: a training data set of N_{DoE} lists of components is generated at the upper level and evaluated using the lower level. The Gaussian Process (GP) surrogate model is created using the data set. The neural network classifier is trained on the mapped location in order to predict the feasibility of an unmapped set of components. The Bayesian Optimization process optimizes the infill criteria and the output point corresponding to the most promising feasible list of components is evaluated at the lower level by performing the CSO-VF algorithm on the list of components. This point is added to the data set which is then used at next iteration to create an enriched GP surrogate model. The upper and lower level are hence performed sequentially until a stopping criterion corresponding to the maximum number of iterations is met.

Figure 6.10 illustrates the surrogate assisted-CSO-VF algorithm framework.

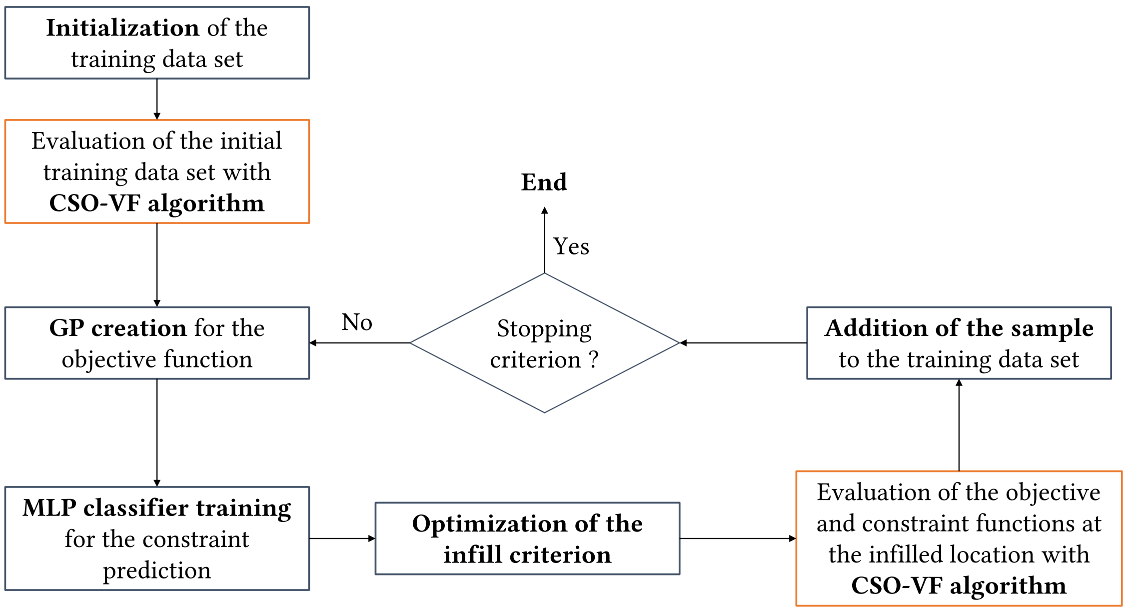


Figure 6.10: Surrogate assisted-CSO-VF algorithm framework. The steps related to the upper and lower level are respectively highlighted in blue and orange.

6.5 Conclusion

In this chapter, two algorithms are developed in order to deal with conditional search space optimal layout problems. These latter consist in optimizing the list of the components to include in the system along with their layout. The choice of components introduces categorical design variables to optimize that are responsible for the number and types of the continuous and discrete variables.

The first approach is based on GA enhanced with a hidden-variable mechanism (HVM for GA). This HVM for GA modifies the implementation of the chromosomes in order to reveal or hide some of the genes that are respectively taken or not taken into account for the objective or constraint functions evaluations. Thus, this method allows to tackle all categorical, continuous and discrete design variables in a single optimization process.

The second approach consists in a bilevel surrogate assisted-Component Swarm Optimization based on Virtual-Force (CSO-VF) algorithm. It combines the strength of the CSO-VF algorithm developed and evaluated in Part I, and a discrete Bayesian Optimization algorithm framework purposely adapted to tackle the categorical aspect of this problem. This second approach thus consists in two nested optimization processes dealing respectively with the optimization of the set of components (*i.e.*, the categorical conditional variables) and with the optimization of the corresponding layout (*i.e.*, the continuous and discrete variables).

These two algorithms are configured, assessed and compared in Chapter 7 by their application to conditional search space single-container satellite module layout problems.

Chapter 7

Applications of Hidden-variable Mechanism for Genetic Algorithm and Surrogate Assisted-CSO-VF Algorithm to Single-container Satellite Optimal Layout Problems

Contents

Chapter contributions

- Extension of fixed search space satellite layout problems to conditional search space optimal layout problems.
- Assessment of the performance of the hidden-variable mechanism for GA and the surrogate assisted-CSO-VF algorithm on the satellite application cases.
- Comparison of the two approaches in terms of global performance.

7.1 Introduction

In this chapter, the performance of the algorithms developed in Chapter 6 are assessed on a a conditional search space optimal layout application case. The optimal layout of the simplified model of the international commercial communication satellite module (INTELSAT-III) is considered as a representative application case of optimal layout problems and without loss of generality [CXW18; CZa19; Ten+09; ZTS08]. The fixed search space formulation of the problem detailed in the Chapter 4 is extended to conditional search space problems. As detailed in Chapter 6, a catalogue of components is defined and the algorithms have to determine the optimal set of components among this catalogue as well as the corresponding optimal layout. The hidden-variable mechanism for Genetic Algorithm and the surrogate assisted-Component Swarm Optimization based on a Virtual-force system algorithm are first configured and used to solve the benchmark problem. Their global performance are subsequently compared.

The rest of this chapter is organized as follows: Section 1 is devoted to the formulation of conditional search space single-container satellite module layout problems. The hidden-variable mechanism for Genetic Algorithm and the surrogate assisted-CSO-VF algorithms are respectively configured and assessed in Section 2 and 3. Finally, a comparison of the global performance of the two methods is conducted in Section 4.

7.2 Conditional search space single-container satellite module layout problems

In this section, conditional search space (CSS) single-container optimal layout problems are specified in the case of satellite module layout problems. The problem at hand is extended from the fixed search space (FSS) single-container satellite layout problem defined in Chapter 4. It involves positioning N generic components, each having n_i ($i \in \{1, ..., N\}$) possible subdivisions, within a container in order to minimize the global inertia of the module. Several constraints must be satisfied. These constraints can be either geometrical (*e.g.*, overlapping constraints, balancing constraints) or functional (*e.g.*, constraints of functional compatibility between components for instance due to electromagnetic or heat radiations threshold within the container).

7.2.1 Geometry of the container and the components

Container. Similarly to the single-container configuration of the FSS satellite layout problem formulation, the container is a one-sided bearing plate defined by its outer radius R_{out} and two exclusion zones which are defined as follows:

- A central circular bus defined by its radius R_{in} and centered at the geometrical center of the plate;
- A rectangular bus defined by its dimensions (L_{bus}, l_{bus}) and its positions on the plate in the cylindrical system of coordinates (r_{bus}, θ_{bus}).

Components. All components are defined using their dimensions and masses. They are considered as rigid bodies of homogeneous density and are located using the position of their centers of inertia. They are of three types:

- Fuel components;
- Energy components;
- Other diverse components (*i.e.*, components with no functional constraints).

Each component is identified by:

- The index of its corresponding generic component, i, for $i \in \{1, ..., N\}$;
- The index of the subdivision it belongs to, j, for $j \in \{1, ..., n_i\}$;
- The index of the component from the subdivision, k, for $k \in \{1, ..., n_{i,j}^{subdiv}\}$. Indeed, the number of components $n_{i,j}^{subdiv}$ in each subdivisions depends on how is subdivided the corresponding generic components and thus depends on the indices i and j.

The geometrical attributes of the kth component of the jth subdivision of generic component i, *i.e.*, component $\{i, j, k\}$, are thus written: $R_i^{j,k}$ for the radius in case of a cylinder, $a_i^{j,k}, b_i^{j,k}$ for the rectangular base of a cuboid, $h_i^{j,k}$ for its height and $m_i^{j,k}$ for its mass.

Figure 7.1 illustrates the definition of the subdivisions of one generic cylinder component i with three possible subdivisions.

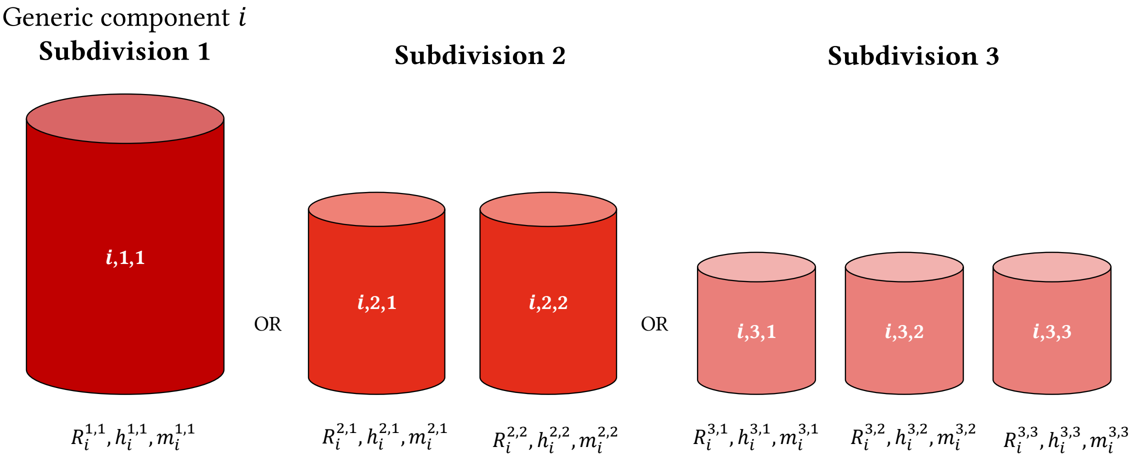


Figure 7.1: Definition of the subdivisions of the components on an example.

In this chapter, three assumptions are made, for the sake of simplicity and without loss of generality:

- The geometry of each generic component (*i.e.*, cylinder or cuboid) remains identical for all the components from its subdivisions;
- The total area of the components of each subdivisions of a same generic component is conserved. It is mathematically translated as follows:

$$\sum_{k=1}^{n_{i,j}^{subdiv}} A_i^{j,k} = \sum_{k=1}^{n_{i,j+1}^{subdiv}} A_i^{j+1,k}, \quad \forall i \in \{1, ..., N\}, \quad \text{for } j \in \{1, ..., n_i - 1\} \tag{7.1}$$

  where $A_i^{j,k}$ is the area of component $\{i, j, k\}$;
- The total mass of the components of each subdivisions of a same generic component is conserved. It is mathematically translated as follows:

$$\sum_{k=1}^{n_{i,j}^{subdiv}} m_i^{j,k} = \sum_{k=1}^{n_{i,j+1}^{subdiv}} m_i^{j+1,k}, \quad \text{for } i \in \{1, ..., N\}, \quad \text{for } j \in \{1, ..., n_i - 1\} \tag{7.2}$$

The configuration designed for the application case consists in 12 generic components that can be divided so that a maximum of 28 components are positioned in the container. Overall, it corresponds to **3888 possible different lists of components**. The details of the generic components and their subdivisions used in this chapter are reported in Table A.4 in Appendix A.

Moreover, as for the fixed search space satellite module layout problems, the dimensions of the components are modified in order to study four increasing different occupation rates of the container, from 30% to 60%. The higher the occupation rate of the container, the more difficult it is to satisfy the constraints.

7.2.2 Design variables

In the conditional search space single-container satellite optimal layout configuration, the design variables are:

- The chosen subdivisions of the generic components, considered as conditional categorical variables $\mathbf{w} = \{w_i\}$ where $i \in \{1, ..., N\}$ and $w_i \in \{1, ..., n_i\}$;
- The positions of the centers of inertia of the chosen components, considered as continuous variables, $\mathbf{x}_p$. The length of vector $\mathbf{x}_p$ depends on the conditional variables and is denoted $n_{x_p}(\mathbf{w})$;
- The orientations of the chosen cuboid components considered as continuous variables, $\mathbf{x}_\alpha$. The length of vector $\mathbf{x}_\alpha$ depends on the conditional variables and is denoted $n_{x_\alpha}(\mathbf{w})$.

It must be noted that the $\mathbf{x}$ vector of continuous variables introduced in Chapter 6 corresponds to $\mathbf{x} = \{\mathbf{x}_p, \mathbf{x}_\alpha\}$. In addition, the layout variables are considered continuous here, but the overall problem formulation is easily adaptable if discrete variables are introduced (*e.g.*, if the orientation of non-cylinder components is considered discrete).

Thus, the number of variables to optimize depends on the values of the conditional variables w_i, $i \in \{1, ..., N\}$. The maximum number of design variables n_{var} used to define a layout is defined as:

$$n_{var} = N + \sum_{i \in CYL} 2 \max_j (n_{ij}^{subdiv}) + \sum_{i \in CUB} 3 \max_j (n_{ij}^{subdiv}) \tag{7.3}$$

where CYL and CUB respectively referred to as the i indices of cylinder and cuboid components. In other words, for $i \in \{1, ..., N\}$, $i \in CYL$ if component i is a cylinder or $i \in CUB$ if component i is a cuboid.

7.2.3 Objective and constraints functions

Objective function. The objective function to minimize is the global inertia of the module I_{tot} which is mathematically formulated in Chapter 4. In this chapter, the inertia function takes as arguments the positions and orientations of the components selected by the conditional variables $\mathbf{w}$.

Constraints functions. The constraints functions are identical to the ones listed and mathematically defined in Chapter 4. They are briefly summarized:

- Overlapping between the components, $\mathbf{h}_{overlap}^{Comp}$;
- Belonging of the components within the container, $\mathbf{h}_{overlap}^{Cont}$;
- Overlapping between the components and the exclusion zones, $\mathbf{h}_{overlap}^{E}$;
- Balancing constraint, g_{CG};
- Functional constraints of distance between fuel and energy components, $\mathbf{h}_{functional}$.

In this chapter, the constraints functions take as arguments the positions and orientations of the components selected by the conditional variables $\mathbf{w}$.

7.2.4 Mathematical formulations

As in Chapter 6, two cases are considered:

Case 1. The conditional variables $\mathbf{w}$ responsible for the choices of the components are optimized simultaneously with the variables $\mathbf{x}_p, \mathbf{x}_\alpha$ related to the corresponding layout, in a single optimization process. Thus, it is formulated mathematically as follows:

$$\begin{aligned}
\min_{\mathbf{x}_p,\mathbf{x}_\alpha,\mathbf{w}} \quad & I_{tot}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{x}_p \in F_{\mathbf{x}_p} \subseteq \mathbb{R}^{n_{\mathbf{x}_p}(\mathbf{w})}, \mathbf{x}_\alpha \in F_{\mathbf{x}_\alpha} \subseteq \mathbb{R}^{n_{\mathbf{x}_\alpha}(\mathbf{w})}, \mathbf{w} \in F_{\mathbf{w}} \subseteq \mathbb{N}^{n_{\mathbf{w}}} \\
\text{s.t.} \quad & \mathbf{h}_{overlap}^{Comp}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{Cont}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{E}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \mathbf{h}_{functional}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& g_{CG}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) \leq 0
\end{aligned} \tag{7.4}$$

This formulation is used for the hidden-variable mechanim for Genetic Algorithm.

Case 2. A nested formulation is proposed in order to optimize the conditional variables and the depending variables in two sequential optimization processes:

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & I_{tot}(\mathbf{x}_p^*, \mathbf{x}_\alpha{}^*, \mathbf{w}) \\
\text{s.t.} \quad & \mathbf{h}_{overlap}^{Comp}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{Cont\ *}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{E\ *}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{w}) = 0 \\
& \mathbf{h}_{functional}^{*}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{w}) = 0 \\
& g_{CG}^{*}(\mathbf{x}_p^*, \mathbf{x}_\alpha^*, \mathbf{w}) \leq 0 \\
\text{w.r.t.} \quad & \mathbf{w} \in F_w \subseteq \mathbb{N}^{n_w} \\
& \{\mathbf{x}_p^*, \mathbf{x}_\alpha{}^*\} = \text{argmin } I_{tot}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) \\
& \text{w.r.t.} \quad \mathbf{x}_p \in F_{\mathbf{x}_p} \subseteq \mathbb{R}^{n_{\mathbf{x}_p}(\mathbf{w})}, \mathbf{x}_\alpha \in F_{\mathbf{x}_\alpha} \subseteq \mathbb{R}^{n_{\mathbf{x}_\alpha}(\mathbf{w})} \\
& \qquad\quad \mathbf{h}_{overlap}^{Cont}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \qquad\quad \mathbf{h}_{overlap}^{E}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \qquad\quad \mathbf{h}_{functional}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) = 0 \\
& \qquad\quad g_{CG}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{w}) \leq 0
\end{aligned}
\tag{7.5}
$$

The design variables denoted with a star exponent ($e.g.$, $\mathbf{x}^*$) referred to as the design variables optimized in the lower level and which are thus fixed in the upper level. In the same way, the constraints denoted with a star exponent ($e.g.$, $\mathbf{h}^*(\cdot)$) referred to as the constraints handled in the lower level but which must ensure the feasibility of the layout for the list of components defined by $\mathbf{w}$ at the upper level.

This formulation is used for the surrogate assisted-CSO-VF algorithm.

7.3 Configuration of the hidden-variable mechanism for Genetic Algorithm

The hidden-variables mechanism for Genetic Algorithm (HVM for GA) addresses the first mathematical formulation of the problem given by Equations 7.4. The following sections aims to configure and analyze the global performance of the HVM for GA using the problem described in Section 7.2 and for three occupation rates: 30%, 50% and 70%.

7.3.1 Configuration of the hidden-variable mechanism for genetic algorithm

In order to configure the HVM for GA, the operators detailed in Chapter 6 are tested and compared thanks to a test campaign.

Design of a test campaign. Given the combinatorial dimension of the problem, operators that promote diversity in order to prevent premature convergence are privileged. The considered operators which have been described in Chapter 6 are briefly summarized below:

- For selection and constraint handling, a tournament with either 5 or 15 individuals (T5 or T15) as well as Constraint Dominance (CD) [Deb00] or Stochastic Ranking (SR) [RY00] are considered;
- For the crossover operator on chromosomes either Simulated Binary Crossover (SBX) [DA95] or Uniform Crossover (UX) [Dan+16] is used. The crossover on tags is described in Chapter 6 and is not recalled here for the sake of conciseness;
- The mutation on chromosomes is Polynomial Mutation (PM) [Tal09] or Uniform Mutation (UM) [Tal09]. In the same way, the mutation on tags is described in Chapter 6 and is not recalled here for the sake of conciseness;
- The replacement scheme is either the non-dominated truncating (NDT) [Deb+02] or the deterministic crowding (DC) [Mah92; Mah94].

In order to analyze in depth the performance of the algorithm with the previously described operators and to choose an operator configuration, two Taguchi's experiment plans are adopted [Roy10]:

- In the first set of Taguchi's experiments, summarized in Table C.2 in Appendix C, a focus is made on the crossover and mutation operators;
- In the second set of Taguchi's experiments, summarized in Table C.3 in Appendix C, a focus is made on the selection and the replacement operators.

The hyperparameters of the algorithm are set for all the schemes using a parametric study. The Taguchi's plans are run for the tag implementation of the chromosomes detailed in Chapter 6. The aim is to find the best combination of operators to get the best balance between exploration and exploitation using analysis criteria defined in Appendix C.

Best obtained configuration. The following sequence of operators as well as hyperparameters are selected after the test campaigns:

- **Constraint handling**: Stochastic Ranking with probability $P_f = 0.45$;
- **Selection**: Tournament with 15 individuals;
- **Crossover design variables**: Simulated Binary Crossover (SBX), with probability $P_c^g = 0.9$;
- **Crossover dimensional variables** (for GA-DV): SBX with probability $P_c^{DV} = 0.9$;
- **Crossover tags** (for GA-tags): 1-point, with probability $P_c^t = 0.9$;
- **Mutation**: Polynomial Mutation (PM) with probability $P_m = 0.2$;
- **Replacement**: Non-Dominated Truncating (NDT).

7.3.2 Global performance of the hidden-variable mechanism for genetic algorithm

The aforementioned operators and hyperparameters configuration are used to study the global performance of the HVM for GA for the two chromosomes implementations detailed in Chapter 6:

- The implementation of the conditional variables as design variables (HVM-DV);
- The implementation of the conditional variables as tag vectors (HVM-TAG).

The HVM-DV and HVM-TAG methods are run 50 times for four different occupation rates: 30%, 40%, 50%, 60%. For each occupation rate, the algorithms are randomly initialized, even though those initializations are the same between the two methods so that they are comparable. The computational budget is set to 5×10^6 objective function evaluations, distributed as follows: 500 individuals in the populations and 10,000 generations.

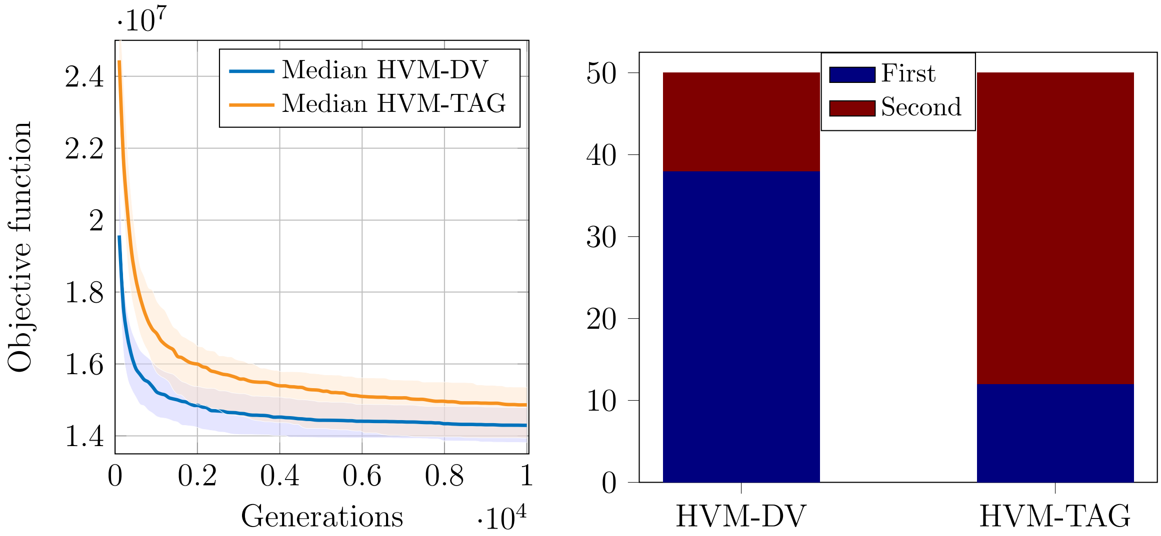
In order to compare the performance provided by the HVM for GA with both proposed conditional variables implementations, the following metrics are adopted:

- **Median of convergence curves:** for each occupation rate, Figures 7.2a, 7.5a, 7.8a and 7.11a show the median of convergence curves of the 50 simulations for the HVM-DV and HVM-TAG implementations.
- **Best run:** for each occupation rate, Figures 7.2a, 7.5a, 7.8a and 7.11a also shows the convergence curves of the simulation leading to the best layout *i.e.*, the smallest final inertia, for each algorithm.
- **Ranks of the algorithms:** Figures 7.2b, 7.5b, 7.8b and 7.11b indicate for each occupation rate the number of times each algorithm ranked first or second in terms of final objective function value for the 50 simulations. This metric characterizes the convergence accuracy of both algorithms.
- **Iterations to best median (mean):** for each occupation rate, Figures 7.3, 7.6, 7.9 and 7.12 show the mean of iterations needed by each median convergence curve to reach 25%, 20%, 15%, 10% and 5% of the best final obtained median. This metric characterizes both the convergence speed and the relative margin between each configuration and the best one in terms of median convergence curve.
- **Successful runs to target:** for each occupation rate, Figures 7.4, 7.7, 7.10 and 7.13 show the number of simulations of each configurations (among the 50) which manage to reach 25%, 15%, 10%, 5%, 3% and 1% of the best obtained median. This metric characterizes the success rate and the robustness of each configuration.
- **Explored configurations:** Figure 7.14 shows the number of reached configurations *i.e.*, lists of components, by both algorithms and the four occupation

rates. This metric characterizes their ability to explore the conditional search space.

Table 7.1 sums up the numerical results obtained for each chromosome implementations of the HVM for GA. The metrics are:

- The success rate *i.e.*, the number of simulations providing with a feasible solution;
- The median of the final objective function values;
- The interquartile range of the final objective function values;
- The best objective function values over the 50 randomly initialized repetitions;
- The mean of the iteration for which a feasible solution is reached for the first time.



(a) Medians, $OR = 30\%$

(b) Best run, $OR = 30\%$

Figure 7.2: Convergence curves and ranks for the 30% occupation rate.

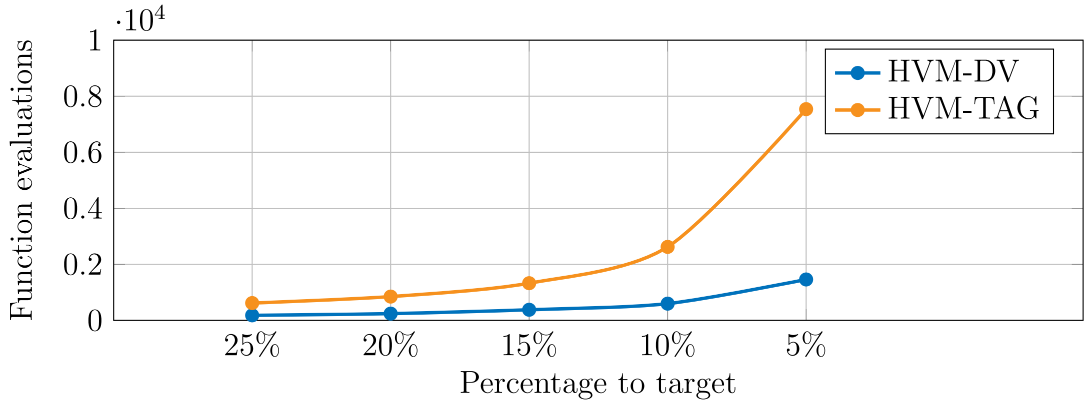


Figure 7.3: Function evaluations to percentages of the median target, $OR = 30\%$.

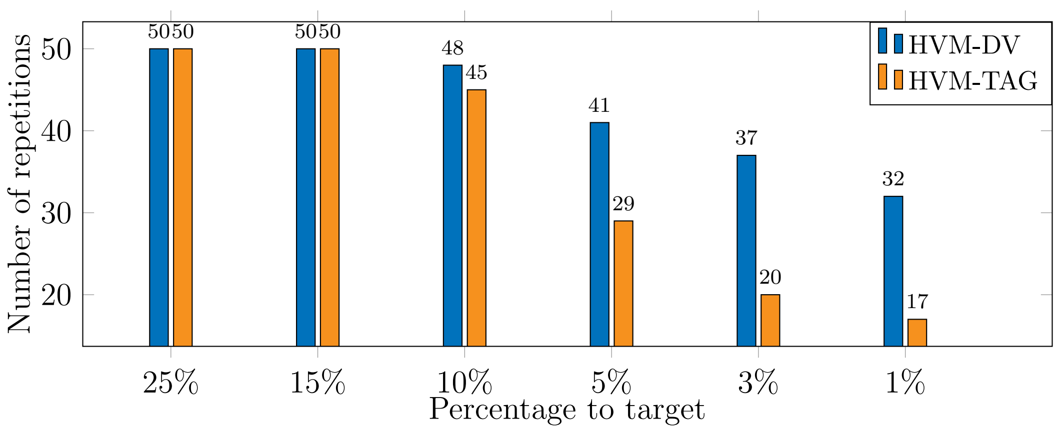
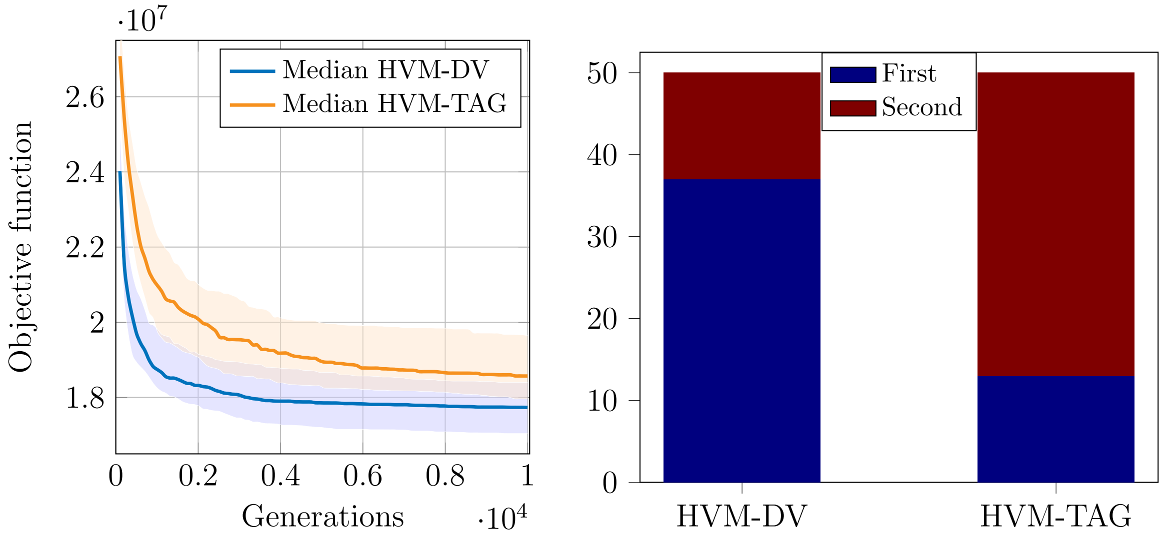


Figure 7.4: Success rate to percentages of the median target, $OR = 30\%$.



(a) Medians and best, $OR = 40\%$

(b) Ranks, $OR = 40\%$

Figure 7.5: Convergence curves and ranks for the 30% occupation rate.

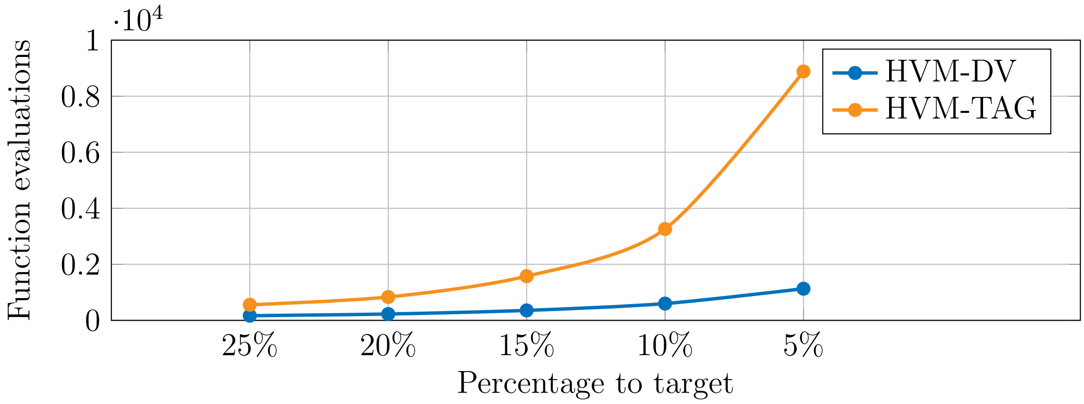


Figure 7.6: Function evaluations to percentages of the median target, $OR = 40\%$.

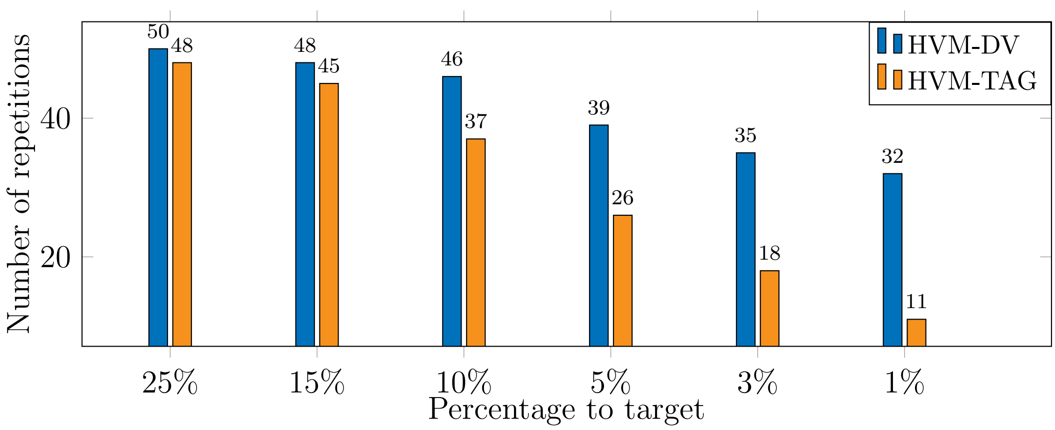
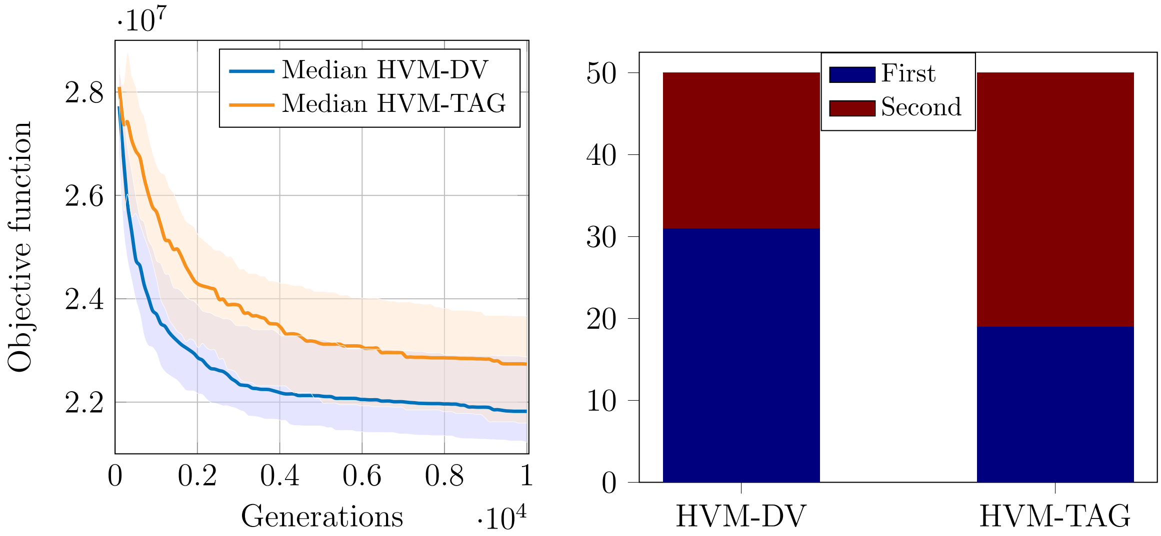


Figure 7.7: Success rate to percentages of the median target, $OR = 40\%$.



(a) Medians and best, $OR = 50\%$

(b) Ranks, $OR = 50\%$

Figure 7.8: Convergence curves and ranks for the 50% occupation rate.

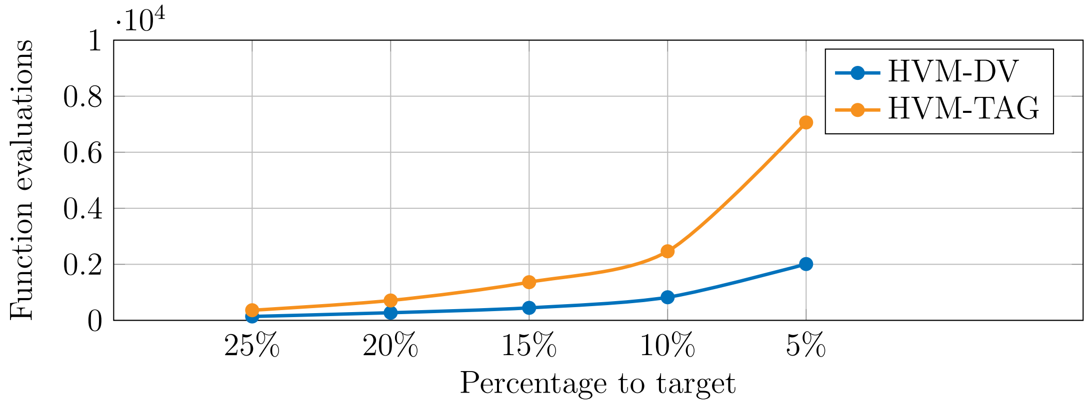


Figure 7.9: Function evaluations to percentages of the median target, $OR = 50\%$.

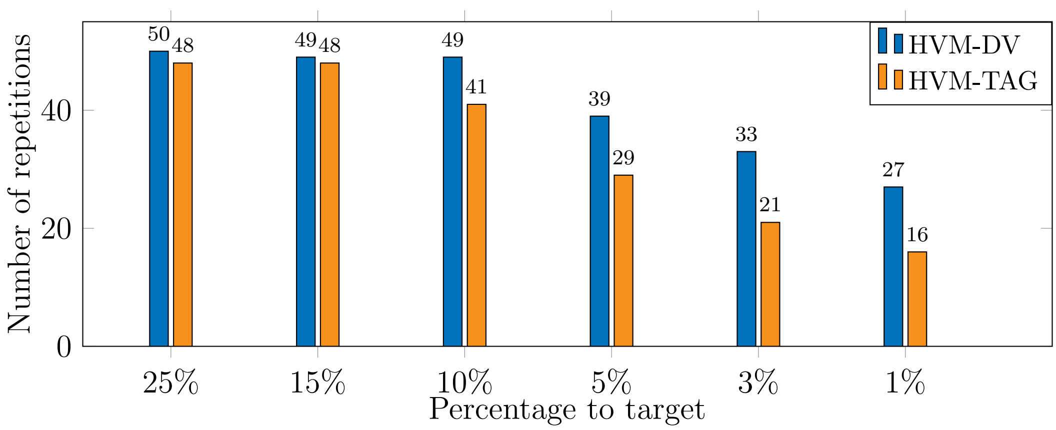
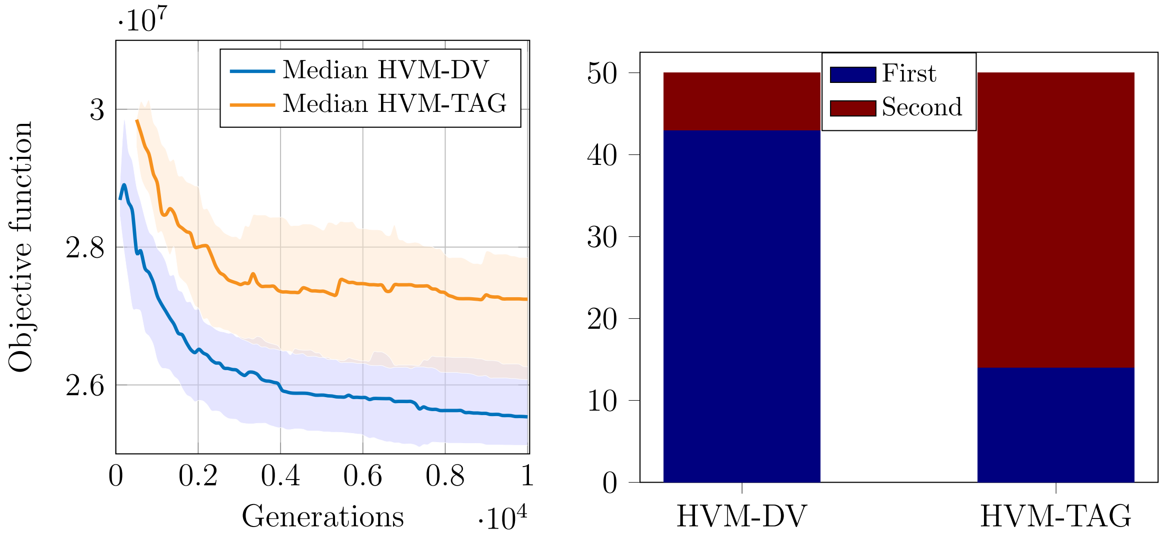


Figure 7.10: Success rate to percentages of the median target, $OR = 50\%$.



(a) Medians and best, $OR = 60\%$

(b) Ranks, $OR = 60\%$

Figure 7.11: Convergence curves and ranks for the 60% occupation rate.

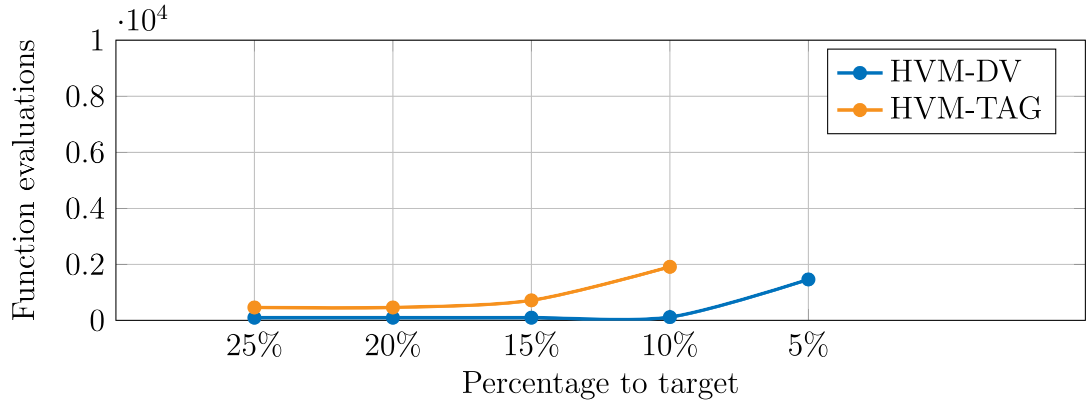


Figure 7.12: Function evaluations to percentages of the median target, $OR = 60\%$.

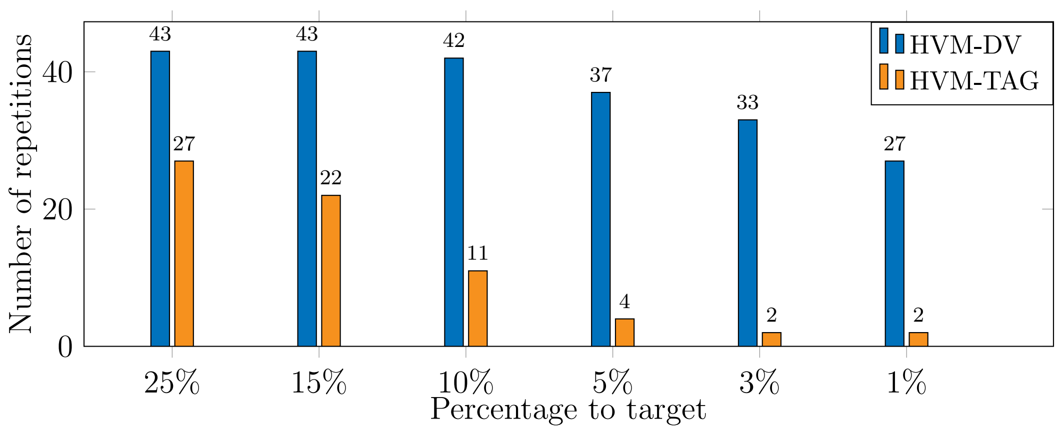


Figure 7.13: Success rate to percentages of the median target, $OR = 60\%$.

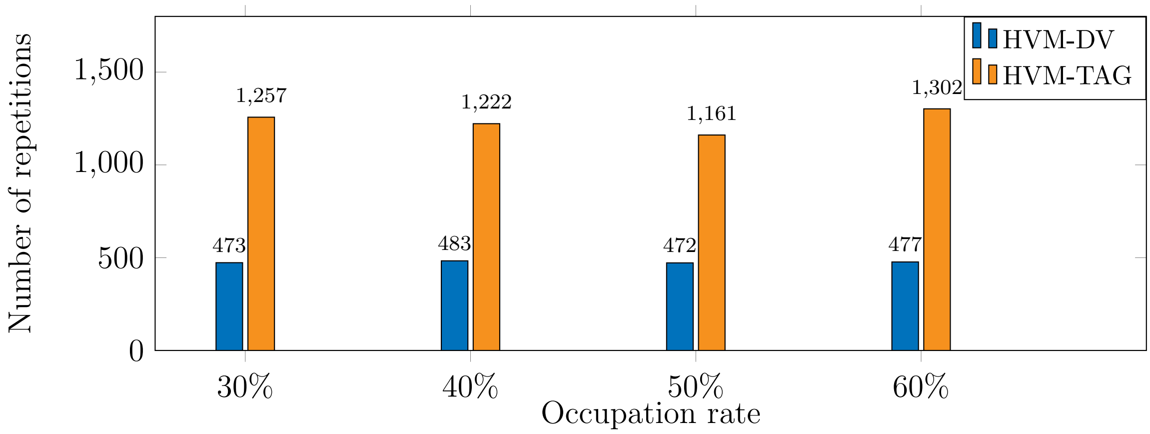


Figure 7.14: Number of configurations reached by both algorithms.

	OR	HVM-DV	HVM-TAG
Success rate	30%	**50**	**50**
	40%	**50**	**50**
	50%	**50**	49
	60%	**43**	35
Final objective (median)	30%	**1.429e7**	1.486e7
	40%	**1.773e7**	1.857e7
	50%	**2.182e7**	2.273e7
	60%	**2.554e7**	2.724e7
Final IQR	30%	**9.190e5**	1.384e6
	40%	**1.327e6**	1.593e6
	50%	**1.646e6**	2.073e6
	60%	**1.140e6**	1.856e6
Best layout	30%	**1.277e7**	1.315e7
	40%	1.626e7	**1.609e7**
	50%	**2.014e7**	2.028e7
	60%	**2.235e7**	2.439e7
First iteration leading to feasible solution (mean)	30%	15.10	43.8
	40%	29.48	89.24
	50%	87.48	352.53
	60%	1247.2	2590.6

Table 7.1: Numerical results for the two implementation of the HVM for GA. Bold values indicate best algorithm.

First of all, Figures 7.2a to 7.14 and Table 7.1 highlight that both methods allow to find feasible solutions to the conditional search space satellite layout problem. As a matter of fact, an appropriately set up Genetic Algorithm enhanced with the proposed mechanism provides a layout of the satellite module among the 3888 possible configurations of components, and manages to solve the constraints for all occupation rates.

Success rate. Table 7.1 shows that until the 50% occupation rate, almost 100% of the runs find a feasible solution from random initializations. However, for the highest occupation rate, respectively 14% and 30% of the runs do not provide any feasible solution for the HVM-DV and HVM-TAG approaches. Indeed, the more the occupation rate of the container increases, the more the constraints are difficult to solve.

Convergence accuracy. The HVM-DV method reaches a better objective function median for all the occupation rates. Indeed, the HVM-DV algorithm allows to improve the final median by respectively 3.8%, 4.5%, 4.0% and 6.2% in comparison to the HVM-TAG configuration for increasing occupation rates. Moreover, regardless of the occupation rate, the HVM-DV algorithm almost always provides the best final inertia (an exception occurs for the 40% occupation rate).

Figure 7.2b, 7.5b, 7.8b and 7.11b rank the values of the final objective function obtained by each algorithm over the 50 instances, for each occupation rate. It is observed that the HVM-DV approach ranks first 31 to 43 times over 50, for all the occupation rates showing better convergence accuracy.

Convergence speed. The third metric shows how many generations are needed in order to reach certain percentages of the best final median obtained (*i.e.*, the final median of the HVM-DV algorithm for all the occupation rates). It is observed that the HVM-DV approach always reaches smaller percentage of the median target in less generations than the HVM-TAG method. Moreover, the HVM-TAG method does not reach 5% of the median target for the 60% occupation. This last observation is consistent with the relative differences between final median objective function values calculated in the previous paragraph. It can be concluded that the HVM-DV method has a higher convergence speed toward the optimum than the HVM-TAG method.

Robustness. For each occupation rate, Figures 7.4, 7.7, 7.10 and 7.13 show the number of runs among the 50 repetitions that reach a certain percentages of the best final median value *i.e.*, the target value. For all the occupation rates, the HVM-DV approach has more runs reaching smaller percentage of the target value in comparion to the HVM-TAG method. It can be concluded that the HVM-DV algorithm has a better robustness with respect the initialization because it leads to a larger number of repetitions reaching the target value than the HVM-TAG method.

Moreover, the final interquartile ranges (IQR) obtained for both HVM chromosome implementations and reported in Table 7.1 allow to quantify the dispersion of the final results in terms of objective function values. The HVM-DV algorithm allows to improve the final IQR by respectively 35.6%, 16.7%, 20.6% and 38.6% in comparison to the HVM-TAG configuration for increasing occupation rates. Overall, it can be concluded that the HVM-DV has a better robustness to initialization in terms of objective function.

First feasible solution. The generations providing the first feasible solution are reported in Table 7.1. First of all, it is observed that it takes more and more time for both algorithms to find a first feasible solution as the occupation rate raises due to the increasing difficulty to solve the constraints by GA. Moreover, the HVM-DV algorithm always find a feasible solution in less generations than the HVM-TAG algorithm.

Exploration of the conditional search space. Figure 7.14 shows that among all individuals of their populations, the HVM-TAG methods explores approximately two to three times more configurations of the conditional search space (*i.e.*, lists of components) than the HVM-DV method. Those results balance the general conclusions drawn in the previous paragraphs. Indeed, even if a better convergence accuracy, speed and robustness to initialization trends were highlighted in favor of the HVM-DV method, the better exploration capacities of the HVM-TAG method must be underlined. It can be explained by the presence of evolutionary operators dedicated to the tag vector responsible for the choices of configurations. On the contrary, the conditional variables of the HVM-DV methods undergo evolutionary operations as part of the chromosome. However, they represent only 9 genes over 129 which may explain the lack of diversity observed in the conditional search space. This may also explain the convergence accuracy and speed lower performance of the HVM-TAG method which explore more in depth the conditional search space.

To conclude, the performance of the HVM for GA method has been assessed using its application to the proposed conditional search space single-container satellite layout problem. The various metrics used in the previous section allow to draw the conclusion that despite the ability of the HVM-TAG chromosome implementation to explore the conditional search space, the HVM-DV implementation provides better performance in terms of convergence accuracy, convergence speed, robustness to initialization and speed in finding feasible solutions, with respect to the given computational budget.

7.4 Configuration of the surrogate assisted-CSO-VF algorithm

The surrogate assisted-CSO-VF algorithm (BO+CSO-VF) addresses the second mathematical formulation of the problem given by Equations 7.5. The following sections aims to configure and assessed the BO+CSO-VF algorithm. It is compared with the HVM for GA in Section 7.5.

7.4.1 Performance analysis on a toy case

As the conditional search space single-container satellite layout problem defined in Section 7.2 has 3888 possible lists of components, a toy case with a smaller number of categories is first designed in order to better understand the different component configurations and to assess the functioning of the surrogate assisted-CSO-VF algorithm.

7.4.1.1 Definition of the toy case

The toy case is defined as follows.

Container. The container is identical as the container defined in Section 7.2 *i.e.*, a one-sided bearing plate with the two exclusion zones.

Components. Three generic components and their respective subdivisions are defined:

- A fuel cylinder component, which can be subdivided in one to four smaller components;
- An energy cuboid component, which can be subdivided in one to four smaller components;
- A diverse cylinder component, which can be subdivided in one to four smaller components.

Thus, 64 different lists of components can be positioned in the container. Figure 7.15 illustrates the components that can be chosen to be part of the layout.

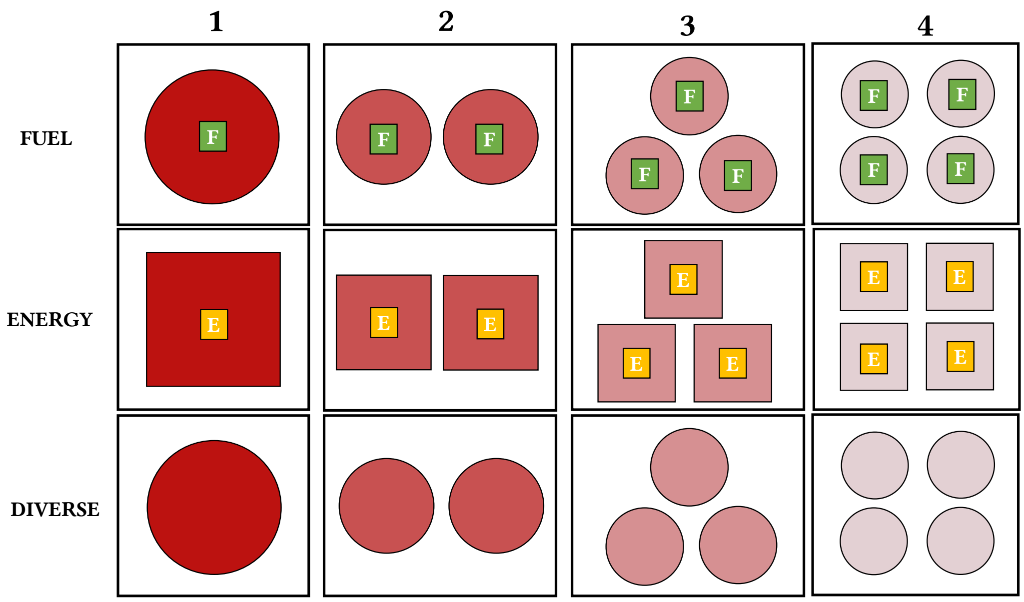


Figure 7.15: Toy case components definition. The redder the components, the heavier their corresponding mass.

Objective and constraint functions. The objective and constraints functions remain identical than the ones detailed in Section 7.2.

7.4.1.2 Toy case analysis

As the conditional search space is composed of 64 lists of components, the layout of all possible lists of components can be optimized using the CSO-VF algorithm. Multistart is employed with 5 instances of the algorithm run over 3000 iterations. The hyperparameters are set using a parametric study.

The goal of this section is to analyze the final obtained inertias (the output), with respect to the 64 lists of components (the input). Figure 7.16 shows the relation between the output and the input with color maps. The final inertia of each subdivision is shown thanks to a blue colorbar (the lighter is the blue, the smaller is the inertia) with respect to the number of components of each type. In other words, each colormap corresponds to a fixed number of diverse components(*i.e.*, 1 to 4 components corresponding to the subdivisions of the generic diverse component). Subsequently, on each colormap, the x and y axes respectively correspond to the number of energy and fuel components. Moreover, some optimal layouts are shown.

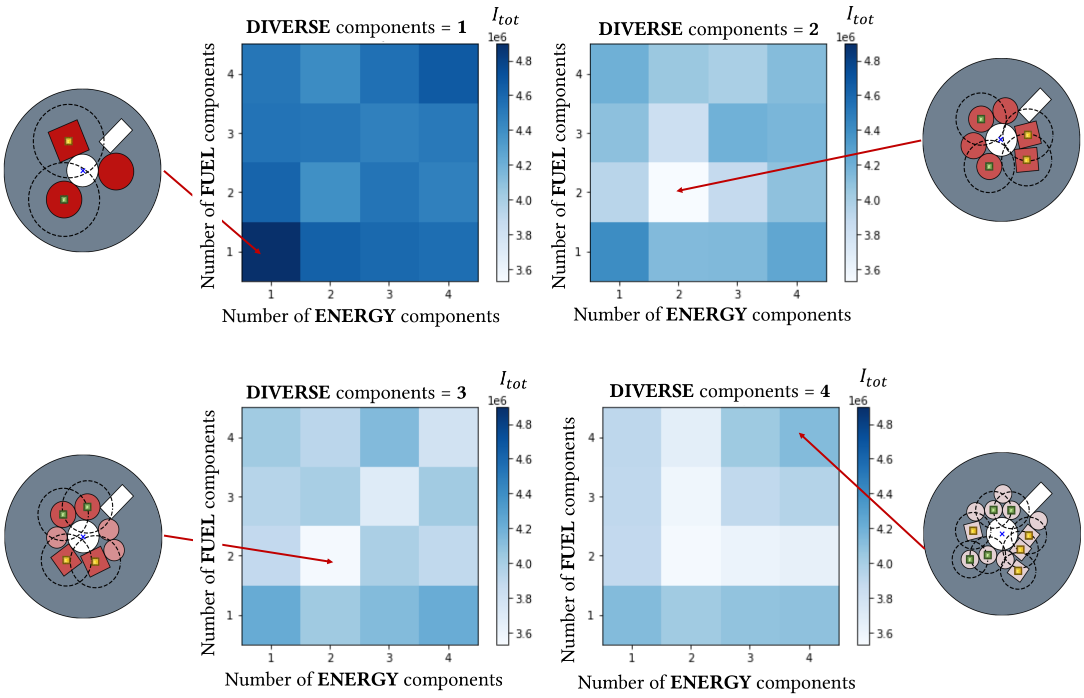


Figure 7.16: Colormaps showing the final inertia values obtained using the CSO-VF algorithm with respect to the number of components of each type, for each of the 64 subdivisions.

It can be seen that the more the diverse components are subdivided, the lighter the overall colormaps. This is due to the fact that when more but smaller components are positioned in the container, the empty space between the components can be reduced, leading to better inertias. Moreover, for a fixed number of diverse components, the inertia is smaller when the fuel and energy components are intermediately subdivided (*i.e.*, around 2 components). This is due to the fact that when the fuel and energy components are too subdivided, the functional exclusion zones around them space the components apart and thus, the corresponding inertia increases.

As a result, the lists of components leading to the best inertias are those for which the fuel and energy components are intermediately subdivided, and the diverse components fully subdivided.

7.4.1.3 Algorithm configuration for the toy case

The surrogate assisted-CSO-VF algorithm is applied to the aforementioned toy case in order to assess its ability to find the promising areas of the conditional search space that is completely known. To do so, the same CSO-VF algorithm than in the previous section is used at the lower level, and standard settings obtained by a parametric analysis are used for Bayesian Optimization at the upper level. They are summarized as follows:

- **Computational budget:** The Bayesian Optimization process is run with an initial training data set of $N_{DoE} = 10$ samples and $N_{it} = 10$ iterations;
- **Kernel:** The categorical Hamming kernel is used;
- **Kernel approach:** The level-wise categorical kernel approach is used;
- **Noisy data modeling:** The noisy data modeling is used and a nugget with a value of 0.1 is set;
- **Gaussian Process training:** The Adam Optimizer, run over 50000 iterations, is used for the GP training;
- **Infill criteria:** The Expected Improvement (EI) infill criterion is used;
- **Optimization of the infill criteria:** a Genetic Algorithm whose settings are obtained using a parametric study is used.

7.4.1.4 Assessment of the algorithm on the toy case

The surrogate assisted-CSO-VF algorithm with the configuration detailed in the previous section is run 50 times on the toy case. Figure 7.17 shows the median and interquartile range of the convergence curves of the Bayesian Optimization process. It must be noted that the curves are convergence ones with respect to the optimization of the conditional variables $\mathbf{w}$. Each point of the curves supposes that the other variables (*i.e.*, depending on continuous and/or discrete variables related to the layout) have been optimized by the lower level. Figure 7.18 shows on the colormaps which subdivisions the 50 instances of the algorithm have converged to. Finally, Table 7.2 sums up the numerical results.

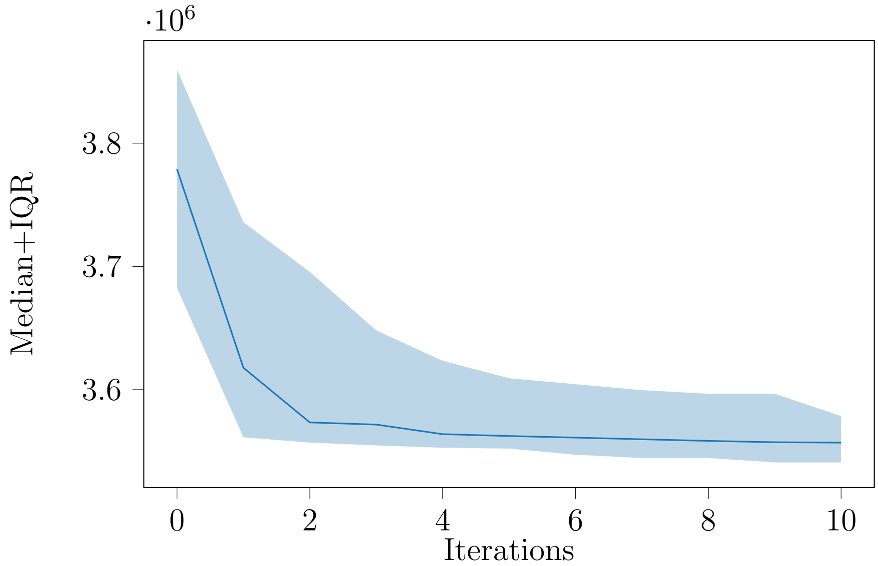


Figure 7.17: Median and interquartile range of the convergence curves of the Bayesian Optimization process (50 repetitions).

Metric	Value
Final median	3.557e6
Final IQR	3.760e4
Best run	3.461e6
Best component list	Fuel = 2 ; Energy = 2 ; Diverse =4

Table 7.2: Numerical results for the toy case (50 repetitions).

First of all, the proposed surrogate assisted-CSO-VF algorithm allows to solve the problem at hand. The median convergence curve shows that the algorithm globally converges in 10 iterations (with respect to an initial training data set of 10 samples). Moreover, the algorithm almost always finds one of the three component lists that provides the best inertia values. They correspond to lists with the fuel and energy components intermediately subdivided. Consequently, the conclusion can be drawn that the proposed surrogate assisted-CSO-VF algorithm can be used in order to solve the conditional search space optimal layout problems defined in Section 7.2.

In the following sections, the surrogate assisted-CSO-VF algorithm is thus configured and assessed on the application case detailed in Section 7.2 with 3888 possible component lists.

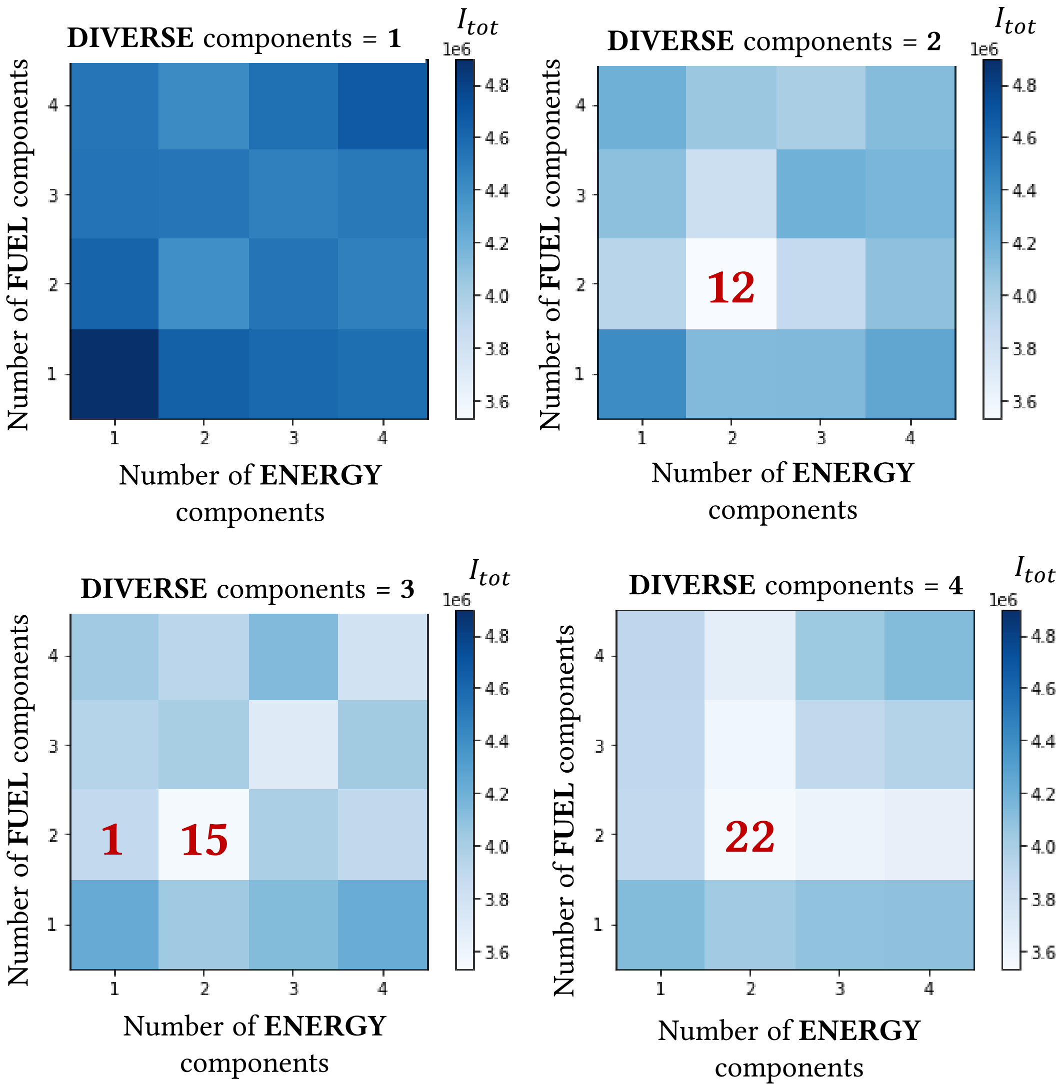


Figure 7.18: Optimal subdivisions found (50 repetitions). The red numbers correspond to the number of times the algorithm converges toward the corresponding subdivision.

7.4.2 Configuration of the surrogate assisted-CSO-VF algorithm

The two levels of the surrogate assisted-CSO-VF algorithm are configured in this section. As the application case detailed in Section 7.2 has a more complex conditional search space than the toy case, both levels are carefully configured.

7.4.2.1 Lower level configuration

The following settings are used to configure the CSO-VF algorithm:

- For each conditional variable set given by the upper level, following the conclusions drawn in Chapter 4, multistart is employed in order to improve the robustness of the algorithm. Five independent initializations are run in parallel.

- Each instance of the CSO-VF algorithm is run with 5000 iterations.
- The hyperparameters of the CSO-VF algorithm are optimized using Bayesian Optimization on representative subdivisions.

7.4.2.2 Upper level configuration

Choice of kernels. In order to configure the Bayesian Optimization algorithm as the upper level, a kernel must be chosen. The three kernels described in Chapter 6 are compared with respect to their prediction capabilities (CS_C: Compound Symmetry kernel with components based distance, CS_H: CS kernel with Hamming distance, LV: Latent Variable kernel). To do so, 20 training data sets of $N_{DoE} = \{50, 100\}$ conditional variables' vectors (*i.e.*, sets of components) are generated and evaluated thanks to the lower level. A training data set is mathematically denoted as follows:

$$\mathcal{T} = \{\mathbf{W}^T = \{\mathbf{w}_1^T, ..., \mathbf{w}_{N_{DoE}}^T\} \in F_w^T, \quad \mathbf{y}^T = \{y_1^T, ..., y_{N_{DoE}}^T\} \in F_y^T\} \tag{7.6}$$

For each training data set, the Gaussian Process (GP) surrogate models with the 3 different kernels are trained. In order to evaluate the modeling errors, a validation data set composed of $N_{VS} = 500$ conditional variables vectors is also generated and evaluated. The validation data set is denoted as follows:

$$\mathcal{V} = \{\mathbf{W}^V = \{\mathbf{w}_1^V, ..., \mathbf{w}_{N_{VS}}^V\} \in F_w^V, \quad \mathbf{y}^V = \{y_1^V, ..., y_{N_{VS}}^V\} \in F_y^V\} \tag{7.7}$$

The goal is thus to compare the GP predicted values for the validation data set and the real evaluated values. Two metrics are used to evaluate the modeling errors:

- The Root Mean Squared Error (RMSE) [CD14] (the smaller the better) to assess the validity of the GP prediction. The RMSE metric is calculated as follows:

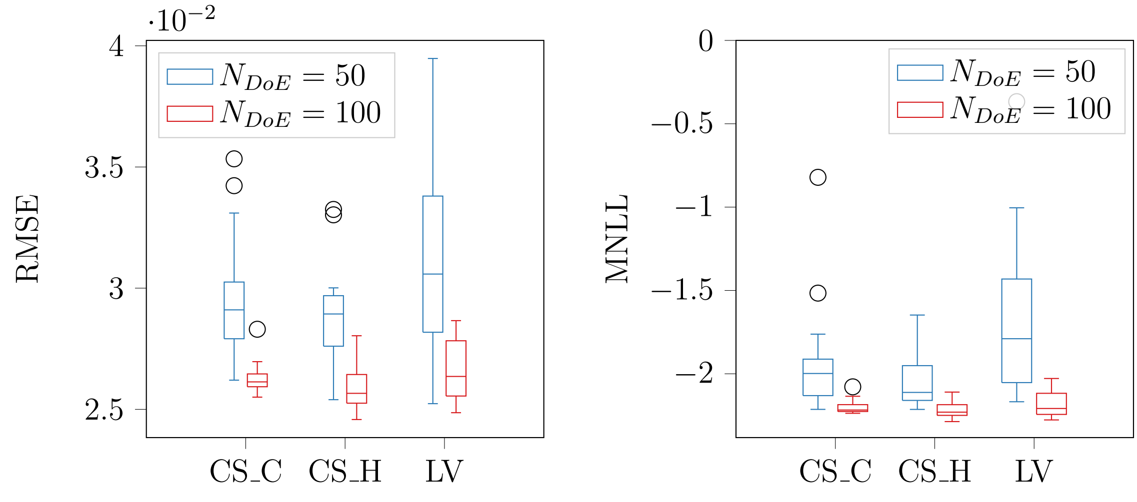
$$RMSE = \sqrt{\frac{1}{N_{VS}} \sum_{i=1}^{N_{VS}} (\hat{y}(\mathbf{w}_i^V) - y(\mathbf{w}_i^V))} \tag{7.8}$$

  where $\hat{y}(\mathbf{w}_i)$ is the predicted value of the black-box function *i.e.*, the predicted value of the inertia of the layout configuration for a set of components $\mathbf{w}_i$, by the GP surrogate modeling. The actual evaluated value is denoted $y(\mathbf{w}_i)$.

- The Mean Negative Log-Likelihood (MNLL) [War08] (the smaller the better) to assess the validity of the GP uncertainty model. The MNLL metric is calculated as follows:

$$MNLL = -\frac{1}{N_{VS}} \sum_{i=1}^{N_{VS}} \log p(\hat{y}(\mathbf{w}_i^V) | \mathbf{W}^T, \mathbf{Y}^T, \mathbf{w}_i^V) \tag{7.9}$$

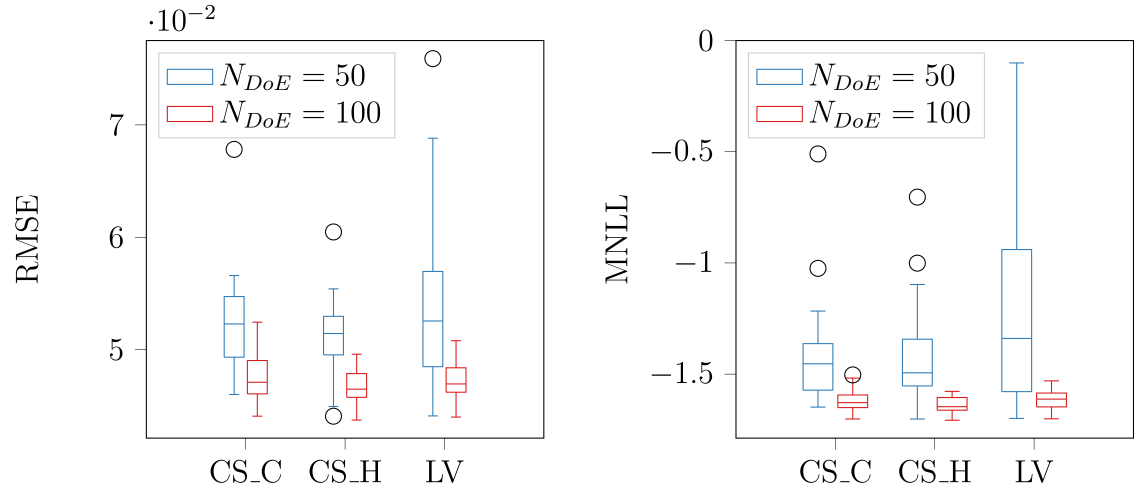
For 30% and 50% occupation rates, Figures 7.19a and 7.20a show the RMSE values for both studied training data set sizes. In the same way, for both occupation rates, Figures 7.19b and 7.20b show the MNLL values for both studied training data set sizes.



(a) RMSE, $OR = 30\%$

(b) MNLL, $OR = 30\%$

Figure 7.19: RMSE and MNLL for the 30% occupation rate and the 20 repetitions.



(a) RMSE, $OR = 50\%$

(b) MNLL, $OR = 50\%$

Figure 7.20: RMSE and MNLL for the 50% occupation rate and the 20 repetitions.

First of all, as might be expected, it is observed for all occupation rates that the larger the size of the training data set, the better are the GP predictions and uncertainty models. Moreover, similar trends are observed between the two occupations rates. The LV kernel has a larger dispersion then the other kernels for $N_{DoE} = 50$ due to the fact that the kernel has more hyperparameters than the CS kernels and the training data set must be large enough for the LV kernel to be trained correctly.

The CS_C and CS_H kernels have similar results in terms of their modeling errors. However, the CS_H kernel has generally either a smaller median value or a slightly smaller interquartile range for $N_{DoE} = 50$ regarding the two modeling error metrics. Consequently, the CS_H kernel is chosen and the initial number of samples in the training data set is set to $N_{DoE} = 50$.

Chosen configuration. In addition to a kernel function, some complementary configuration choices are made:

- Category-wise and level-wise categorical kernels: as explained in Chapter 6, the best approach must be chosen according to the characteristics of the conditional (*i.e.*, categorical) variables of the problem at hand. In the case of the detailed single-container satellite optimal layout problem, 12 categorical variables leading to 3888 categories (*i.e.*, different list of categorical variables) are considered. Thus, the category-wise kernels are in this case inoperable due to the large number of categories. Moreover, the assumption is reasonably made that the components can be chosen independently from each other. The level-wise approach is hence chosen;

- As concluded in Chapter 4, the CSO-VF algorithm outputs depend on the initialization. Consequently, a noisy data modeling approach is considered as detailed in Chapter 6 which require the definition of a nugget. Using a parametric analysis, the nugget value is set to 1×10^{-1} which corresponds to the standard deviation of the dispersion of CSO-VF algorithm;

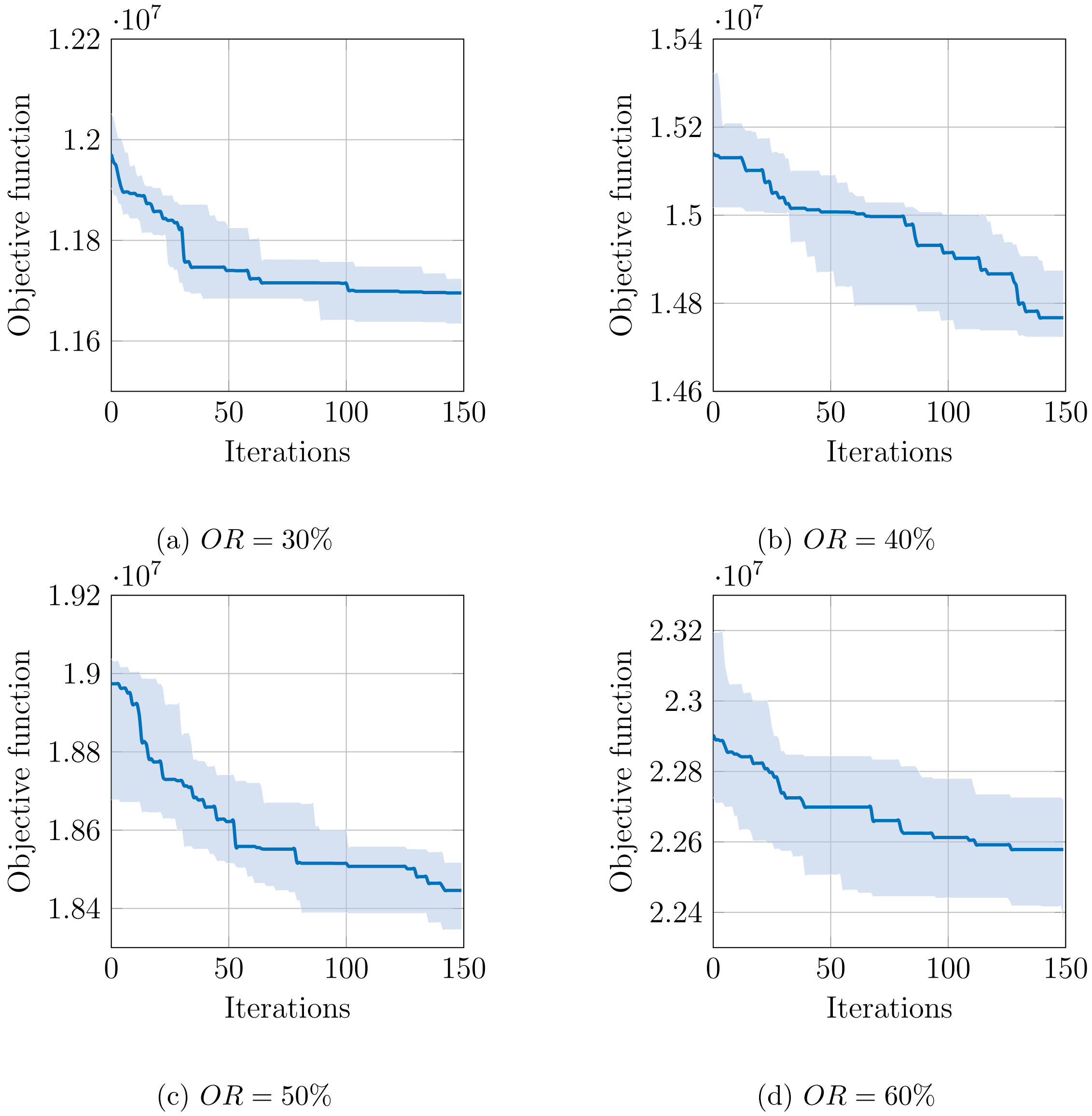
- The Adam Optimizer responsible for the GP training is run during 50000 iterations;

- The EI infill criterion is considered;

- The settings of Genetic Algorithm responsible for the infill criterion optimization are determined using a parametric analysis: population of 200 individuals, 500 generations, probability of crossover: 0.9, probability of mutation: 0.1.

7.4.3 Application of the surrogate assisted-CSO-VF algorithm to the global layout problem

In this section, the bilevel approach combining Bayesian Optimization and the CSO-VF algorithm is applied to the conditional search space satellite layout problem. Four occupation rates are studied: 30%, 40%, 50% and 60%. For each occupation rate, the algorithm is run 50 times. Figure 7.21 shows the medians and interquartile ranges of the convergence curves of the Bayesian Optimization process (*i.e.*, the upper level) for the four occupation rates. It must be noted that the convergence curves illustrate only the convergence with respect to the conditional variables. Indeed, the convergence with respect to the depending continuous and discrete variables related to the layout aspect *i.e.*, the convergence curves of the lower levels cannot be seen on the Bayesian Optimization convergence curves. Thus, each point of the upper level convergence curves has already optimized continuous and discrete variables.



(a) $OR = 30\%$

(b) $OR = 40\%$

(c) $OR = 50\%$

(d) $OR = 60\%$

Figure 7.21: Median and interquartile ranges of the convergence curves obtained with the BO+CSO-VF algorithm for the four studied occupation rates (50 repetitions).

Figure 7.21 shows that for all studied occupation rates, the proposed approach converges toward lists of components that provide feasible layouts and contribute to decrease the global inertia.

Furthermore, it must be noted that the algorithm has an exploration ability of the conditional search space which is completely fixed by the computational budget of the upper level. This computational budget is distributed between the number of initial samples in the training data set, N_{DoE}, and the number of iterations of the Bayesian Optimization process, N_{it}. Therefore, in this case, 200 lists of components are evaluated over the 3888 possible ones which corresponds approximately to 5% of the conditional categorical search space. This remains a small number of evaluations in the conditional search space, in spite of the fact that the initial samples are generated using discrete optimized LHS and that the Bayesian Optimization process aims to optimize the next samples to evaluate.

For the sake of conciseness, further analysis on the surrogate assisted-CSO-VF algorithm is conducted in the next section devoted to the comparison between the hidden-variable mechanism for GA and the present approach.

7.5 Comparison of both algorithms

The global performance of the two following algorithms are compared in this section:

- The hidden-variable mechanism for GA with the design variables implementation of chromosomes (HVM-DV) that provided better performance in terms of convergence accuracy, convergence speed, robustness to initialization than the HVM for GA with tags in Section 7.3.2;
- The surrogate assisted-CSO-VF algorithm (BO+CSO-VF) with the configuration determined in the previous section.

In Section 7.3.2 and Section 7.4.3, the HVM-DV and BO+CSO-VF methods have both been run 50 times for four different occupation rates: 30%, 40%, 50%, 60%. They have been run with the same computational budget: 5×10^6 function evaluations, distributed as follows according to the algorithm:

- HVM-DV: 500 individuals in the populations and 10,000 generations;
- BO+CSO-VF: for the upper level, a training data set of $N_{DoE} = 50$ samples and $N_{it} = 150$ iterations. For the lower level: $N_{start} = 5$ instances of the CSO-VF algorithm over $N_{it,CSO} = 5000$ iterations.

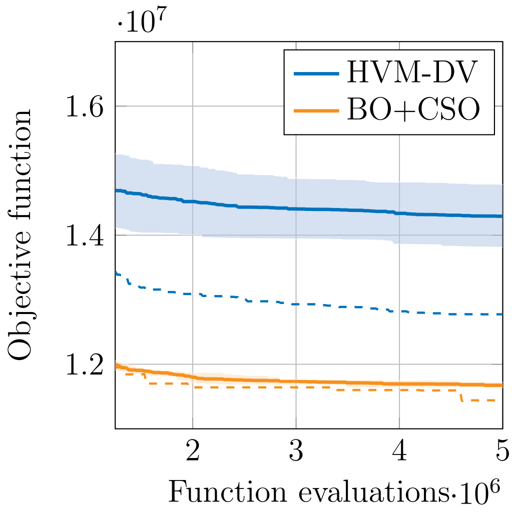
In order to compare the performance provided by the HVM for GA and the surrogate assisted-CSO-VF algorithm, the following metrics are adopted:

- **Median of convergence curves:** for each occupation rate, Figures 7.22a, 7.25a, 7.28a and 7.31a show the median of convergence curves of the 50 simulations for each algorithm. It must be noted that the convergence curves are plotted from the moment the Bayesian Optimization process of the BO+CSO-VF algorithms starts (*i.e.*, after the evaluation of the initial training data sets). Thus, the curves begin from 1.25×10^6 function evaluations (which correspond to the 2500th generations of the HVM-DV algorithm).
- **Best run:** for each occupation rate, Figures 7.22a, 7.25a, 7.28a and 7.31a also show the convergence curves, in dotted lines, of the simulation leading to the best layout *i.e.*, the smaller final inertia, for each algorithm,. As mentioned for the previous metric, the best runs are thus also plotted from 1.25×10^6 function evaluations.
- **Ranks of the algorithms:** Figures 7.22b, 7.25b, 7.28b and 7.31b indicate for each occupation rate the number of times each algorithm ranked first or second in terms of final objective function value for the 50 simulations.

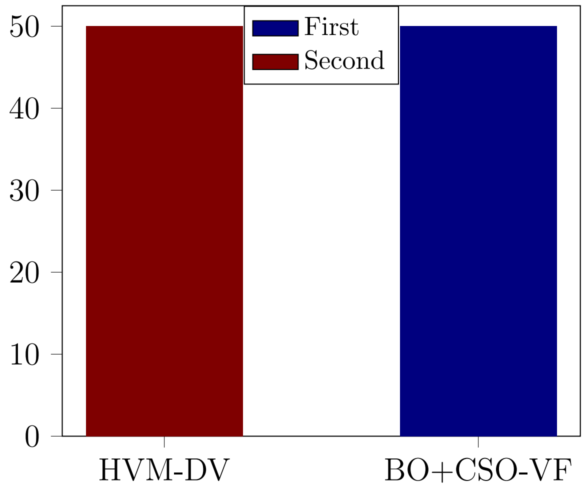
- **Successful runs to target:** for each occupation rate, Figure 7.23, 7.26, 7.29 and 7.32 show the number of simulations of each algorithm (among the 50) which manages to reach 25%, 15%, 10%, 5%, 3% and 1% of the best obtained median.
- **Number of components:** Figures 7.24, 7.27, 7.30 and 7.33 indicate the average number of each type of components (*i.e.*, fuel, energy and diverse) obtained in the final 50 layouts of both algorithms and the four occupation rates. The minimum and maximum numbers of components of each type is indicated in square brackets (*e.g.*, *Fuel [3-7]* indicates that from 3 to 7 fuel components can be chosen and positioned in the container).

Moreover, Figure 7.34 show the optimal layouts obtained for each occupation rate and each algorithm. Finally, Table 7.3 sums up the numerical results obtained for each algorithm. The metrics are:

- The success rate *i.e.*, the number of simulations providing a feasible solution;
- The median of the final objective function values;
- The interquartile range of the final objective function values;
- The best objective function values over the 50 simulations;
- The average number of explored lists of components over the 50 simulations.



(a) Median and best, $OR = 30\%$



(b) Ranks, $OR = 30\%$

Figure 7.22: Convergence curves and ranks for the 30% occupation rate.

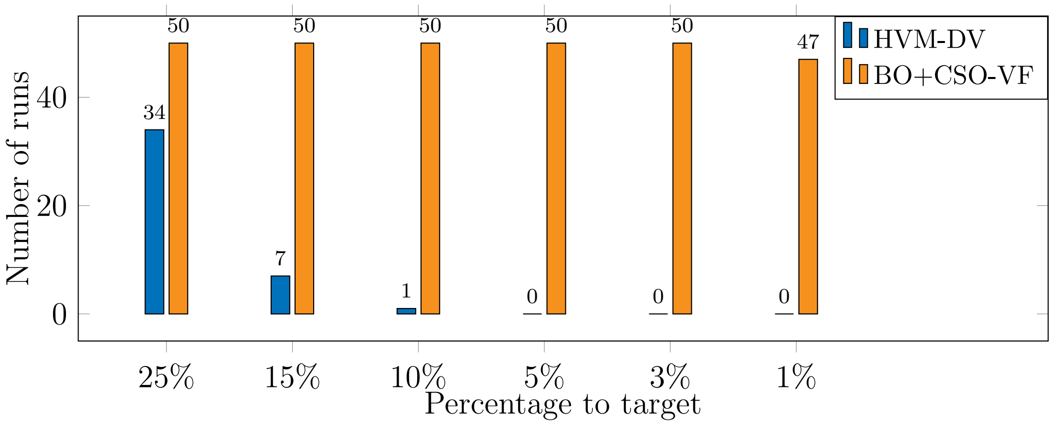


Figure 7.23: Success rate to percentages of the median target, $OR = 30\%$.

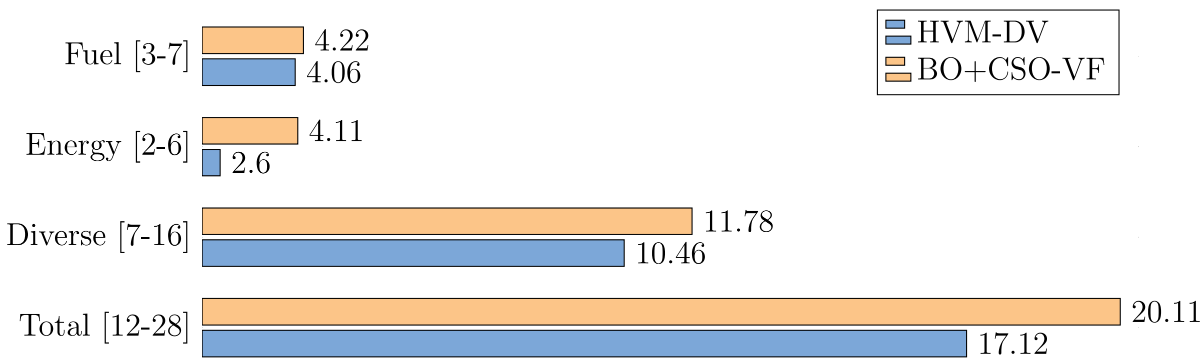
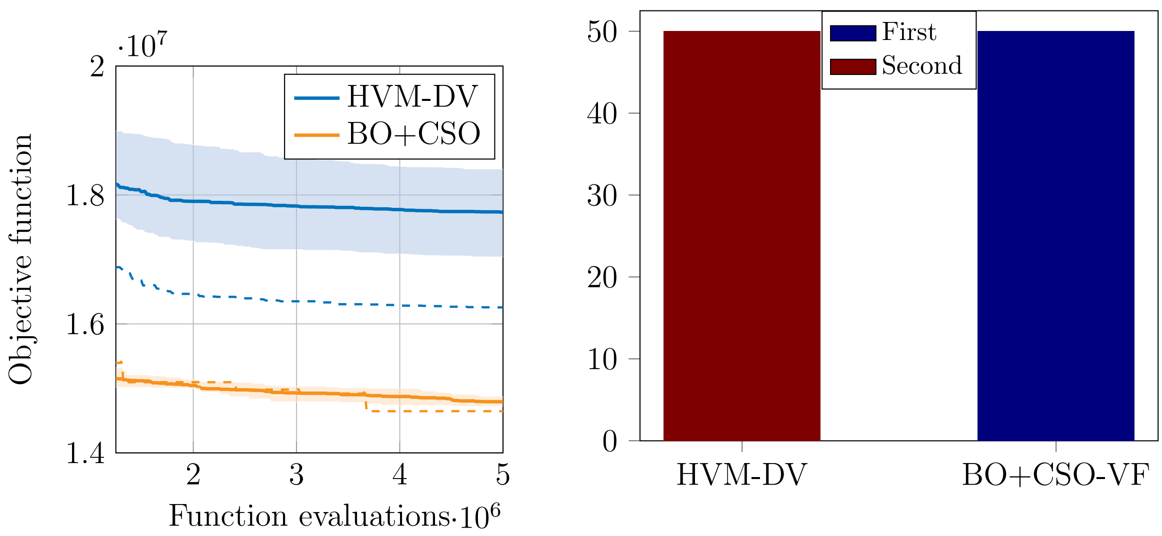


Figure 7.24: Number and types of components (mean), $OR = 30\%$.



(a) Median and best, $OR = 40\%$

(b) Ranks, $OR = 40\%$

Figure 7.25: Convergence curves and ranks for the 40% occupation rate.

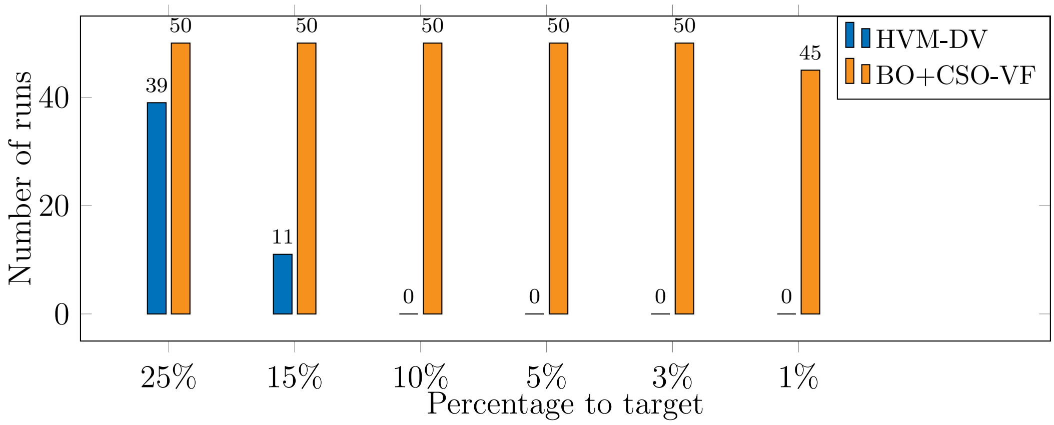


Figure 7.26: Success rate to percentages of the median target, $OR = 40\%$.

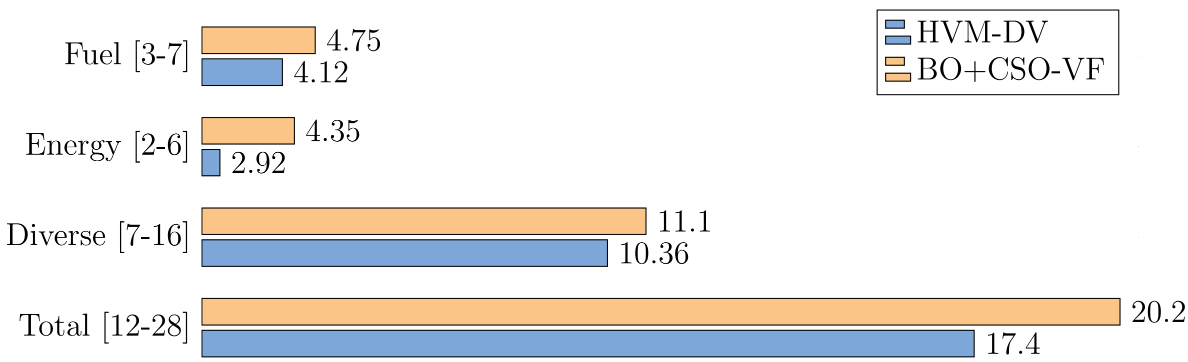
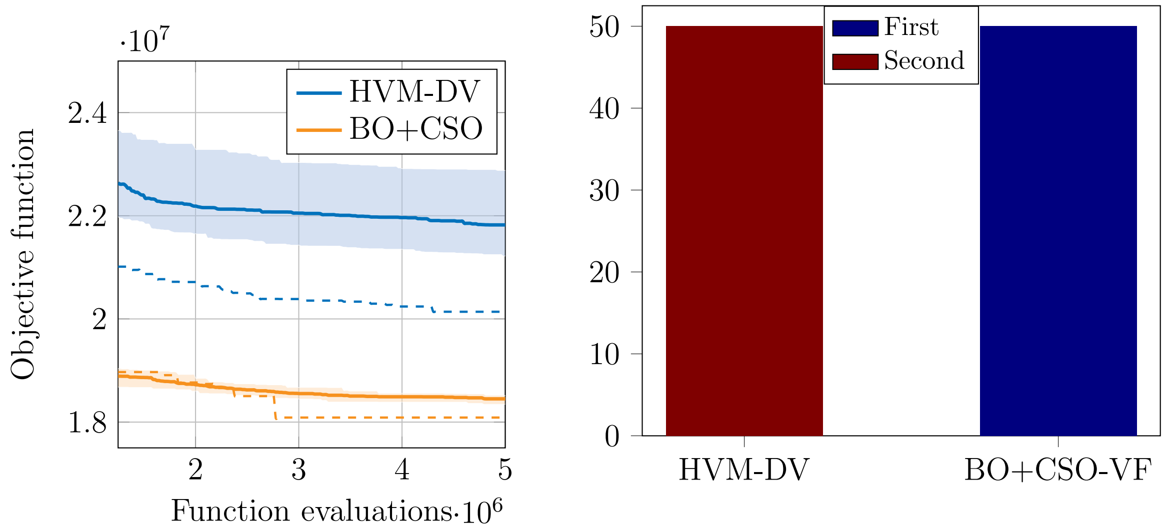


Figure 7.27: Number and types of components (mean), $OR = 40\%$.



(a) Median and best, $OR = 50\%$ (b) Ranks, $OR = 50\%$

Figure 7.28: Convergence curves and ranks for the 50% occupation rate.

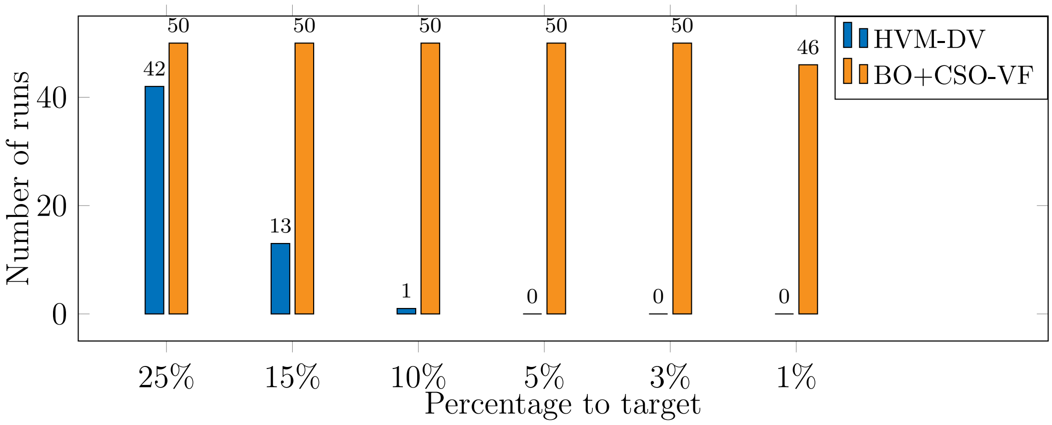


Figure 7.29: Success rate to percentages of the median target, $OR = 50\%$.

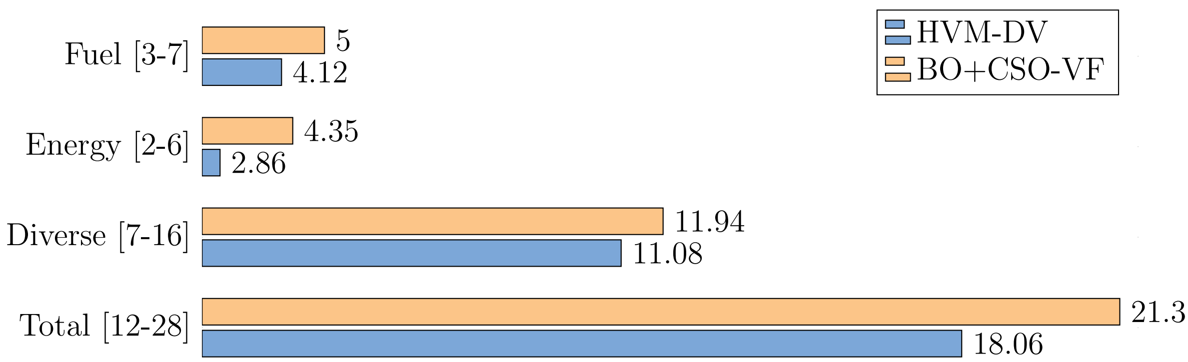
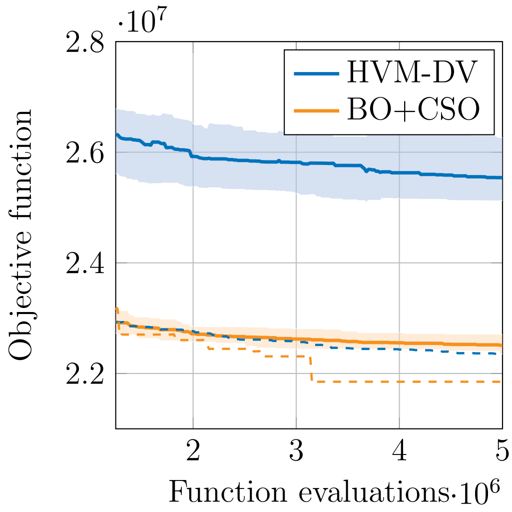
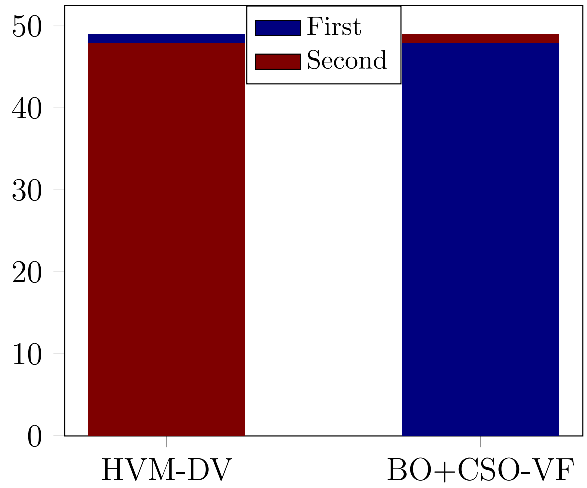


Figure 7.30: Number and types of components (mean), $OR = 50\%$.



(a) Median and best, $OR = 60\%$



(b) Ranks, $OR = 60\%$

Figure 7.31: Convergence curves and ranks for the 60% occupation rate.

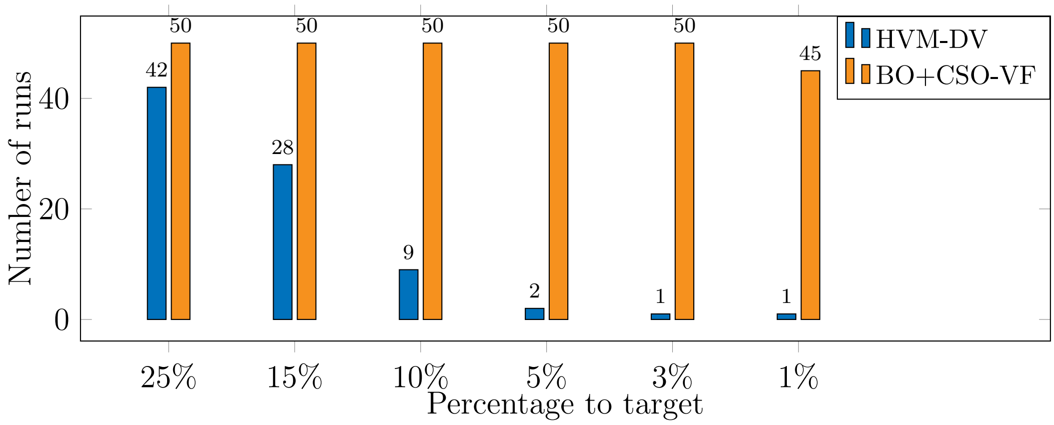


Figure 7.32: Success rate to percentages of the median target, $OR = 60\%$.

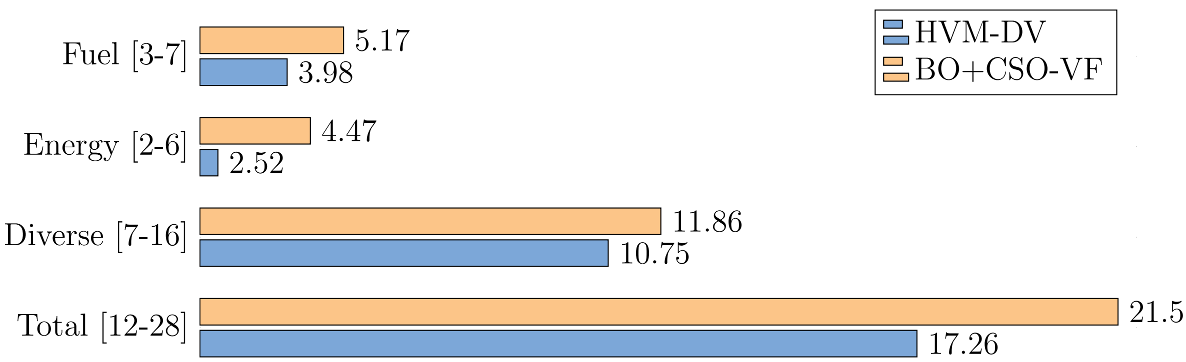


Figure 7.33: Number and types of components (mean), $OR = 60\%$.

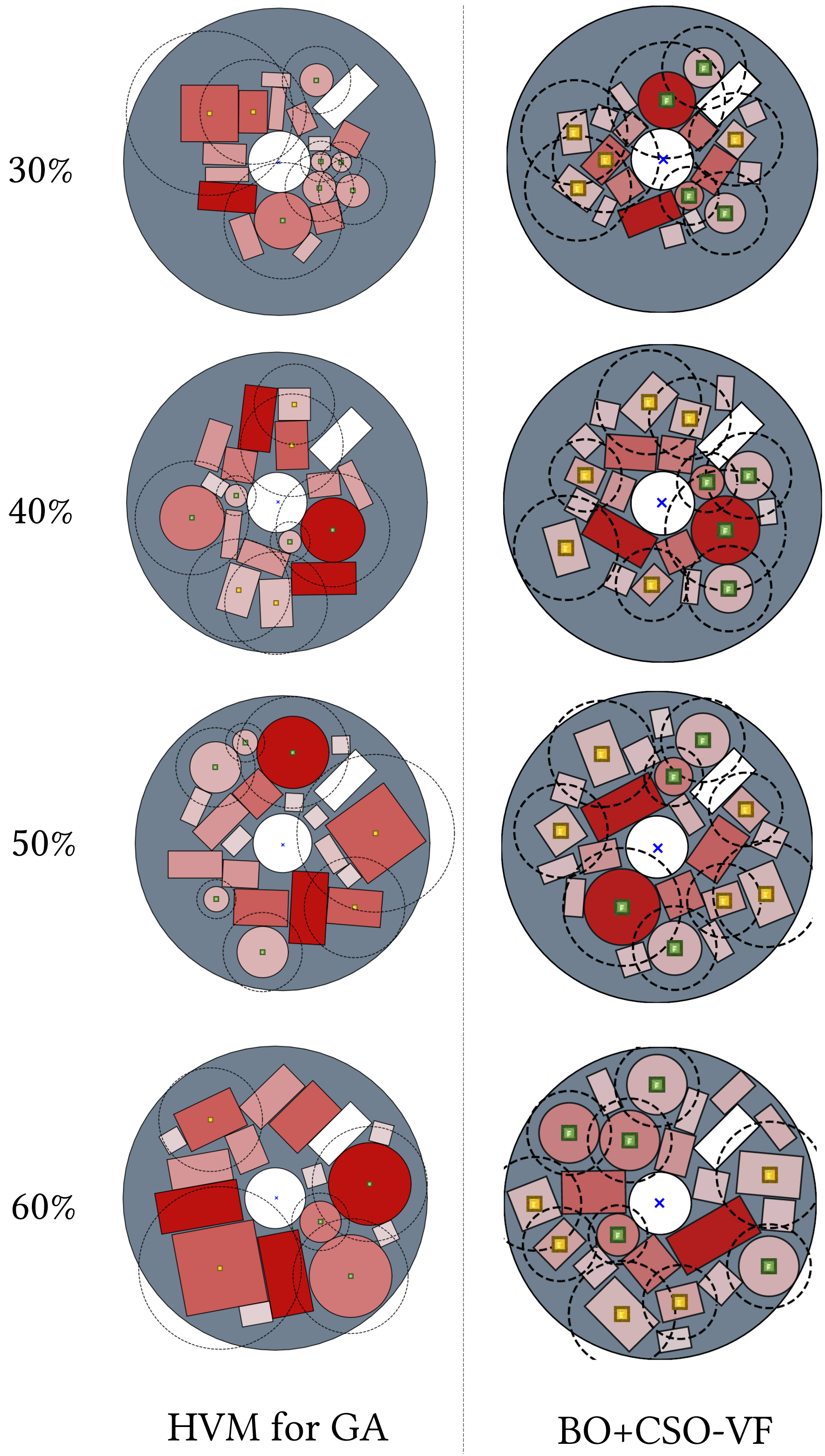


Figure 7.34: Best optimal layouts obtained for both algorithms and the four occupation rates.

	OR	HVM-DV	BO+CSO-VF	Relative difference
Success rate	30%	**50**	**50**	-
	40%	**50**	**50**	-
	50%	**50**	**50**	-
	60%	43	**50**	-14%
Final objective (median)	30%	1.429e7	**1.170e7**	-18.12%
	40%	1.773e7	**1.477e7**	-16.98%
	50%	2.182e7	**1.845e7**	-15.44%
	60%	2.554e7	**2.258e7**	-11.59%
Final IQR	30%	9.190e5	**8.954e4**	-90.25%
	40%	1.327e6	**1.435e5**	-89.19%
	50%	1.646e6	**1.719e5**	-89.56%
	60%	1.140e6	**3.711e5**	-67.44%
Best layout	30%	1.277e7	**1.144e7**	-10.42%
	40%	1.626e7	**1.465e7**	-9.90%
	50%	2.014e7	**1.809e7**	-10.18%
	60%	2.235e7	**2.185e7**	-2.24%
Explored subdivisions	30%	**473**	200	-57.72%
	40%	**483**	200	-58.59%
	50%	**472**	200	-57.63%
	60%	**477**	200	-58.07%

Table 7.3: Numerical results for the HVM for GA and the BO+CSO-VF algorithm and the four occupation rates. Bold values indicate best algorithm.

Success rate. Table 7.1 shows that until the 50% occupation rate, both methods allow 100% of the runs to find a feasible solution to the problem. However, for the higher occupation rate, 14% of the runs do not provide any feasible solution for the HVM-DV while the BO+CSO-VF approach maintains a 100% success rate.

Convergence accuracy. The BO+CSO-VF algorithm reaches a better objective function median for all occupation rates. Indeed, as reported in Table 7.3, the BO+CSO-VF algorithm allows to improve the final median by respectively 18.12%, 16.98%, 15.44% and 11.59% in comparison to the HVM-DV approach, for increasing occupation rates. Moreover, regardless of the occupation rate, the BO+CSO-VF algorithm always provides the best final inertia allowing to improve the best inertia by respectively 10.42%, 9.90%, 10.18% and 2.24%, for increasing occupation rates.

Figure 7.22b, 7.25b, 7.28b and 7.31b rank the values of the final objective function obtained by each algorithm over the 50 instances, for each occupation rates. It is observed that the BO+CSO-VF approach always ranks first for the three smaller occupation rates. For the 60% occupation rates it ranks first 98% of the time.

Robustness. For each occupation rate, the third figure shows the number of runs among the 50 trials that reach a certain percentages of the best final median value *i.e.*, the target value. For all the occupation rates, the BO+CSO-VF approach has

100% of its repetitions that reach 3% of its final median value, while the HVM-DV approach barely reaches 10% of this same value. It can be concluded that the BO+CSO-VF algorithm has a better robustness with respect to the number of runs reaching the target value than the HVM-DV method.

Moreover, the final interquartile ranges (IQR) obtained for both approaches and reported in Table 7.3 allow to quantify the dispersion of the final results in terms of objective function values. The BO+CSO-VF algorithm allows to improve the final IQR by respectively 90.25%, 89.19%, 89.56% and 67.44% in comparison to the HVM-DV algorithm, for increasing occupation rates. Overall, it can be concluded that the BO+CSO-VF has a better robustness to initialization.

Exploration of the conditional search space. The last metric from Table 7.3 reports the number of explored subdivisions. In the case of the HVM for GA approach, this corresponds to the number of different lists of components that are evaluated in the population over the generations. In the case of the BO+CSO-VF approach, this value is completely fixed by the computational budget of the upper level distributed between the number of samples in the initial training data set and the number of infilled samples during the Bayesian Optimization process (*i.e.*, $N_{DoE} = 50$ and $N_{it} = 150$ leading to a total number of 200 samples evaluations). As mentioned in the previous section, this means exploring approximately 5% of the conditional search space while the HVM for GA explore approximately 12% of the conditional search space. However, in addition to the Bayesian Optimization process that optimizes the evaluated lists of components, the structure of the BO+CSO-VF algorithm provides information about the quality of each list of components in terms of final layout. Indeed, the continuous and discrete design variables are optimized for each list of components that is being studied. On the other hand, when a subdivision is reached by the HVM-DV algorithm, the other design variables related to the layout are not optimized yet and some subdivisions can be quickly discarded in case of high constraints violations (related to continuous and discrete variables) because of the constraint-dominance rules. Then, the subdivisions providing the lower violations of constraints at a given time, are likely to be promoted in the population while they do not necessarily correspond to the most promising list of components in the end.

Furthermore, Figures 7.24, 7.27, 7.30 and 7.33 as well as the illustrations of the plates shown on Figure 7.34 demonstrate that the BO+CSO-VF algorithm converges toward subdivisions providing more components to be laid out, for a same occupation rate. Indeed, for all types of components, the BO+CSO-VF algorithm leads to a larger number of components positioned in the container. This is also due to the fact that for the HVM for GA algorithm, the Constraint-Dominance tends to give priority to individuals involving easily solved constraints *i.e.*, individuals with few components. On the other hand, the two-level structure of the BO+CSO-VF algorithm does not give an *a priori* priority to certain subdivisions.

Convergence speed. It must be noted that unlike the other algorithms studied in this thesis, the convergence speed is not properly studied in this section. In Chapter 4 or in Section 7.3.2 of the present chapter, the iterations providing the first feasible solutions are reported and compared, as well as the number of iterations (or number of function evaluations) needed to reach percentages of the best final median value. However, the BO+CSO-VF algorithm does not really lend itself to the study of these metrics because of its inherent structure. Indeed, the iterations of the Bayesian Optimization process begin after the evaluation of the initial training data set which already corresponds to function evaluations without being part of the convergence *per se*. Moreover, as the design variables are optimized in two distinct optimization processes, a metric based on the number of function evaluations is difficult to define. However, it is worth noting that for the 30% to 50% occupation rates, all the samples from the initial training data set as well as the samples evaluated during the iterations of Bayesian Optimization provide feasible layouts. For the 60% occupation rate, on average 5% to 10% of the samples from the 50 initial training data set provide unfeasible solutions. Despite this, 100% of the infilled samples during the Bayesian Optimization processes provide feasible solutions. Finally, the metrics used to study the convergence speed of the HVM for GA are reported and analyzed in Section 7.3.2.

In summary, the performance of tje surrogate assisted-CSO-VF algorithm have been assessed using its application to the proposed conditional search space single-container layout problem. The various used metrics allow to draw the conclusion that this approach provides better performance in terms of convergence accuracy, convergence speed, robustness to the initialization, with respect to the given computational budget. Moreover, even if the HVM for GA algorithm reaches more lists of components during its convergence process, the BO+CSO-VF algorithm has better ability to identify promising lists of components in terms of final layout objective function. These trends can be due both to the Bayesian Optimization technique which optimizes the choice of the components lists to be evaluated, as well as to the virtual-force system and operators of the nested lower level of the BO+CSO-VF algorithm which provide dedicated techniques to address the constraints easier than for the hidden-variables method in which the constraints are handled using Constraint-Dominance rules.

7.6 Advantages and limitations of the algorithms

7.6.1 Hidden-variables mechanism for Genetic Algorithm

The main advantages of the hidden-variables mechanism for Genetic Algorithm (HVM for GA) are as follows:

- This method is generic and versatile and allows to deal with a large range of conditional search space optimal layout problems;

- The classical GA evolutionary operators can directly be used in the case where the conditional variables are implemented as design variables (*i.e.*, the HVM-DV configuration). When tags are used to implement the conditional variables (*i.e.*, the HVM-TAG configuration), some adaptations are needed in order to provide semantically correct recombinations and mutations. Few hyperparameters are required and correspond to the hyperparameters of each GA evolutionary operator;

- In theory, the algorithm is not limited in the number of components it can handle. Moreover, it could also deal with multi-container configurations, in which depending categorical variables related to the assignment of the chosen components to the containers would be implemented as hidden or revealed discrete variables. However, the length of the chromosome as well as the number of constraints would increase which is likely to alter the global performance of the algorithm.

Some limitations have been identified:

- Even if the length of the chromosome is not theorically limited, all the design variables related to all the components from the given catalogue are implemented in the chromosome. This can lead to very long chromosomes in case of a large number of generic components and number of subdivisions which can affect the global performance of the HVM for GA.

- The main drawback of the approach is the constraint handling based on constraint-dominance rules. The constraint-dominance promotes individuals with low constraints violation values among the population and thus does not allow to explore sufficiently the conditional search space and to identify promising lists of components.

7.6.2 Surrogate assisted-CSO-VF algorithm

The main advantages of the surrogate assisted-CSO-VF algorithm (BO+CSO-VF) approach are as follows:

- The BO+CSO-VF, because of its bilevel inherent structure, allows to study each list of components in depth and to identify the area of the conditional search space proving the most promising lists of components in terms of final layout objective function.

- It benefits from the advantages of the Bayesian Optimization technique at the upper level and particularly its ability to converge quickly even in case of expensive black-box functions.

- It also benefits from the advantages of the CSO-VF algorithm at the lower level and particularly its ability to solve the constraints even for a large number of components using dedicated operators.

Some limitations are also identified:

- In its original form, the Bayesian Optimization technique is limited in terms of number of design variables that can be handled. The number of generic components that are subdivided are thus limited to a few dozens. Recent enhancements of the Bayesian Optimization technique for dealing with more conditional variables need to be used (*e.g.*, the Kriging model with Partial Least Squares [Sav+22]) in case of more generic components. It must be noted that on the contrary, the number of components does not affect the CSO-VF algorithm subsequently because of its decentralized inherent aspect.

- The conditional search space exploration ability of the algorithm is completely fixed by the given computational budget (number of samples in the initial training data set and then infilled during the Bayesian Optimization process).

- The algorithm relies on a quite large number of settings and hyperparameters because it uses two dedicated algorithms at each level.

- It should also be noted that the bilevel BO+CSO-VF algorithm cannot deal with multi-container configurations of the problem. Indeed, such problems add depending categorical design variables characterizing the assignment of the chosen components to the containers. However, the CSO-VF cannot deal with multi-container configurations. Moreover, if both conditional and depending variables are optimized in the upper level by the Bayesian Optimization technique, kernels for conditional search space must be employed [Hor+19; HO13a; Pel+21; ZH18b], as well as scalability related enhancements of the Bayesian Optimization [Sav+22].

7.7 Conclusion

In this chapter, the fixed search space single-container satellite layout problem described in Chapter 4 was extended in order to integrate a conditional search space (CSS). This proposed benchmark was used to assess the performance of the algorithms described in Chapter 6.

First, the hidden-variables mechanism for GA (HVM for GA) has been configured and assessed. The list of operators was defined using Taguchi's plans of experiments and both proposed implementations of the chromosomes have been compared using their applications to the proposed CSS single-container satellite layout benchmark. As a result, both implementations of the algorithm solve the problem at hand. When conditional variables are implemented as design variables (*i.e.*, the HVM-DV configuration), the algorithm provides better performance in terms of convergence accuracy, convergence speed and robustness to the initialization. However, the tag-based implementation of the conditional variables has more abilities in exploring the conditional search space.

Subsequently, the surrogate assisted-CSO-VF (BO+CSO-VF) algorithm has been configured and assessed. A toy case characterized with 64 possible different lists of components has been devised and the performance has been illustrated using this toy case. It has then been configured for the satellite benchmark characterized with 3888 possible lists of components. The global performance of the HVM for GA and the BO+CSO-VF algorithm have been compared using their application to the CSS satellite benchmark. As a result, the BO+CSO-VF algorithm provides better performance notably in terms of success rate, convergence accuracy and robustness to initialization in comparison to the HVM for GA, for a given computational budget.

Indeed, the bilevel structure of the BO+CSO-VF algorithm combines the strength of both dedicated algorithms. The Bayesian Optimization at the upper level helps to efficiently explore the conditional search space while the CSO-VF algorithm at the lower level provides quality layouts by efficiently solving the constraints.

However, the surrogate assisted-CSO-VF algorithm cannot deal with multi-containers satellite layout problems. Thus, the next chapter is devoted to the definition of conditional search space multi-container satellite layout problems and to the development and assessment of a tri-level approach based on Bayesian Optimization, Genetic Algorithm and the CSO-VF algorithm for solving the aforementioned problems.

Chapter 8

Tri-level Approach for Conditional Search Space Multi-container Layout Problems

Contents

Chapter contributions

- Design of the conditional search space multi-container layout problems.
- Development of a tri-level approach combining Bayesian Optimization, Genetic Algorithm and the CSO-VF algorithm.
- Assessment of the performance of the proposed tri-level algorithm on a conditional search space multi-container satellite layout problem.

8.1 Introduction

In Chapter 7, the satellite benchmarks are limited to single-container satellite configurations. Therefore, the objective of this chapter is to extend the application cases tackled in the previous chapters to conditional search space multi-container satellite layout problems and to develop a method to handle them. The proposed approach is a tri-level approach combining the previous techniques proposed in this manuscript: Bayesian Optimization for the choices of components, Genetic Algorithm for the assignment of the chosen components to containers and the CSO-VF algorithm for the layout of the chosen components in their assigned containers.

The rest of the chapter is organized as follows: Section 1 is devoted to the formulation of conditional search space multi-container optimal layout problems. The tri-level approach is described in Section 2. Section 3 is dedicated to the assessment of the method using a satellite module layout benchmark.

8.2 Formulation of conditional search space multi-container optimal layout problems

Conditional search space multi-container optimal layout problems (OLPs) consist in assigning N generic components, each having n_i possible subdivisions ($i \in \{1, ..., N\}$, to n containers along with optimizing their layout in the assigned containers. In other words, three tasks must be solved in order to obtain an optimal layout: the choice of the components, their assignment to containers and their layout optimization within their containers. The following sections are dedicated to the description and formulation of such problems.

8.2.1 Geometry of the components and containers

Multi-container OLPs involve:

- A set of N generic components of any shape, each having n_i possible subdivisions;
- A set of n containers of any shape.

Figure 8.1 illustrates the problem at hand, decomposed in three tasks: the choice of components, the assignment and layout tasks.

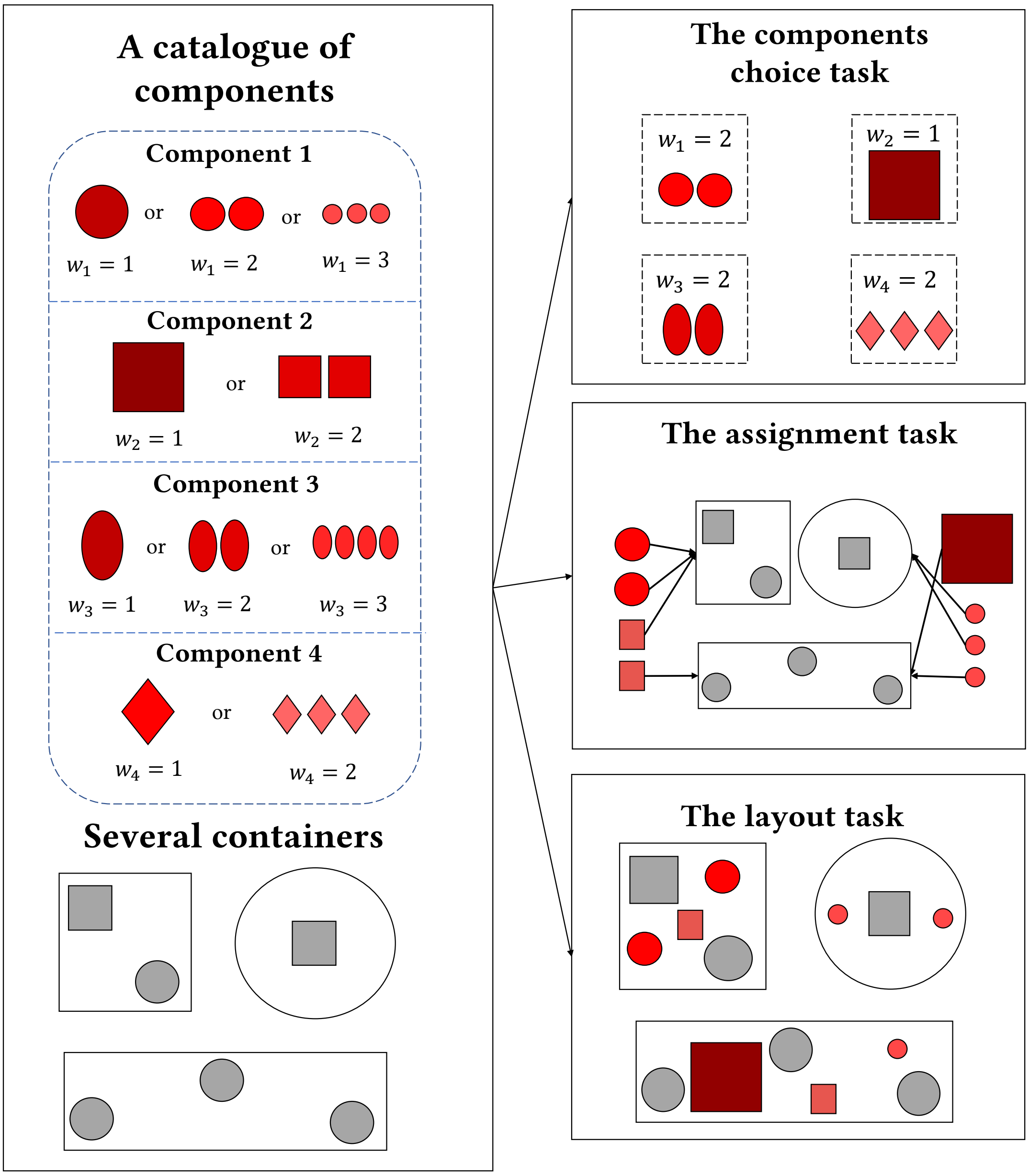


Figure 8.1: Illustration of the CSS multi-container satellite layout problem on an example.

8.2.2 Design variables

The design variables that require to be optimized are the following:

- The selected subdivision of each generic component, which correspond to conditional categorical variables $\mathbf{w} = \{w_i\}$ with $w_i \in \{1, ..., n_i\}$ for $i \in \{1, ..., N\}$;
- The index of the container assigned to each component, which corresponds to dependent discrete unordered variables *i.e.*, categorical variables, $\mathbf{z}_c = \{z_{c,i}\}$ with $z_{c,i} \in \{1, ..., n\}$ for $i \in \{1, ..., N\}$;
- The positions of the center of inertia of each component in its assigned container which are considered as dependent continuous variables $\mathbf{x}_p$;
- The orientations of the components whose shape is not circular. According to the problem specifications, design variables relative to components orientations can correspond either to continuous variables or to discrete variables. For illustrative purposes, orientations are considered continuous variables in this section: $\mathbf{x}_\alpha$.

8.2.3 Objective function

The objective functions that can be optimized are for instance:

- Any dynamical requirements (*e.g.*, overall mass, inertia or stability of the system);
- Any costs related to the layout (*e.g.*, manufacturing or handling costs);
- Any performance of the layout (*e.g.*, power consumption or emission).

8.2.4 Constraint functions

The constraint functions can be split into three categories:

Constraints relative to the choice of components. Some constraints are specific to the choice of the components. They can involve for instance a maximum value of total mass or a maximum value of occupation rate.

Constraints relative to the assignment of the chosen components to the containers. Some constraints are specific to the assignment of the components to the containers. They can involve for instance balancing constraints along one axis, a maximum value of occupation rate of one or several containers, functional constraints that position compatible or incompatible components in respectively the same or distinct containers for functional reasons, *etc.*

Constraints relative to the layout. The layout constraints rely on the positioning of the chosen components. They are similar to those defined for the FSS OLPs and are briefly summarized as follows:

- **Geometrical constraints:** These constraints translate geometric specifications on the layout and only depend on geometric and masses considerations such as no overlapping between the components, the total inclusion of the components into the container, balancing constraints (*i.e.*, the global center of mass must be positioned at the geometrical center of the container), *etc.*;

- **Functional constraints:** These constraints do not only depend on geometric considerations, but also on system-level specifications. One example is, for instance, proximity or minimal distance requirements between components for functional requirements (*e.g.*, radiative constraints).

8.2.5 Mathematical formulation

As in Chapter 6 and Chapter 7, two cases are considered herein:

Case 1. The conditional variables $\mathbf{w}$ responsible for the choices of the components are optimized simultaneously with the dependent variables $\mathbf{z}_c$, $\mathbf{x}_\mathbf{p}, \mathbf{x}_\alpha$ related to the corresponding assignment and layout, in a single optimization process. Thus, it is formulated mathematically as follows:

$$
\begin{aligned}
\min_{\mathbf{x}_p,\mathbf{x}_\alpha,\mathbf{z_c},\mathbf{w}} \quad & f_{obj}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}(\mathbf{w})}, \mathbf{x}_\alpha \in F_{x_\alpha} \subseteq \mathbb{R}^{n_{x_\alpha}(\mathbf{w})}, \\
& \mathbf{z_c} \in F_{z_c} \subseteq \mathbb{N}^{n_{z_c}(\mathbf{w})}, \mathbf{w} \in F_w \subseteq \mathbb{N}^{n_w} \\
\text{s.t.} \quad & \mathbf{h}_{choice}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) = 0 \\
& \mathbf{g}_{choice}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) \leq 0 \\
& \mathbf{h}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) = 0 \\
& \mathbf{g}_{assignment}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) \leq 0 \\
& \mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) = 0 \\
& \mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z_c}, \mathbf{w}) \leq 0
\end{aligned}
\tag{8.1}
$$

where f_{obj} is the objective function, $\mathbf{h}_{choice}, \mathbf{g}_{choice}$ are the equality and inequality constraints relative to the choice of the components, $\mathbf{h}_{assignment}, \mathbf{g}_{assignment}$ are the equality and inequality constraints relative to the assignment of the components to their containers and $\mathbf{h}_{layout}, \mathbf{g}_{layout}$ are the equality and inequality constraints relative to the layout of the components in the containers.

Case 2. The previous formulation is written as a nested formulation coupled with a two-stage formulation. This problem formulation is proposed so as to optimize both the conditional variables and the dependent variables in two sequential optimization processes as in Chapter 6 and Chapter 7, and the dependent variables in a two-stage process as in Chapter 3 and Chapter 4.

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & f_{obj}(\mathbf{x}_p^{**}, \mathbf{x}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{w} \in F_w \subseteq \mathbb{N}^{n_w} \\
\text{s.t.} \quad & \mathbf{h}_{choice}(\mathbf{w}) = 0 \\
& \mathbf{g}_{choice}(\mathbf{w}) \leq 0 \\
& \mathbf{h}^*_{assignment}(\mathbf{z}_c^*, \mathbf{w}) = 0 \\
& \mathbf{g}^*_{assignment}(\mathbf{z}_c^*, \mathbf{w}) \leq 0 \\
& \mathbf{h}^{**}_{layout}(\mathbf{x}_p^{**}, \mathbf{x}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& \mathbf{g}^{**}_{layout}(\mathbf{x}_p^{**}, \mathbf{x}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) \leq 0 \\
& \{\mathbf{z_c}^*\} = \text{argmin } f_{obj}^{assignment}(\mathbf{z}_c, \mathbf{w}) \\
& \quad \text{w.r.t.} \quad \mathbf{z}_c \in F_{z_c} \subseteq \mathbb{N}^{n_{z_c}(\mathbf{w})} \\
& \quad \text{s.t.} \quad \mathbf{h}_{assignment}(\mathbf{z}_c, \mathbf{w}) = 0 \\
& \quad \phantom{\text{s.t.}} \quad \mathbf{g}_{assignment}(\mathbf{z}_c, \mathbf{w}) \leq 0 \\
& \quad \phantom{\text{s.t.}} \quad \{\mathbf{x_p}^{**}, \mathbf{x}_\alpha{}^{**}\} = \text{argmin } f_{obj}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c, \mathbf{w}) \\
& \quad \phantom{\text{s.t.}} \quad \text{w.r.t.} \quad \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}(\mathbf{w})}, \mathbf{x}_\alpha \in F_{x_\alpha} \subseteq \mathbb{R}^{n_{x_\alpha}(\mathbf{w})} \\
& \quad \phantom{\text{s.t.}} \quad \text{s.t.} \quad \mathbf{h}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c, \mathbf{w}) = 0 \\
& \quad \phantom{\text{s.t.}} \quad \phantom{\text{s.t.}} \quad \mathbf{g}_{layout}(\mathbf{x}_p, \mathbf{x}_\alpha, \mathbf{z}_c, \mathbf{w}) \leq 0
\end{aligned}
\tag{8.2}
$$

The design variables denoted with a star exponent (*e.g.*, $\mathbf{x}^*$) referred to as the design variables optimized in the lower level and which are thus fixed in the upper level. In the same way, the constraints denoted with a star exponent (*e.g.*, $\mathbf{h}^*(\cdot)$) referred to as the constraints handled in the lower level but which must ensure the feasibility of the layout for the list of components defined by $\mathbf{w}$ at the upper level.

It must be noted that other formulations can be proposed, for instance considering either the choices of the components and the assignment tasks in the same optimization process, or the assignment and the layout tasks in the same optimization process. However, these cases have been indirectly discarded in Chapter 3 in which the two-stage approach for the assignment and layout tasks has been chosen. Thus, they are not detailed here, for conciseness purposes.

In the following, the second formulation (*i.e.*, Equations 8.2) is chosen. Indeed, the first formulation could be addressed using the hidden-variable mechanism for Genetic Algorithm proposed in Chapter 6 and by adding the $\mathbf{z}_c$ variables as depending hidden or revealed genes. As concluded in Chapter 7, it appears that this technique provided reduced performance when compared to its surrogate assisted-CSO-VF algorithm counterpart relying on a nested formulation, notably in terms of constraint satisfaction, convergence accuracy and robustness to initialization. As the former conditional search space single-container layout problems are enhanced with the multi-container peculiarities, which includes additional design variables and constraints, the search space become even more complex. Consequently, the second formulation which arises from the nested formulation proposed in Chapter

6 and Chapter 7 for the choices of the components and the layout tasks, as well as that from the two-stage formulation proposed in Chapter 3 and Chapter 4 for the assignment and layout tasks is considered.

8.3 Tri-level approach combining Bayesian Optimization, Genetic Algorithm and Component Swarm Optimization based on a Virtual-force system algorithm

In this section, the tri-level approach developed to handle the chosen mathematical formulation is developed.

8.3.1 General description

The proposed tri-level algorithm is designed to solve the conditional search space multi-container satellite module layout problem formulated as a combination of a nested and a two-stage approaches. Three levels aim to handle each optimization process with the help of three dedicated algorithms:

- Upper level: it addresses the choices of the components task, using Bayesian Optimization;
- Intermediate level: addresses the assignment of the chosen components on the different containers, using Genetic Algorithm;
- Lower level: it addresses the layout of the chosen components, with the help of the CSO-VF algorithm described and assessed in Chapter 3 and Chapter 4.

The tri-level algorithm is thus a combination of the two-stage approach which combines the use of a Genetic Algorithm with the CSO-VF algorithm for solving fixed search space multi-container OLPs (Cf. Chapter 3 and Chapter 4), and the surrogate assisted-CSO-VF algorithm used for solving conditional search space single-container OLPs (Cf. Chapter 6 and Chapter 7). Figure 8.2 describes the tri-level algorithm and the three levels are described in the following sections.

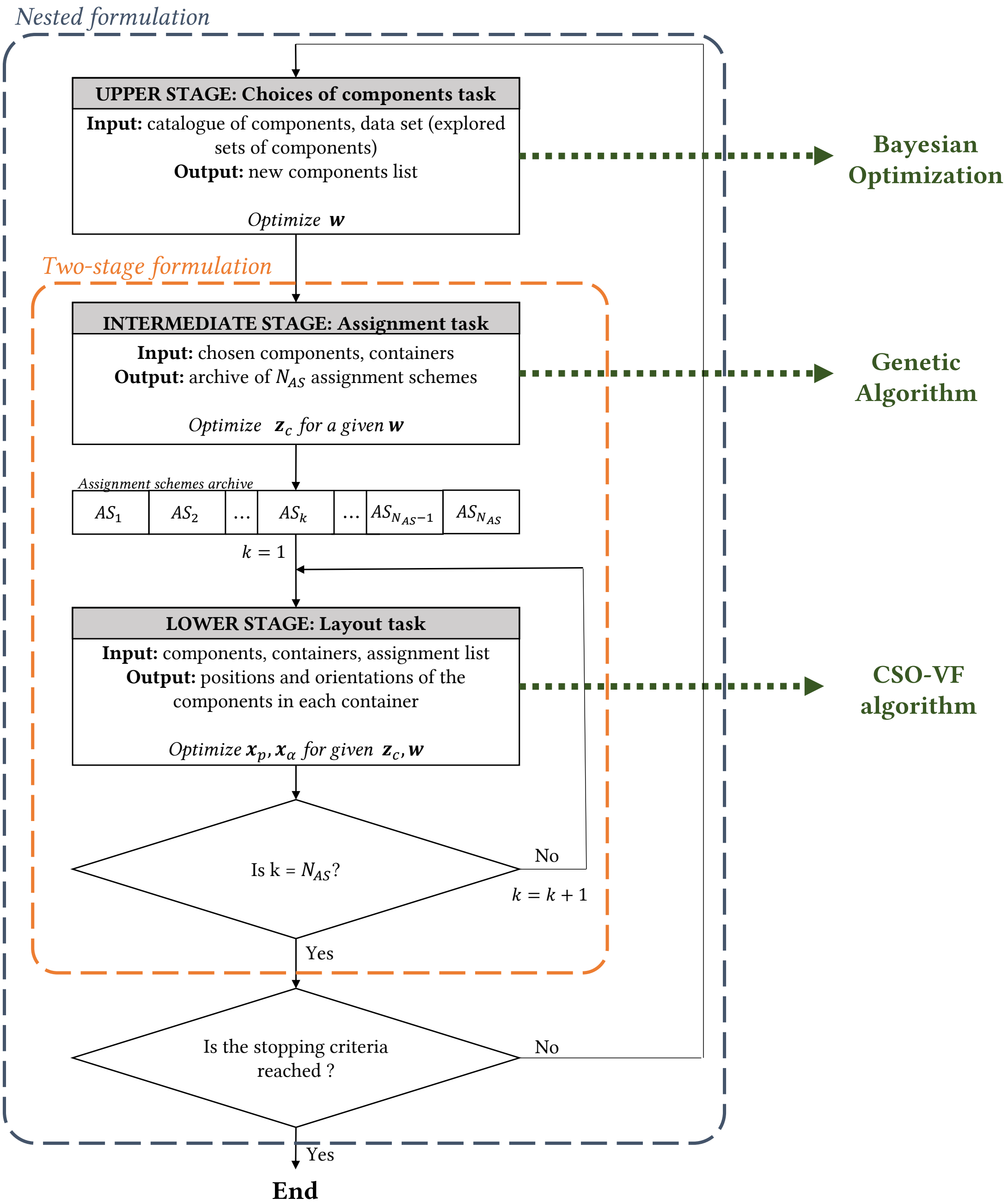


Figure 8.2: Tri-level algorithm description.

8.3.2 Upper level: Bayesian Optimization for categorical variables

The Bayesian Optimization framework used to deal with the conditional categorical variables is similar to the one employed in Chapter 7 for CSS single-container layout problem. The configuration is summarized:

- **Computational budget:** the size of initial training data set is $N_{DoE} = 50$ and the number of iterations is $N_{it} = 100$;

- **Kernel:** the categorical Hamming kernel is employed;
- **Category-wise and level-wise categorical kernels:** for the same reasons as the CSS single-container satellite layout problem, the level-wise approach is chosen;
- **Noisy-Modeling** is employed as the two-stage GA+CSO-VF algorithm outputs depend on the initialization of both intermediate and lower stages. The noisy data modeling approach requires the definition of a nugget set to 10;
- **Gaussian Process training:** it is handled using the Adam Optimizer run during 50000 iterations;
- **Infill criteria:** the EI infill criteria is considered;
- **Infill criteria optimization:** it is handled with the help of a Genetic Algorithm which settings are determined by a parametric analysis: population of 200 individuals, 500 generations, probability of crossover: 0.9, probability of mutation: 0.1.

8.3.3 Intermediate level: Genetic Algorithm

The intermediate level is dedicated to assigning the chosen components to one of the containers. The corresponding mathematical problem at hand is the same as the one solved for fixed search (FSS) multi-container OLPs and detailed in Chapter 3 and Chapter 4. Genetic Algorithm is used for the same reasons as for the aforementioned FSS application cases. The Genetic Algorithm output is $N_{AS} = 10$ assignment schemes (*i.e.*, the N_{AS} best different individuals of the GA, sorted by objective function values). The settings of the Genetic Algorithm are optimized using Bayesian Optimization. In the FSS multi-container satellite problem, the assignment schemes were sequentially used until the lower level returned a feasible layout. In this chapter, all the N_{AS} assignment schemes are laid out thanks to the lower level. The best overall layout in terms of final objective function corresponds to the output of the two-stage part of the algorithm. It must be noted that this level is not computationally expensive especially because very few constraints are solved at this stage.

8.3.4 Lower level: CSO-VF algorithm

The lower level solves the layout task taking as inputs the list of the chosen components given by the upper level and the corresponding current assignment scheme of the archive of the intermediate level. This stage takes as an input the assignment list optimized by the upper stage. The outputs are the lists of positions and orientations of each component in its assigned container. It is considered in this thesis that as for the fixed search space multi-container OLPs, the layout of each container can be solved independently. The CSO-VF algorithm described in Chapter 3 is chosen. Multistart is employed in order to increase the robustness to the initialization of the approach, as described in Chapter 4. 10 CSO-VF instances are run in parallel for each assignment scheme. Moreover, as described in Chapter 4, the hyperparameters as well as the number of iterations depend on the occupation rate of the container. As mentioned in the previous section, the output is the best obtained layout in terms of objective function value.

The global stopping criteria of the tri-level algorithm corresponds to the computational budget of the Bayesian Optimization (BO) process and thus to the given number of BO iterations.

8.3.5 Algorithm framework

Figure 8.3 illustrates the proposed tri-level algorithm framework. The steps related to the upper level and thus to the Bayesian Optimization process are highlighted in blue. The orange frame highlights the two-stage parts of the algorithm.

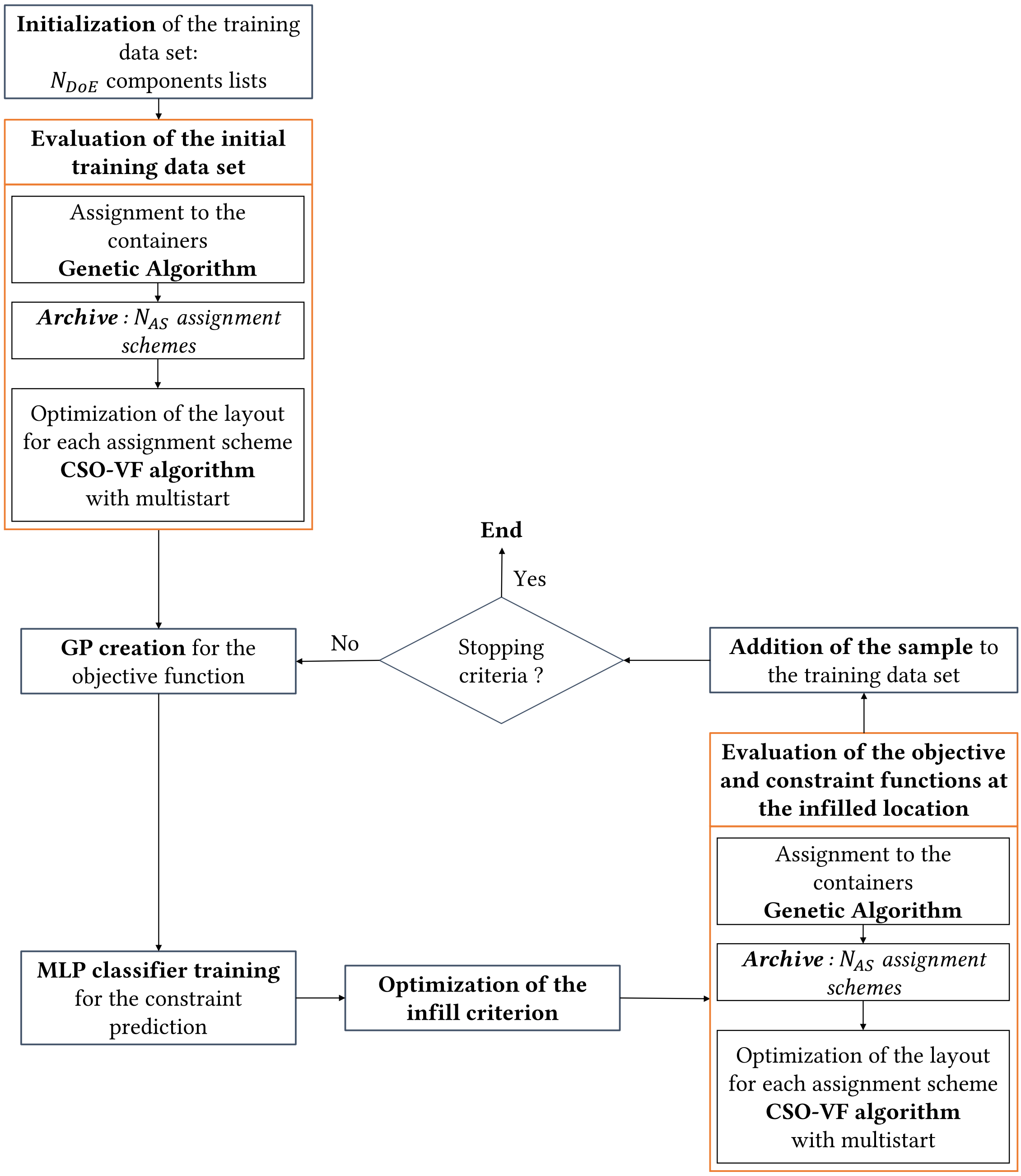


Figure 8.3: Tri-level algorithm framework.

8.4 Application to conditional search space multi-container satellite module layout problem

8.4.1 Conditional search space multi-container satellite module layout problem formulation

In this section, the conditional search space (CSS) single-container OLPs detailed in Chapter 7 are derived in the case of a multi-container configuration. The problem at hand involves positioning N generic components, each having n_i ($i \in \{1, ..., N\}$) possible subdivisions, within n_c containers in order to minimize the global inertia of the whole module. Several constraints must be satisfied that can be either geometrical (*e.g.*, overlapping constraints, balancing constraints) or functional (*e.g.*, constraints of functional compatibility between components).

8.4.1.1 Geometry of the container and the components

Container. In the multi-container configuration and similarly to the fixed search space satellite layout problem formulation, the container corresponds to the simplified module of the INTELSAT-III satellite introduced in Chapter 4 [CZa19; WTS09b; ZTS08]. Two two-sided bearing plates (*i.e.*, four surfaces in total) are defined by their outer radius R_{out} and a cylindrical exclusion zone (*e.g.*, meant to represent a central bus) is defined by its radius R_{in} and centered at the geometrical center of the plates.

Components. The generic components, N_{cyl} cylinders and N_{cub} cuboids, are defined by their dimensions and masses. They are considered as rigid bodies of homogeneous density and are located with the help of the position of their centers of inertia. For each generic components, n_i subdivisions are defined. In this chapter, three assumptions are made:

- The geometry of the generic component (*i.e.*, cylinder or cuboid) is conserved for all the corresponding subdivisions;
- The total area of the components of each subdivisions of a same generic component is conserved. It is mathematically translated as follows:

$$\sum_{k=1}^{n_{i,j}^{subdiv}} A_i^{j,k} = \sum_{k=1}^{n_{i,j+1}^{subdiv}} A_i^{j+1,k}, \quad \text{for } i \in \{1, ..., N\}, \quad \text{for } j \in \{1, ..., n_i - 1\} \quad (8.3)$$

  where $A_i^{j,k}$ is the area of component $\{i, j, k\}$.
- In this chapter, the total mass of the components is not conserved. Indeed, in order to describe more realistically the problem, the assumption is made that when the size of the components is subdivided, the mass of cylindrical (resp. cuboids) components is penalized of 10% (resp. 15%) with respect to

a conservation of the mass assumption. As a consequence, the more a generic components is subdivided, the heavier is the total mass.

The configuration designed for the application case consists in $N = 37$ generic components. Among them, $N_s = 7$ components can be subdivided so that a maximum of 60 components are positioned in the container. It must be noted that the configuration with 60 components *i.e.*, the configuration corresponding to the maximum number of subdivisions, corresponds to the list of components used for the fixed search space multi-container satellite configuration addressed in Chapter 4. Overall, it corresponds to **1728 possible different lists of components**. Moreover, due to the assumption of non-conservation of mass between the subdivisions of a same generic component, the total mass of the components vary between 764.7 and 815.5 kg.

The details of the generic components and their subdivisions as well as the containers dimensions and geometrical features used in this chapter are reported in Table A.5 in Appendix A.

8.4.1.2 Design variables

In the conditional search space multi-container satellite configuration the design variables are:

- The chosen subdivisions of the generic components considered as conditional categorical design variables $\mathbf{w} = \{w_i\}$ where $i \in \{1, ..., N\}$ and $w_i \in \{1, ..., n_i\}$;
- The positions of the centers of inertia of the chosen components by the conditional variables, considered as continuous variables, $\mathbf{x}_p$. The length of vector $\mathbf{x}_p$ depends on the conditional variables, $n_{x_p}(\mathbf{w})$;
- The orientations of the chosen cuboids components considered as discrete variables $\mathbf{z}_\alpha$. The length of vector $\mathbf{z}_\alpha$ on the conditional variables, $n_{z_\alpha}(\mathbf{w})$;
- The attributed plates to each chosen component considered as discrete, categorical variables, $\mathbf{z}_c$. The length of vector $\mathbf{z}_c$ depends on the conditional variables, $n_{z_c}(\mathbf{w})$

Thus, the number of variables to optimize depends on the values of the conditional variables w_i, $i \in \{1, ..., N\}$. The maximum number of design variables n_{var} used to define a layout is defined as:

$$n_{var} = N_s + \sum_{i \in CYL} 3 \max_j(n_{ij}^{subdiv}) + \sum_{i \in CUB} 4 \max_j(n_{ij}^{subdiv}) \tag{8.4}$$

where CYL and CUB respectively referred to as the i indices of cylinder and cuboid components. In other words, for $i \in \{1, ..., N\}$, $i \in CYL$ if component i is a cylinder or $i \in CUB$ if component i is a cuboid.

For instance, in the case of the most subdivided components, $N = 60$ components must be positioned on the four bearing surfaces, $N_s = 7$ components are subdivided in 36 cylinders and 24 cuboids. This corresponds to 211 design variables.

8.4.1.3 Objective and constraints functions

Objective function. The objective function to minimize is the global inertia of the module I_{tot} which is mathematically defined in Chapter 4 for the fixed search space multi-container satellite layout problem. In this chapter, the inertia is function of the positions and orientations of the components selected as a result of the conditional variables as well as the corresponding assigned surfaces.

Constraints functions. The constraints functions are identical to the ones listed and mathematically defined in Chapter 4 and can be split into two categories: the constraints respectively related to the assignment and to the layout tasks. No particular constraints are considered for the choices of the component task. The assignment constraints are briefly summarized:

- Balancing constraint along the z-axis, g^z_{CG};
- Occupation rate constraint for each surface, g^j_O $(j \in \{1, ..., 4\})$.

The layout constraints are also briefly summarized:

- Overlapping between the components, $\mathbf{h}^{Comp}_{overlap}$;
- Overlapping between the components and the container, $\mathbf{h}^{Cont}_{overlap}$;
- Overlapping between the components and the exclusion zones, $\mathbf{h}^{E}_{overlap}$;
- Balancing constraint, g_{CG};
- Angle of inertia constraint, g_{AI};
- Functional constraint of distance between fuel and energy components, $\mathbf{h}_{functional}$.

In this chapter, the constraint functions take as arguments the positions and orientations of the selected components based on the conditional variables as well as the corresponding assigned surfaces.

8.4.1.4 Mathematical formulations

As written with Equations 8.2, a combination of a nested and two-stage formulations is proposed in order to deal with the choices of the components, the assignment and the layout tasks in three sequential optimization processes. The nested formulation is used in Chapter 6 and Chapter 7 to deal with the choice of components and layout tasks while the two-stage formulation is used in Chapter 3 and Chapter 4 to deal with the assignment and layout tasks.

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & I_{tot}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) \\
\text{w.r.t.} \quad & \mathbf{w} \in F_w \subseteq \mathbb{N}^{n_w} \\
\text{s.t.} \quad & g_{CG}^{z\,*}(\mathbf{z}_c^{*}, \mathbf{w}) \leq 0 \\
& g_O^{j\,*}(\mathbf{z}_c^{*}, \mathbf{w}) \leq 0 \quad \text{for } j \in \{1,2,3,4\} \\
& \mathbf{h}_{overlap}^{Comp\,**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{Cont\,**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& \mathbf{h}_{overlap}^{E\,**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& \mathbf{h}_{functional}^{**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& g_{AI}^{**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) \leq 0 \\
& g_{CG}^{**}(\mathbf{x}_p^{**}, \mathbf{z}_\alpha^{**}, \mathbf{z}_c^{*}, \mathbf{w}) \leq 0 \\
& \{\mathbf{z_c}^{*}\} = \operatorname{argmin} \; I_{tot}^{assignment}(\mathbf{z}_c, \mathbf{w}) \\
& \text{w.r.t.} \quad \mathbf{z}_c \in F_{z_c} \subseteq \mathbb{N}^{n_{z_c}(\mathbf{w})} \\
& \quad \text{s.t.} \quad g_{CG}^{z}(\mathbf{z}_c, \mathbf{w}) \leq 0 \\
& \qquad\quad g_O^{j}(\mathbf{z}_c, \mathbf{w}) \leq 0 \quad \text{for } j \in \{1,2,3,4\} \\
& \qquad\quad \{\mathbf{x_p}^{**}, \mathbf{z}_\alpha^{**}\} = \operatorname{argmin} \; I_{tot}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) \\
& \qquad\quad \text{w.r.t.} \quad \mathbf{x}_p \in F_{x_p} \subseteq \mathbb{R}^{n_{x_p}(\mathbf{w})}, \mathbf{z}_\alpha \in F_{z_\alpha} \subseteq \mathbb{N}^{n_{z_\alpha}(\mathbf{w})} \\
& \qquad\qquad \text{s.t.} \quad \mathbf{h}_{overlap}^{Comp}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) = 0 \\
& \qquad\qquad\qquad \mathbf{h}_{overlap}^{Cont}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) = 0 \\
& \qquad\qquad\qquad \mathbf{h}_{overlap}^{E}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c^{*}, \mathbf{w}) = 0 \\
& \qquad\qquad\qquad \mathbf{h}_{functional}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) = 0 \\
& \qquad\qquad\qquad g_{AI}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) \leq 0 \\
& \qquad\qquad\qquad g_{CG}(\mathbf{x}_p, \mathbf{z}_\alpha, \mathbf{z}_c, \mathbf{w}) \leq 0
\end{aligned}
\tag{8.5}
$$

The design variables respectively denoted with a two-star exponent and a one-star exponent (*e.g.*, $\mathbf{x}^{**}$ and $\mathbf{x}^{*}$) referred to as the design variables respectively optimized in the lower and intermediate levels and which are thus fixed in the upper level. In the same way, the constraints respectively denoted with a two-star exponent and a one-star exponent (*e.g.*, $\mathbf{h}^{**}(\cdot)$ and $\mathbf{h}^{*}(\cdot)$) referred to as the constraints respectively handled in the lower and intermediate levels but which must ensure the feasibility of the layout for the list of components defined by $\mathbf{w}$ at the upper level.

8.4.2 Application of the tri-level algorithm to the multi-container satellite layout problem

The tri-approach is run over 20 independent training data sets. Figure 8.4 shows the median and interquartile range of the convergence curves for the 20 repetitions and Table 8.1 reports the numerical results. Figure 8.5 shows the optimal layout of the four surfaces.

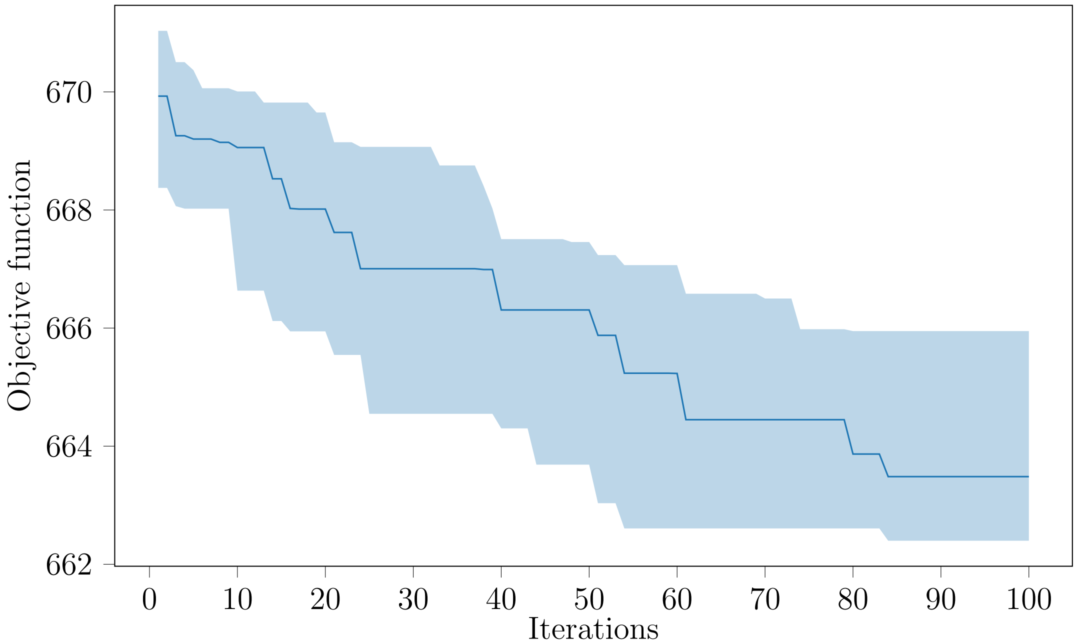


Figure 8.4: Median and interquartile range (IQR) of the Bayesian Optimization convergence curves (20 repetitions).

Metric	Value
Final median	663.48
Final interquartile range	3.55
Best run	661.91
Worst run	668.45
Mean of number of components	42

Table 8.1: Numerical results.

It is observed that the proposed algorithm converges toward lists of components that provide feasible layouts and contribute to decrease the global inertia. It is worth noting that the computational budget of the Bayesian Optimization upper level leads to evaluate 150 lists of components over the 1728 possible ones. This corresponds to explore less than 10% of the conditional search space. Thus, increasing the computational budget should improve the convergence abilities of the approach.

The advantage of the proposed tri-level algorithm is that it deals separately with each type of design variables (*i.e.*, conditional categorical variables, dependent categorical variables related to the assignment task and depending continuous and discrete variables related to the layout task) thanks to three adapted techniques that have been selected throughout this thesis. The algorithm was successfully applied to the complex benchmark of conditional search space multi-container satellite layout problem.

The main limitation is consequently that this approach is limited to OLPs in which the aforementioned variables can be handled separately in different optimization process. More generally, this algorithm benefits from the advantages of the three techniques used at each level but also possesses the same limitations.

8.5 Conclusions of the second part

This second part of the manuscript was devoted to conditional search space optimal layout problems (CSS OLPs). In Chapter 5, a survey of existing methods for solving CSS OLPs has been conducted. The advantages and limits of the methods have been highlighted, which lead to particularly focus on approaches based on metaheuristics, Bayesian Optimization as well as bilevel techniques. This range of methods seemed promising in order to address conditional search space OLPs characterized by a large number of possible conditional variables vectors.

Consequently, Chapter 6 focused on formulating the CSS OLPs at hand. Two approaches have been devised for solving this type of problems. First of all, a derivation of a hidden-variable mechanism for Genetic Algorithm has been adapted. This algorithm allows to deal with the CSS aspect of the problem with the help of two proposed implementations of conditional variables which reveal or hide the depending variables that are respectively used or ignored to compute the objective and constraint functions.

The second approach is a bilevel surrogate assisted-CSO-VF algorithm which involves an hybridization of the CSO-VF algorithm described and assessed in Chapter 3 and Chapter 4 with categorical Bayesian Optimization to deal with the CSS aspect of the problem. In this approach, the conditional and depending variables are handled in two nested optimization processes. The upper level consists in optimizing the list of components using categorical Bayesian Optimization based on Gaussian Process surrogate modeling while the CSO-VF algorithm is responsible of optimizing the layout of the input lists of components at a lower level.

Chapter 7 extended the fixed search space single-container Satellite Module Layout Problems used as a benchmark in Part I of the manuscript in order to integrate the choices of the components task and consequently a conditional search space. This benchmark has been used in order to configure both algorithm and assess their functioning. Their global performance was subsequently compared. As a result, the surrogate assisted-CSO-VF algorithm provides better performance notably in terms of success rate, convergence accuracy, robustness to initialization in comparison to the hidden-variable mechanism for Genetic Algorithm. Indeed, the Bayesian Optimization technique at the upper level has strong ability to identify the promising areas of the conditional search space while the CSO-VF algorithm at the lower level provides good quality layouts thanks to constraints handling dedicated operators.

Finally, the aforementioned benchmark problem was extended to multi-container configurations in the opening Chapter 8. A tri-level approach combining Bayesian Optimization, Genetic Algorithm and the CSO-VF algorithm has been devised for solving the CSS multi-container satellite layout problem. It has been successfully applied to the benchmark at hand, drawing on the strength of each of the dedicated methods chosen for each level throughout this thesis.

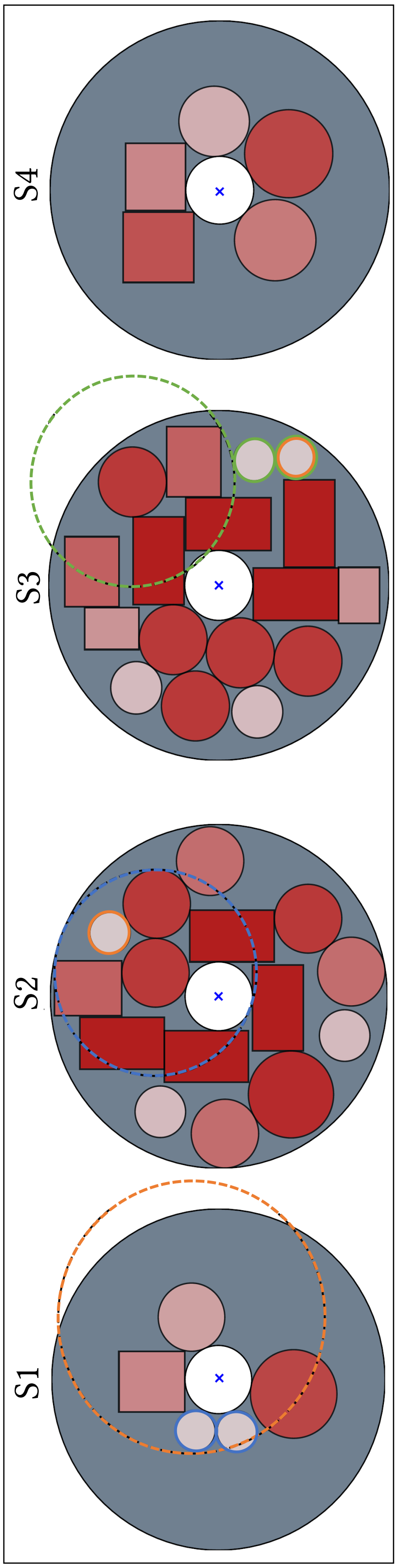


Figure 8.5: Best final layout obtained with the tri-level algorithm. The redder the component, the higher their corresponding masses. The dotted lines correspond to functional safety zones that must not be violated by the components highlighted in the same color.

Chapter 9

Conclusions and perspectives

9.1 Conclusions

This thesis is focused on the development of algorithms for optimal layout problems (OLPs). In the first part of the thesis, OLPs characterized by a fixed set of components referred to as fixed search space problems are addressed. Methods are devised for solving mixed-variable highly constrained fixed search space OLPs with both single-container and multi-container configurations. In the second part of the thesis, the aforementioned OLPs are extended in order to take into account the choice of components during the optimization process. As a result, the number of components and thus the number and type of design variables and constraints vary during the optimal layout process. Approaches are developed in order to deal with those problems referred to as conditional search space OLPs, also characterized by both single and multi-container configurations. This thesis focused on different types of algorithms, such as evolutionary, quasi physical and Bayesian optimization in order to efficiently solve the OLPs. These algorithms have also been hybridized in order to tackle specific challenges of this problem such as the presence of numerous constraints or mixed variables. The contributions related to both category of problems are summarized in the following paragraphs.

The first part of the thesis is devoted to fixed search space optimal layout problems (FSS OLPs). In Chapter 2, existing methods for FSS OLPs are reviewed and some techniques were identified as promising for solving mixed-variable constrained fixed search space OLPs. Among them, quasi-physical algorithms based on virtual-force systems are techniques having dedicated operators for constraint handling as well as an inherent decentralized system which allows to deal with a high number of components.

Then, in Chapter 3, a new quasi-physical approach based on a virtual-force system, named Component Swarm Optimization based on Virtual-Force system (CSO-VF) algorithm, has been devised for solving generic OLPs corresponding to extended packing problems which have been previously tackled in the literature by similar methods. The proposed virtual-force system and operators of the algorithms allow to deal with OLPs characterized with up to several hundreds of heterogeneous components leading to several hundreds or thousands of design variables and constraints.

It has been applied to satellite module layout problems in Chapter 4, which constitute representative benchmarks of OLPs, as well as balanced circular bin packing problems in Appendix E. These application cases are notably characterized by problems with 10 to 300 components (*i.e.*, up to 600 design variables), a high number of constraints (*e.g.*, overlapping, balancing, functional, *etc.*) and different objective functions (*i.e.*, size of the container, inertia of the system). Different occupation rates up to 60% of the container have been studied in order to progressively raise the difficulty to solve the constraints.

The CSO-VF algorithm has been published in an international conference communication :

Gamot, J., Wuilbercq, R., Balesdent, M., Tremolet, A., Melab, N., & Talbi, E. G. (2022, October). Component Swarm Optimization Using Virtual Forces for Solving Layout Problems. In International Conference on Swarm Intelligence (pp. 292-299). Cham: Springer International Publishing.

However, the CSO-VF algorithm described in this thesis only allows to deal with single-container OLPs. Sometimes, OLPs encompass a multi-container configuration in which the given components must be positioned in several containers. Therefore, the components need to be assigned to a container while optimizing the containers' layout. Solving both assignment and layout tasks in the same optimization process may suffer from poor convergence capabilities due to the high number of additional design variables and constraints related to the assignation tasks which induce a very highly combinatory design space. Consequently, in this thesis, a two-stage algorithm framework was proposed in order to solve both assignment and layout tasks sequentially, in the case where the design variables related to the assignment and to the layout are separable. The assignment task is solved thanks to a Genetic Algorithm in an upper stage. Subsequently, the layout of each container is optimized thanks to the CSO-VF algorithm as they correspond to single-container OLPs. This algorithm has been successfully applied to multi-container satellite module layout problems characterized with 60 components (*i.e.*, 276 mixed design variables), 4^{60} possible assignment schemes and hundreds of constraints. The main obtained experimental results outperformed previously published results.

In a second part of this manuscript, conditional search space optimal layout problems (CSS OLPs) are addressed. Specifically, the choice of the components must be optimized along with the layout of the chosen components. The choice of components introduces conditional categorical variables. The design variables related to the layout of chosen components are depending continuous and discrete variables. In Chapter 5, existing methods for CSS problems are reviewed. It has been highlighted that very few OLPs included a conditional search space. Some techniques were identified as promising for solving CSS OLPs. Among them, derivations of metaheuristic techniques are the most employed and often provide a generic framework. Moreover, Bayesian Optimization generally benefits from a high convergence speed. Finally, the inherent structure of bilevel approaches allows to deal with conditional and depending variables with two adapted sequential optimization processes.

Subsequently, based on the previous reviews, two approaches have been developed for solving CSS OLPs. The first approach is a derivation of a hidden-variable mechanism for Genetic Algorithm (HVM for GA). The HVM for GA modifies the implementation of the chromosomes in order to reveal or hide some of the depending genes that are respectively taken or not taken into account for the objective or constraint functions evaluations depending on the conditional variables. Thus, this method allows to tackle all design variables in a single optimization process. This method provides a generic and versatile framework, requires few hyperparameters and few adaptations of the classical evolutionary operators. The HVM for GA approach has been the subject of an international journal paper :

Gamot, J., Balesdent, M., Tremolet, A., Wuilbercq, R., Melab, N., & Talbi, E. G. (2023). Hidden-variables genetic algorithm for variable-size design space optimal layout problems with application to aerospace vehicles. Engineering Applications of Artificial Intelligence, 121, 105941.

The second approach consists of a bilevel surrogate assisted-Component Swarm Optimization based on Virtual-Force (CSO-VF) algorithm. It combines the strength of the CSO-VF algorithm developed in Part I, and a discrete Bayesian Optimization algorithm framework purposely adapted to tackle the categorical aspect of this problem. This surrogate assisted-CSO-VF algorithm thus consists in two nested optimization processes dealing respectively with the optimization of the set of components (*i.e.*, the conditional variables) and with the optimization of the corresponding layout (*i.e.*, the depending variables). This algorithm benefits from its inherent bilevel structure which allows to study each selected list of components in depth and to identify the area of the conditional search space proving the most promising lists of components in terms of final layout. It also benefits from the strength of each algorithm purposely chosen at each level: the convergence abilities of the Bayesian Optimization and the constraint handling abilities of the CSO-VF algorithm. The surrogate assisted-CSO-VF approach has been the subject of an international conference communication :

Gamot, J., Balesdent, M., Wuilbercq, R., Tremolet, A., Melab, N., & Talbi, E. G. (2023, July). Two-Level Algorithm Combining Bayesian Optimization and Swarm Intelligence for Variable-Size Optimal Layout Problems. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (pp. 139-142).

Both algorithms have been configured and assessed thanks to a single-container satellite layout problem benchmark, extended to take into account the CSS aspect. This benchmark is notably characterized with 3888 possible lists of components. As a result, both approaches show abilities to solve the problem. However, the surrogate assisted-CSO-VF algorithm provides better performance notably in terms of success rate, convergence accuracy and robustness to initialization in comparison to the HVM for GA, for a given computational budget.

The surrogate assisted-CSO-VF algorithm cannot deal with multi-container configurations of the problem. Indeed, such problems add depending categorical design variables which characterize the assignment of the chosen components to the containers. However, the CSO-VF cannot deal with multi-container configurations. Consequently, the last approach developed in this thesis is a tri-level algorithm combining Bayesian Optimization, Genetic Algorithm and the CSO-VF algorithm for solving multi-container CSS OLPs. This approach combines the nested formulation of the surrogate assisted-CSO-VF algorithm developed in Chapter 6 with the two-stage formulation of the two-stage algorithm combining Genetic Algorithm and the CSO-VF algorithm developed in Chapter 3. As a result, the upper level deals with the choice of components task thanks to Bayesian Optimization, the intermediate level addresses the assignment of the chosen components to the containers thanks to Genetic Algorithm and the lower level solves the layout of the chosen components within the assigned containers thanks to the CSO-VF algorithm. The tri-level algorithm has been applied to a CSS multi-container satellite layout benchmark characterized with 1728 possible lists of components, 4^{36} to 4^{60} possible assignment schemes (depending on the selected components), as well as several hundreds of mixed variables and constraints. The algorithm shows ability to solve the problem, drawing on the strength of each of the dedicated methods chosen for each level throughout this thesis.

Figure 9.1 sums up the search space and container configuration addressed by the five algorithms devised in this thesis, as well as the Chapters related to each of them. The algorithms are summarized as follows:

- The Component Swarm Optimization algorithm based on a Virtual-force system (CSO-VF);
- The two-stage approach combining Genetic Algorithm and the CSO-VF algorithm (GA+CSO-VF);
- The hidden-variable mechanism for Genetic Algorithm (HVM for GA);
- The surrogate assisted-CSO-VF algorithm (BO+CSO-VF);
- The tri-level algorithm combining Bayesian Optimization, Genetic Algorithm and CSO-VF algorithm (BO+GA+CSO-VF).

		Part I		Part II		
		CSO-VF	**Two-stage GA+CSO-VF**	**HVM for GA**	**Bilevel BO+CSO-VF**	**Tri-level BO+GA+CSO-VF**
Search space	Fixed Search Space	X	X			
	Conditional Search Space			X	X	X
Container configuration	Single-container	X		X	X	
	Multi-container		X	X		X
	Chapters	3 and 4	3 and 4	6 and 7	6 and 7	8

Figure 9.1: Algorithms and characteristics of the problem solved.

9.2 Perspectives

Different improvements and extensions of the approaches developed in this thesis can be identified.

First of all, the CSO-VF algorithm can only deal with single-container configurations. Indeed, the algorithm is based on a virtual-force system which does not take into account the categorical assignment design variables. It could therefore be interesting to extend the proposed virtual-force system in order to give the algorithm the ability to assign the components to several containers.

Moreover, in this thesis, the actual algorithm devised to deal with multi-container configurations is a two-stage algorithm combining Genetic Algorithm and the CSO-VF algorithm. Thus, the optimization process is sequential and it could be interesting to try a nested approach inspired from the surrogate assisted-CSO-VF algorithm devised for conditional search space. However, in its original form, Bayesian Optimization allows to deal with only a few dozens variables and the satellite benchmark problem considered in order to assess the method involves 60 components and thousands of combinations. Consequently, if the Bayesian Optimization technique is chosen as an upper level to deal with the assignation of the components to the containers, it should be, beforehand, enhanced with kernels allowing for large-scale problems [Sav+22].

Additionally, the CSO-VF algorithm has been applied to satellite module layout problems and to balanced circular bin packing problems. It could be interesting to extend the virtual-force systems for solving other category of problems such as wind farms layout problems, facility layout problems, coverage problems, *etc.*

Furthermore, one current limit of the surrogate assisted-CSO-VF algorithm dealing with CSS OLPs is the number of design variables handled by the Bayesian Optimization process which is restricted to a few dozens. It could be interesting to enhance the actual upper level of the method with large-scale kernels solutions [Sav+22] in order to deal with up to a hundred categorical variables. As many components could be addressed.

Moreover, the described algorithms are currently devoted to early design layout in which simplified descriptions and models of the concept are used. It would be interesting to test the proposed methods on more advanced concepts in terms of design process. A greater variety of components along with additional requirements could be introduced in order to improve the realism level of the layout. The algorithms could then be adapted for instance through extensions of the virtual-force system of the CSO-VF algorithm.

Then, the bilevel structure of the surrogate assisted-CSO-VF algorithm could be extended to other conditional search space application cases. Indeed, the Bayesian Optimization technique at the upper level of the proposed algorithm is used in order to explore efficiently the conditional search space. Subsequently, the dependent variables are handled at the lower level where any technique adapted to the problem at hand can be used. Consequently, this structure could be applied to any conditional search space problems as long as the depending variables (*i.e.*, the fixed search space subproblems) can be optimized thanks to a dedicated technique which would be implemented into the lower level.

Finally, it could be interesting to investigate other techniques or hybridizations for solving various optimal layout application cases, as for instance Reinforcement Learning [Mir+21], in order to overcome some limitations of the previous algorithm as for instance the lack of genericity of some of the proposed approaches with respect to the application cases.

Appendix A

Components configurations

A.1 Components in fixed search space optimal layout problems

A.1.1 Single-container configuration

The components used for fixed search space single-container satellite layout benchmark (Chapter 4) are listed in Table A.1. Dimension 1 corresponds to the radius of cylinder components. Dimension 1 and 2 correspond to the dimensions of the basis of the cuboid components.

Index	Geometry	Type	Dimension 1	Dimension 2	Height	Mass
1	Cylinder	Fuel	100		240	20
2	Cylinder	Fuel	100		240	15
3	Cylinder	Fuel	50		240	8
4	Cuboid	Energy	100	75	200	7.5
5	Cuboid	Energy	100	75	200	7.5
6	Cuboid	Energy	125	160	200	10
7	Cuboid	Energy	125	160	200	10
8	Cuboid	Diverse	150	50	200	5
9	Cuboid	Diverse	150	50	200	5
10	Cuboid	Diverse	100	50	200	6
11	Cuboid	Diverse	150	50	200	5
12	Cuboid	Diverse	150	150	200	12
13	Cuboid	Diverse	200	100	200	15
14	Cylinder	Diverse	50		240	5
15	Cylinder	Diverse	75		240	8
16	Cuboid	Diverse	100	120	200	10

Table A.1: Components for the fixed search space single-container satellite benchmark. Dimensions are in mm, masses in kg.

A.1.2 Multi-container configuration

The components used for fixed search space multi-container satellite layout benchmark (Chapter 4) are listed in Table A.2.

Table A.2: Components for the fixed search space multi-container satellite benchmark.

Index	Geometry	Dimension 1	Dimension 2	Height	Mass
1	Cuboid	250	150	250	28.13
2	Cuboid	250	150	250	28.13
3	Cuboid	250	150	250	28.13
4	Cuboid	250	150	250	28.13
5	Cuboid	250	150	250	28.13
6	Cuboid	250	150	250	28.13
7	Cuboid	250	150	250	28.13
8	Cuboid	250	150	250	28.13
9	Cuboid	200	160	200	19.2
10	Cuboid	200	160	200	19.2
11	Cuboid	200	160	200	19.2
12	Cuboid	160	120	250	15.36
13	Cuboid	160	120	250	15.36
14	Cuboid	160	120	150	8.64
15	Cuboid	160	120	150	8.64
16	Cuboid	160	120	150	8.64
17	Cuboid	150	100	100	5.40
18	Cuboid	150	100	100	5.40
19	Cuboid	150	100	100	5.40
20	Cuboid	150	100	100	5.40
21	Cuboid	150	100	100	5.40
22	Cuboid	150	100	100	5.40
23	Cuboid	150	100	100	5.40
24	Cuboid	150	100	100	5.40
25	Cylinder	100		250	23.56
26	Cylinder	100		250	23.56
27	Cylinder	100		250	23.56
28	Cylinder	100		250	23.56
29	Cylinder	100		250	23.56
30	Cylinder	100		250	23.56

Continued on next page

Table A.2: Components for the fixed search space multi-container satellite benchmark. (Continued)

Index	Geometry	Dimension 1	Dimension 2	Height	Mass
31	Cylinder	100		250	23.56
32	Cylinder	100		250	23.56
33	Cylinder	100		200	18.85
34	Cylinder	100		200	18.85
35	Cylinder	100		200	18.85
36	Cylinder	100		160	15.08
37	Cylinder	100		160	15.08
38	Cylinder	100		160	15.08
39	Cylinder	75		160	8.48
40	Cylinder	75		160	8.48
41	Cylinder	75		160	8.48
42	Cylinder	75		160	8.48
43	Cylinder	75		150	7.95
44	Cylinder	75		150	7.95
45	Cylinder	75		150	7.95
46	Cylinder	75		150	7.95
47	Cylinder	75		150	7.95
48	Cylinder	75		150	7.95
49	Cylinder	60		150	5.09
50	Cylinder	60		150	5.09
51	Cylinder	60		150	5.09
52	Cylinder	60		150	5.09
53	Cylinder	60		150	5.09
54	Cylinder	60		150	5.09
55	Cylinder	60		150	5.09
56	Cylinder	60		150	5.09
57	Cylinder	60		150	5.09
58	Cylinder	60		150	5.09
59	Cylinder	60		150	5.09
60	Cylinder	60		150	5.09

The functional constraint imposes that pairs of components must be positioned apart from a certain distance. The indices of the pairs of components, their types and the minimal distance between them is reported in Table A.3.

Index 1	Index 2	Type	Distance
25	56	Heat	200
25	60	Heat	200
29	49	Heat	200
29	55	Heat	200
37	55	Electromagnetic	300
37	58	Electromagnetic	300

Table A.3: Functional constraint for the fixed search space multi-container satellite benchmark.

The geometrical and dynamical features of the container illustrated in Figure A.1 are summarized below.

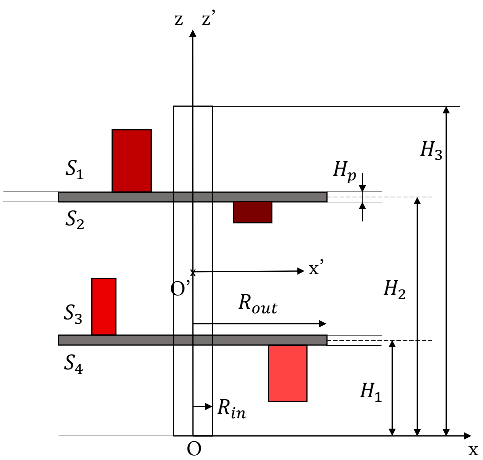


Figure A.1: Geometry of the satellite module.

$$H_1 = 300mm \tag{A.1}$$
$$H_2 = 830mm \tag{A.2}$$
$$H_3 = 1150mm \tag{A.3}$$
$$H_p = 20mm \tag{A.4}$$
$$R_{in} = 100mm \tag{A.5}$$
$$R_{out} = 500mm \tag{A.6}$$

Moreover, the empty module is defined by:

- Its mass: 576.53 kg;
- Its center of mass located at: $(0, 0, 553.56)$ in the $Oxyz$ system of coordinates;
- Its matrix of inertia:

$$I_0 = \begin{bmatrix} 352.2 & 0 & 0 \\ 0 & 352.2 & 0 \\ 0 & 0 & 106.8 \end{bmatrix}$$

A.2 Components in conditional search space optimal layout problems

A.2.1 Single-container configuration

The components used for conditional search space multi-container satellite layout benchmark (Chapter 7) are listed in Table A.4.

Index	Type	Geometry	Dim. 1	Dim. 2	Height	Mass	Subdivisions
1	Fuel	Cylinder	100		240	15	[1,2,3]
2	Fuel	Cylinder	100		240	8	[1,2]
3	Fuel	Cylinder	50		240	4	[1,2]
4	Energy	Cuboid	200	200	200	10	[1,2,3]
5	Energy	Cuboid	150	100	200	10	[1,2]
6	Diverse	Cuboid	100	100	200	8	[1,2,4]
7	Diverse	Cuboid	150	100	200	10	[1,2]
8	Diverse	Cuboid	150	150	200	12	[1,2,4]
9	Diverse	Cuboid	200	100	200	15	[1,2,3]
10	Diverse	Cuboid	100	75	200	6	[1]
11	Diverse	Cuboid	75	50	200	2	[1]
12	Diverse	Cuboid	200	100	200	15	[1]

Table A.4: Components for the conditional search space single-container satellite benchmark.

The subdivisions are written as [i,j,k] meaning that the generic component can be subdivided in three different ways: in i components or j components or k components. The dimensions of those sub-components are obtained such that the area and mass of the initial component are conserved.

A.2.2 Multi-container configuration

The components used for conditional search space multi-container satellite layout benchmark (Chapter 8) are listed in Table A.5.

Table A.5: Components for the conditional search space multi-container satellite benchmark.

Index	Geometry	Dimension 1	Dimension 2	Height	Mass	Subdivisions
1	Cuboid	250	150	250	28.13	[1]
2	Cuboid	250	150	250	28.13	[1]
3	Cuboid	250	150	250	28.13	[1]
4	Cuboid	250	150	250	28.13	[1]
5	Cuboid	250	150	250	28.13	[1]

Continued on next page

Table A.5: Components for the conditional search space multi-container satellite benchmark. (Continued)

Index	Geometry	Dimension 1	Dimension 2	Height	Mass	Subdivisions
6	Cuboid	250	150	250	28.13	[1]
7	Cuboid	250	150	250	28.13	[1]
8	Cuboid	250	150	250	28.13	[1]
9	Cuboid	200	160	200	19.2	[1]
10	Cuboid	200	160	200	19.2	[1]
11	Cuboid	200	160	200	19.2	[1]
12	Cuboid	160	120	250	15.36	[1]
13	Cuboid	160	120	250	15.36	[1]
14	Cuboid	207.85	207.85	200	20.09	[1,3]
15	Cuboid	198.27	177.34	137.37	15.61	[1,2,4]
16	Cuboid	198.27	177.34	137.37	15.61	[1,4]
17	Cylinder	100		250	23.56	[1]
18	Cylinder	100		250	23.56	[1]
19	Cylinder	100		250	23.56	[1]
20	Cylinder	100		250	23.56	[1]
21	Cylinder	100		250	23.56	[1]
22	Cylinder	100		250	23.56	[1]
23	Cylinder	100		250	23.56	[1]
24	Cylinder	100		250	23.56	[1]
25	Cylinder	173.21		200	48.07	[1,3]
26	Cylinder	126.49		200	25.64	[1,2]
27	Cylinder	100		160	15.08	[1]
28	Cylinder	150		160	27.48	[1,2,4]
29	Cylinder	129.90		150	20.27	[2,4,6]
30	Cylinder	120		150	16.288	[1,4]
31	Cylinder	103.92		150	12.98	[1,3]
32	Cylinder	60		150	5.09	[1]
33	Cylinder	60		150	5.09	[1]
34	Cylinder	60		150	5.09	[1]
35	Cylinder	60		150	5.09	[1]
36	Cylinder	60		150	5.09	[1]

The functional constraint imposes that pairs of components must be positioned apart from a certain distance. The indices of the pairs of components, their types and the minimal distance between them is reported in Table A.6.

Index 1	Index 2	Type	Distance
17	34	Heat	200
17	36	Heat	200
21	32	Heat	200
21	33	Heat	200
27	33	Electromagnetic	300
27	35	Electromagnetic	300

Table A.6: Functional constraint for the conditional search space multi-container satellite benchmark.

Appendix B

Inertia equations

B.1 Solid inertias

This appendix is devoted to the equations used to compute the solid inertias of cylinder and cuboids components in their local systems of coordinates.

Cylinders. For a cylinder component i, the following formalism is used: the radius is denoted r_i, the height h_i, the mass m_i, as illustrated on Figure B.1.

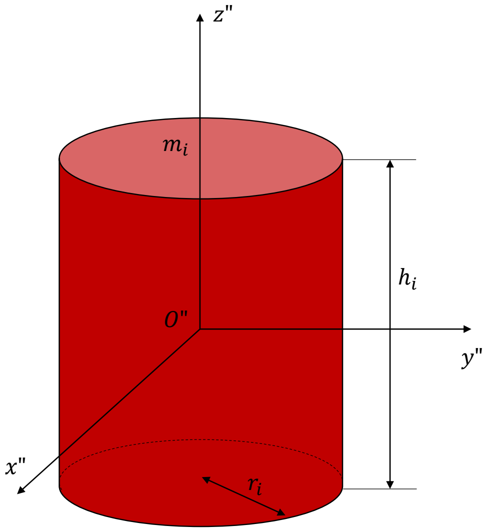


Figure B.1: Cylinder component i geometry.

With the formalism used on Figure B.1, the solid inertias in the local system of coordinates $O"x"y"z"$ are calculated as follows:

$$I_{x",i} = \frac{m_i}{12}(3r_i^2 + h_i^2) \tag{B.1}$$

$$I_{y",i} = \frac{m_i}{12}(3r_i^2 + h_i^2) \tag{B.2}$$

$$I_{z",i} = \frac{1}{2}m_i r_i^2 \tag{B.3}$$

Cuboids. For a cuboid component i, the following formalism is used: the dimensions of the basis are denoted a_i, b_i, the height h_i, the mass m_i, as illustrated on Figure B.2.

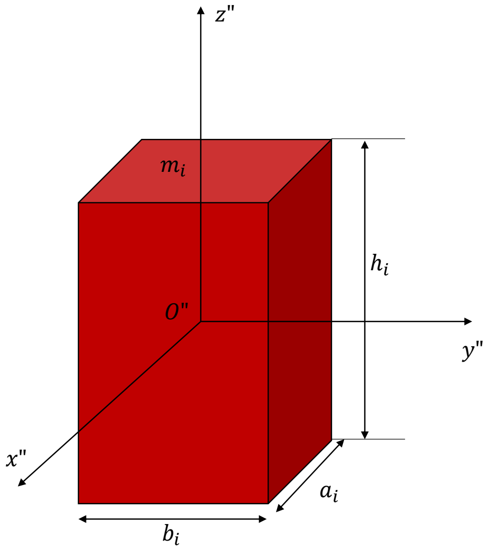


Figure B.2: Cylinder component i geometry.

With the formalism used on Figure B.2, the solid inertia along the three axis of the local system of coordinates $O"x"y"z"$ are calculated as follows:

$$I_{x",i} = \frac{m_i}{12}(b_i^2 + h_i^2) \tag{B.4}$$

$$I_{y",i} = \frac{m_i}{12}(a_i^2 + h_i^2) \tag{B.5}$$

$$I_{z",i} = \frac{m_i}{12}(a_i^2 + b_i^2) \tag{B.6}$$

B.2 Correction of inertia equations

As explained in Chapter 4, the actual inertia equations are expressed in the system of coordinates attached to the current centroid of the system. However, it has been shown that the equations should rather be expressed in the system of coordinates attached to either the theoretical centroid of the system or to the geometrical center of the container if no balancing constraints are considered.

With the simplified model of the INTELSAT-III considered as a benchmark in this thesis, the inertia should then be expressed in the system of coordinates with origin the center of mass of the empty module and denoted $O_{sat}x_{sat}y_{sat}z_{sat}$ as illustrated on Figure B.3.

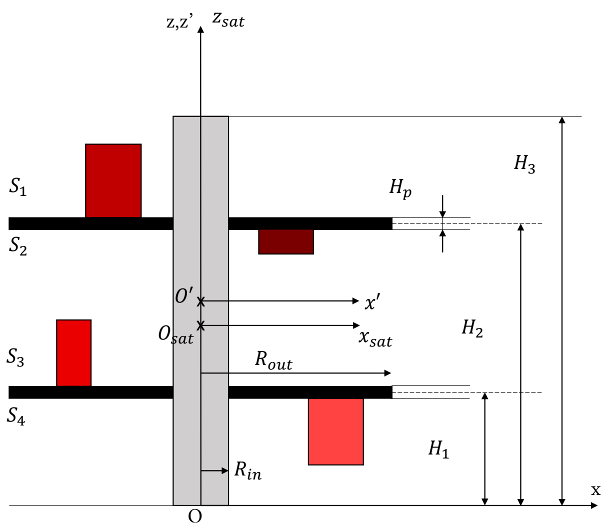


Figure B.3: Simplified model of the INTELSAT-III satellite module.

Thus, the inertia equations for a component i in the $O_{sat}x_{sat}y_{sat}z_{sat}$ system of coordinates are expressed as follows:

$$\begin{aligned}
I_{x_{sat},i} &= I_{x",i}\cos(x_{\alpha,i})^2 + I_{y",i}\sin(x_{\alpha,i})^2 &+& \quad m_i(x_{y,i}^2 + x_{z,i}^2) && \text{(B.7)}\\
I_{y_{sat},i} &= I_{y",i}\cos(x_{\alpha,i})^2 + I_{x",i}\sin(x_{\alpha,i})^2 &+& \quad m_i(x_{x,i}^2 + x_{z,i}^2) && \text{(B.8)}\\
I_{z_{sat},i} &= I_{z",i} &+& \quad m_i(x_{x,i}^2 + x_{y,i}^2) && \text{(B.9)}
\end{aligned}$$

where $\{x_{x,i}, x_{y,i}, x_{z,i}\}$ are the coordinates of the ith component's center of inertia, in the $O_{sat}x_{sat}y_{sat}z_{sat}$ system of coordinates. $x_{\alpha,i}$ corresponds to its orientation.

If N components have to be laid out in the container, the inertia of the module along each axis calculated at the point O_{sat} are expressed as follows:

$$I_x = I_{x_{sat},0} + \sum_{i=1}^{N} I_{x_{sat},i} \tag{B.10}$$

$$I_y = I_{y_{sat},0} + \sum_{i=1}^{N} I_{y_{sat},i} \tag{B.11}$$

$$I_z = I_{z_{sat},0} + \sum_{i=1}^{N} I_{z_{sat},i} \tag{B.12}$$

where $I_{x_{sat},0}, I_{y_{sat},0}, I_{z_{sat},0}$ are the inertia of the empty module calculated in its local system of coordinates which coincides with $O_{sat}x_{sat}y_{sat}z_{sat}$.

Appendix C

Configuration of algorithms

C.1 Configuration of the CSO-VF algorithm

C.1.1 Sensitivity analysis study

The sensitivity analysis study with the Sobol's index is reported in Chapter 4. Table C.1 sums up the bounds used for each hyperparameter of the CSO-VF algorithm.

Hyperparameter	Lower bound	Upper bound
F_{max}	150	500
T_{max}	10	100
v_{max}	1	20
ω_{max}	10	100
α_{CG}^{F}	10	1000
α_{I}^{F}	10	1000
α_{I}^{T}	0.01	10
p_{min}	10	100
p_{max}	50	500
r	1	10

Table C.1: Upper and lower bounds for the CSO-VF hyperparameters' sensitivity analysis with Sobol's indices.

C.2 Configuration of surrogate assisted-CSO-VF algorithm

C.2.1 Taguchi's plans of experiments

As described in Chapter 7, two Taguchi's plans of experiments are conducted in order to choose the best sequence of operators.

Definition of the Taguchi's plans of experiments. Table C.2 and Table C.3 report the operators used for each scheme of both Taguchi's plans of experiments.

Scheme	Constraint Handling	Selection	Crossover genes	Crossover tags	Mutation	Replace-ment
1	CD	T5	SBX	1 point	PM	NDT
2	CD	T5	SBX	2 points	UM	NDT
3	CD	T15	HUX	1 point	PM	NDT
4	CD	T15	HUX	2 points	UM	NDT
5	CD	T5	HUX	1 point	UM	NDT
6	CD	T5	HUX	2 points	PM	NDT
7	CD	T15	SBX	1 point	UM	NDT
8	CD	T15	SBX	2 points	PM	NDT

Table C.2: Taguchi table of the first set of schemes to run (10 runs each).

Scheme	Constraint Handling	Selection	Crossover genes	Crossover tags	Mutation	Replace-ment
9	CD	T15	SBX	1 point	PM	NDT
10	SR	T15	SBX	1 point	PM	NDT
11	CD	T15	SBX	1 point	PM	DC
12	SR	T15	SBX	1 point	PM	DC

Table C.3: Table of second set of schemes to run (10 runs each).

The Taguchi's plans are run for 3 different occupation rates: 30%, 50% and 70% of the occupied container. This represents 36 schemes and so 360 optimizations to run on the whole.

Results of the Taguchi's plans of experiments. Some analysis criteria to either quantify the exploitation or the exploration are defined in order to conclude on the performance of the operators:

- The final fitness value of the best individual of the last generation;
- The generation where the best individual fitness value of the current generation is at 10% of the final fitness value. It gives a valuable indicator on the convergence speed;
- The generation where the first feasible solution appears;
- The generation where the last non-feasible solution disappears;
- The number of tags explored amongst all the solutions (feasible or not);
- The number of tags explored amongst feasible solutions;
- The maximum number of tags which are different amongst one generation. Those tags information gives an idea of the exploration faculties of the algorithm.

Appendix D

Illustrations of the layout obtained thanks to the CSO-VF algorithm

Figures D.1 and D.2 show the feasible layouts obtained throughout one instance of the CSO-VF algorithm for a 40% occupation rate of the container.

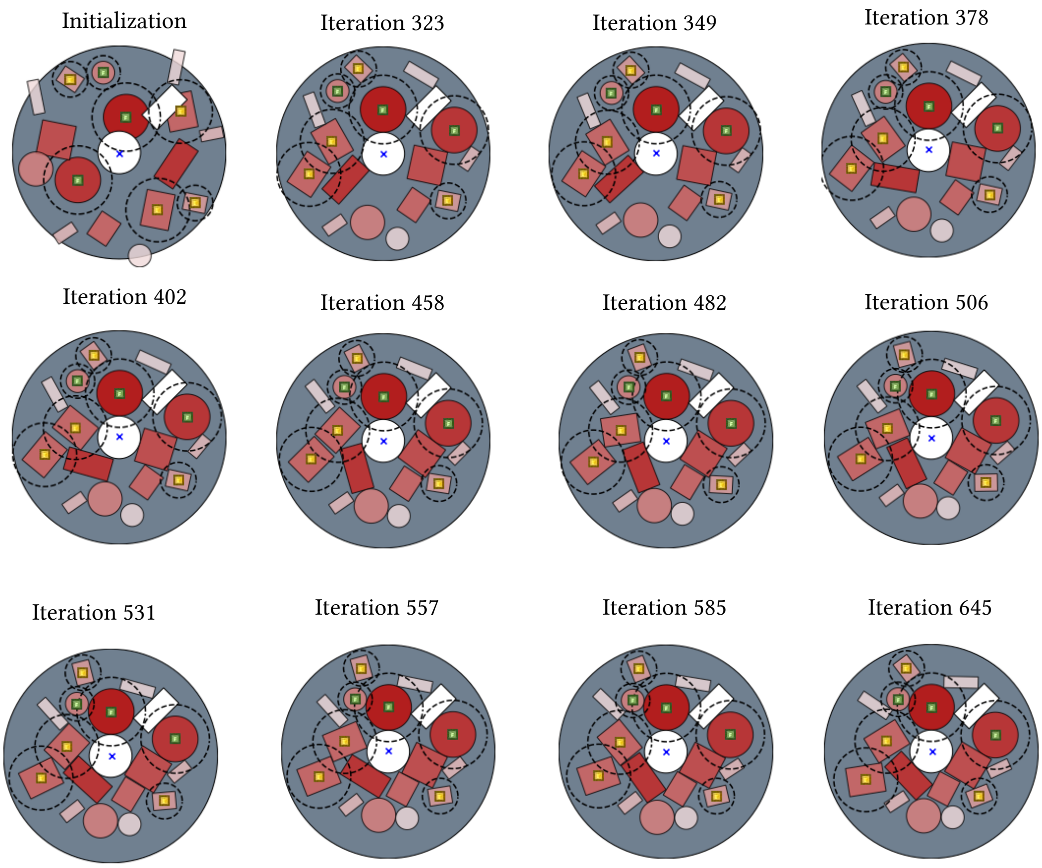


Figure D.1: Feasible layouts throughout convergence with occupation rate 40%, part 1.

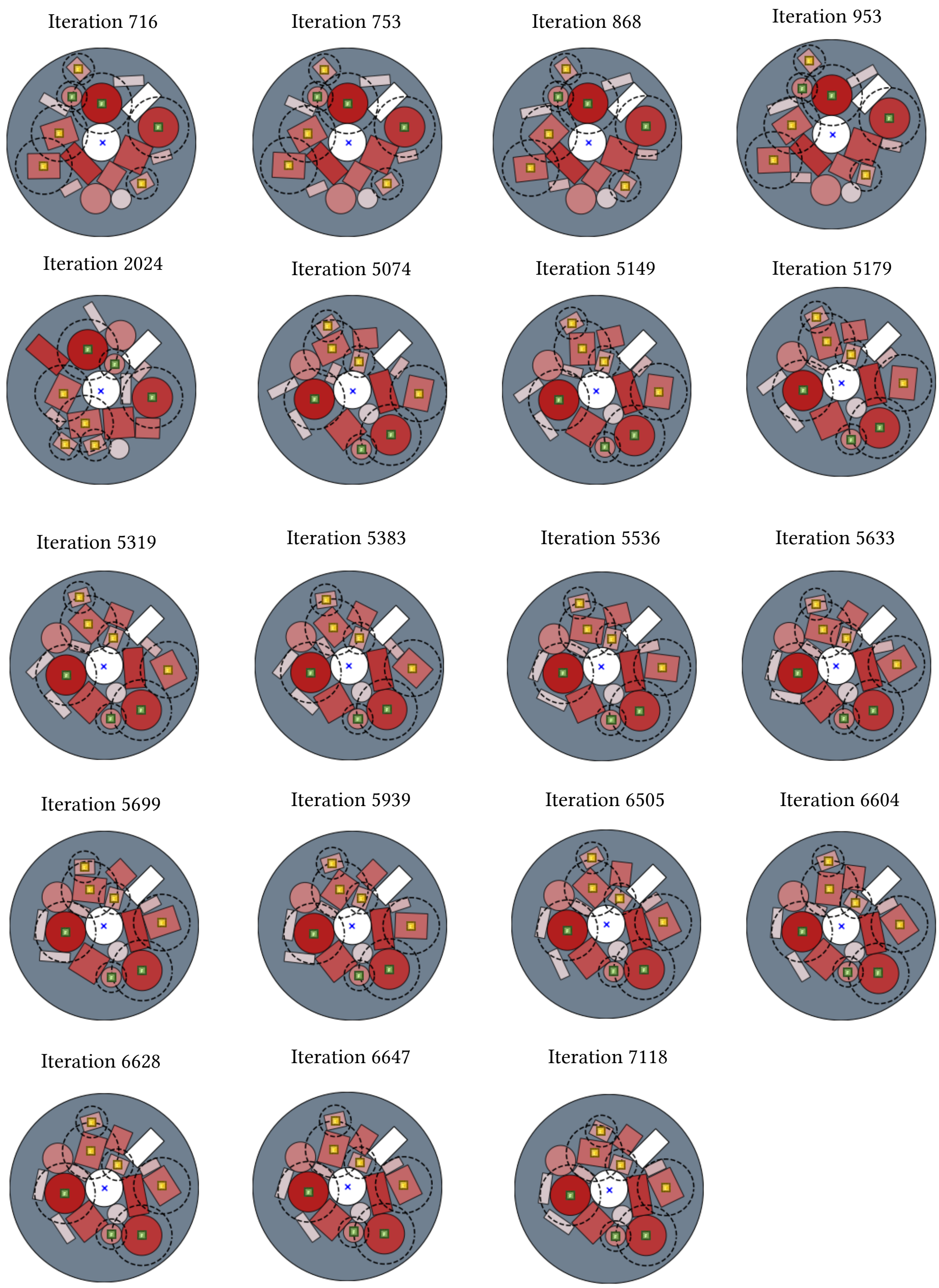


Figure D.2: Feasible layouts throughout convergence with occupation rate 40%, part 2.

Appendix E

CSO-VF algorithm for solving balanced circular bin packing problems

The CSO-VF algorithm have been used to solve balanced circular bin packing problems. An article has been published on Arxiv:

Gamot, J., Balesdent, M., Wuilbercq, R., Tremolet, A., Melab, N., & Talbi, E. G. (2023). A Virtual-Force Based Swarm Algorithm for Balanced Circular Bin Packing Problems. arXiv preprint arXiv:2306.01021.

The introduction of the paper as well as the state of the art section are not reported.

E.1 Formulation of the problem

The two-dimensional balanced circular bin packing problem considered in this paper is formulated as follows: pack a set of non-overlapping circles with given radii and masses, in the smallest circular container as possible, such that the center of gravity of the circles is located at the geometrical center of the container [XXA07a; TT99]. Let $C_i, i \in \{1, N\}$ be a set of N circles of radius r_i^c and mass m_i. Each circle C_i is located thanks to the coordinates of its center of inertia (x_i, y_i). The vectors $\mathbf{x}$ and $\mathbf{y}$ encompass the 2D cartesian coordinates of all the circles. The balanced circular bin packing problem can be mathematically formulated as follows:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & r(\mathbf{x},\mathbf{y}) = \max_{i\in\{1,N\}} \left(\sqrt{x_i^2+y_i^2} + r_i^c\right) \\ \text{where} \quad & \mathbf{x} = \{x_1, ..., x_N\} \in \mathbb{R}^N, \mathbf{y} = \{y_1, ..., y_N\} \in \mathbb{R}^N \\ \text{s.t.} \quad & \mathbf{h}_{overlap}(\mathbf{x},\mathbf{y}) = 0 \\ & h_{CG}(\mathbf{x},\mathbf{y}) = 0 \end{aligned} \tag{E.1}$$

where r corresponds to the objective function *i.e.*, minimize the container radius, $\mathbf{x}, \mathbf{y}$ are the vectors of continuous coordinates of the circles and $\mathbf{h}_{overlap}(x, y)$,

$h_{CG}(x,y)$ are respectively the overlapping and balancing constraints with CG the center of gravity of the circles. The constraints are mathematically expressed as:

$$\mathbf{h}_{overlap}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \Delta S_{ij}(x_i, x_j, y_i, y_j) \tag{E.2}$$

where ΔS_{ij} is the area of intersection of two circles i and j.

$$h_{CG}(\mathbf{x},\mathbf{y}) = \sqrt{\left(\frac{\sum_{i=1}^{N} m_i x_i}{\sum_{i=1}^{N} m_i}\right)^2 + \left(\frac{\sum_{i=1}^{N} m_i y_i}{\sum_{i=1}^{N} m_i}\right)^2} \tag{E.3}$$

To sum up, the problem is to solve is a $2N$-dimensional continuous, single-objective and constrained optimization problem.

E.2 Swarm Intelligence algorithm for balanced circular bin packing problems

The proposed approach is a component swarm optimization algorithm based on a virtual-force system (CSO-VF), adapted from [Gam+22] for solving the balanced circular bin packing problem. Similar approaches are mainly used in the field of robotics in order, for instance, to control a swarm of robots moving in a given area with one or several objectives (*e.g*, reaching a target destination), as well as hard constraints (*e.g.*, avoid collisions and forbidden zones) [Bra+13; KB86]. This strategy is adapted in this paper in order to optimize the layout of various items in a container and is in line with quasi-physical models developed for instance in [He+13; Liu+16a]. The main focus of the approach is to define dedicated operators for the evolution of a dynamical system of the components (*i.e.*, the circles) based on the fundamental principle of dynamics to satisfy the constraints and minimize the objective function in order to ensure an efficient resolution of the constraints as well as optimization capabilities. In this algorithm, each component is assumed to be a particle in a swarm. At each iteration of the algorithm, depending of the virtual forces that are applied to the particle, each of them evolves in the container in order to minimize the objective function and solve the constraints. It is important to note that this type of algorithm is different from classical particle swarm optimization. Indeed, in the CSO-VF algorithm, each particle of the swarm corresponds to a single circle and so to a part of the entire solution while in classical particle swarm optimization a particle corresponds to an entire solution.

E.2.1 The virtual-force system

In the CSO-VF algorithm, each circle i is described by its dynamic features: its acceleration $\mathbf{a_i}$, its speed $\mathbf{v}_i$ and its position $\mathbf{p}_i = \{x_i, y_i\}$. Forces ($\mathbf{F}_i^k, k \in \{1, ..., N_F\}$, for circle i) with a resultant $\mathbf{F}_i$ are applied to the circle in order to move it at each iteration as illustrated on Figure E.1a. Each of the forces of the virtual-force system aims at solving the constraints as well as optimizing the objective function. Consequently, the forces $\mathbf{F}_i^k$ applied to the circle i are of three types: $\mathbf{F}_{ij}^{overlap}$ to solve the overlapping constraints between the circles i and j, $\mathbf{F}_i^{CG}$ to solve the balancing constraint and $\mathbf{F}_i^{radius}$ to minimize the objective function.

By the end of an iteration, the resulting force $\mathbf{F}_i$ is calculated for each circle thanks to Equation E.4 and the fundamental principle of dynamics is applied in order to update the position of the swarm of circles at step $t+1$ (separated from step t by Δt which corresponds to a time unit) according to Equations E.5, E.6 and E.7 (detailed for circle i).

$$\mathbf{F}_i = \begin{cases} \displaystyle\sum_{k=1}^{N_F} \mathbf{F}_i^k & \text{if } \left\| \displaystyle\sum_{k=1}^{N_F} \mathbf{F}_i^k \right\| < F_{max} \\ \dfrac{\sum_{k=1}^{N_F} \mathbf{F}_i^k}{\left\| \sum_{k=1}^{N_F} \mathbf{F}_i^k \right\|} F_{max} & \text{otherwise.} \end{cases} \tag{E.4}$$

where F_{max} is a hyperparameter corresponding to the maximum value of the norm of the resulting force vector.

$$\mathbf{a}_{i,t+1} = \frac{\mathbf{F}_i}{m_i} \tag{E.5}$$

$$\mathbf{v}_{i,t+1} = \mathbf{v}_{i,t} + \mathbf{a}_{i,t+1}\Delta t \tag{E.6}$$

$$\mathbf{p}_{i,t+1} = \mathbf{p}_{i,t} + \mathbf{v}_{i,t+1}\Delta t \tag{E.7}$$

The three forces of the virtual-force system are detailed and formulated as follows (for a circle i):

- **The overlap constraint forces:** If two circles i and j are overlapping each other, repulsive forces $\mathbf{F}_{ij}^{overlap}$ and $\mathbf{F}_{ji}^{overlap}$ are applied to each of them as illustrated on Figure E.1b. The overlap forces are expressed as:

$$\mathbf{F}_{ij}^{overlap} = \begin{cases} -\dfrac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\| + \epsilon} v_{max} - \mathbf{v}_i & \text{if } \Delta S_{ij}(\mathbf{p}_i, \mathbf{p}_j) \neq 0 \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{E.8}$$

  where v_{max} is a hyperparameter corresponding to the maximum value of the norm of the speed vector, ϵ ensures numerical stability and $\mathbf{0} = (0, 0)$ the null vector.

- **The balance constraint forces:** In order to position the center of mass of the circles on the geometrical center of the container, gradient-based forces are applied along the opposite of the gradient of the position of the global center of mass according to the position of the center of inertia of each circle. This force is named $\mathbf{F}_i^{CG}$. It is illustrated on Figure E.1b with circle k. The balancing forces are expressed as:

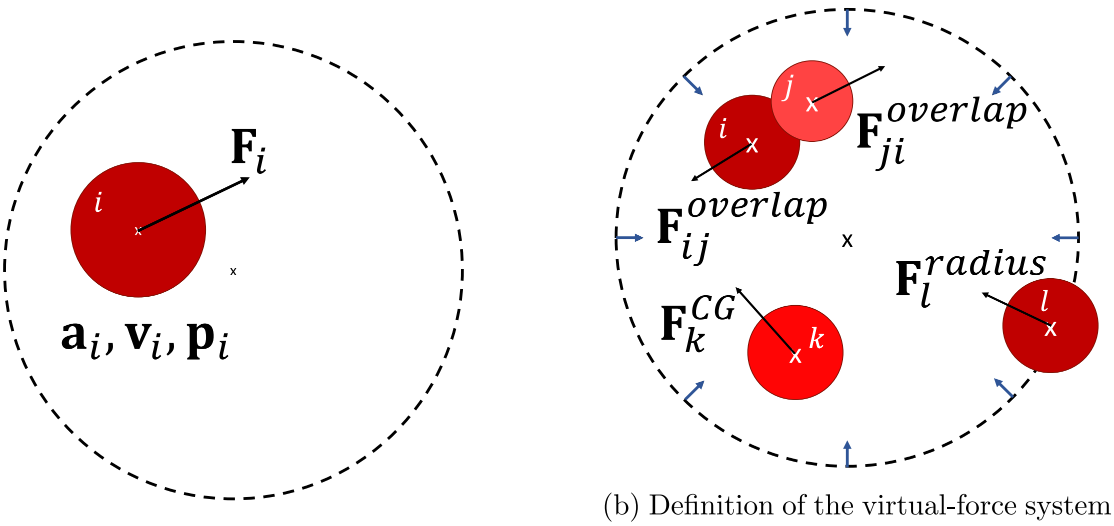
$$\mathbf{F}_i^{CG} = -\alpha \nabla \mathbf{h}_{CG}(\mathbf{p}_i) \tag{E.9}$$

  where α is a "step-size" hyperparameter of the algorithm and $\nabla \mathbf{h}_{CG}(\mathbf{p}_i)$ corresponds to the gradient of the position of the center of gravity of the circles with respect to the position of the considered circle i. This type of force is inspired from gradient-based descent algorithms.

- **The objective function forces:** To minimize the radius of the container, the container is initialized to a given radius which decreases along with the iterations as described in Section E.2.2. The circles must be contained by the container and thus, an attractive force directed toward the geometrical center of the container is applied to each of the circles which does not belong to the container. This force is named $\mathbf{F}_i^{radius}$ and is illustrated on Figure E.1b with circle l. The radius forces are expressed as:

$$\mathbf{F}_i^{radius} = \begin{cases} \frac{\mathbf{p}_c - \mathbf{p}_i}{\|\mathbf{p}_c - \mathbf{p}_i\| + \epsilon} v_{max} - \mathbf{v}_i & \text{if } \Delta S_{i,container}(\mathbf{p}_c, \mathbf{p}_i) < S_i \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{E.10}$$

where $\Delta S_{i,container}$ corresponds to the area of intersection of circle i with the container, S_i to the area of circle i, $\mathbf{p}_c = (0,0)$ corresponds to the position's vector of the geometrical center of the container and ϵ ensures numerical stability.



(a) Definition of a circle.

(b) Definition of the virtual-force system for four circles i, j, k and l.

Figure E.1: Definition of the circles' dynamic features and the associated virtual-force system.

E.2.2 The evolution of the container's radius

In order to minimize the radius of the container and consequently to maximize the occupation rate of the circles in the container, the radius of the container is initialized as described in Section E.2.3 and then decreases along with the iterations. Indeed, at each iteration (*i.e.*, timestep in the evolution of the dynamical system), if a feasible solution has been found (*i.e.*, all the constraints are satisfied), the current radius of the container r_{t-1} is updated to a target container radius r_t^* (where * stands for target), until a new feasible solution is found and the new actual container radius r_t corresponding to the enclosing circle of the items centered on their center of gravity is calculated, and so on. The container is centered on the center of gravity of the circles to ensure that the center of gravity constraint (Equation E.3) is automatically satisfied. The proposed law of evolution of the target container radius r_t^* is expressed as follows:

$$r_t^* = r_{t-1} - s_t \tag{E.11}$$

$$s_t = s_{min} + (s_{max} - s_{min}) \exp\left(t \ln\left(\frac{1}{1 + \frac{c}{N_{it}}}\right)\right) \tag{E.12}$$

where N_{it} is the maximum number of iterations, s_{min} and s_{max} are respectively the minimum and maximum radius steps, and c is an hyperparameter characterizing the speed of evolution of the container radius along with the iterations. Figure E.2 shows the influence of the hyperparameter c on the law of evolution of the step s_t along with the number of iterations.

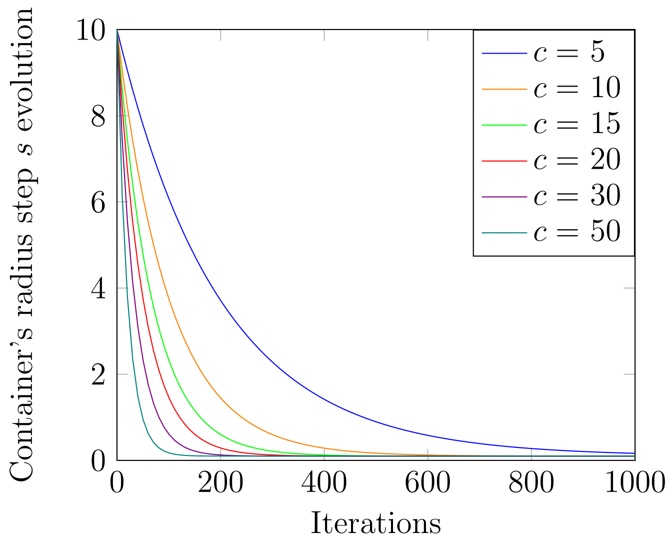


Figure E.2: Influence of the hyperparameter c on the evolution of the container's radius step size along with the iterations for: 1000 iterations, $s_{max} = 10, s_{min} = 0.1$.

Moreover, if a stagnation of the convergence of the container's radius is observed during at least 10% of the total number of iterations (*i.e.*, no feasible solution is found, at least one constraint is not satisfied), s_t is set to s_{min}. Indeed, a stagnation of the convergence of the container's radius might be due to a too large step of the container's radius with respect to the convergence step.

E.2.3 Initialization of the algorithm

E.2.3.1 Initialization of the container

The radius of the container is initialized such that the occupation rate of the circles is equal to 15%. The occupation rate O of a layout is defined as:

$$O = \frac{\sum_{i=1}^{N} S_i}{S_{container}} \tag{E.13}$$

where S_i is the surface of the circle i and $S_{container}$ the surface of the container.

E.2.3.2 Initialization of the positions of the circles

The initialization of the circles is computed in two steps: The circles are grouped according to their masses in order to be positioned separately and each group is then positioned in the container thanks to a Latin Hypercube Sampling. This promotes an homogeneous distribution of the circles in the container according to their masses. Figure E.3 shows the evolution between an initialized layout of the container, an intermediate layout and the corresponding optimal layout obtained with the CSO-VF algorithm for 100 unequal circles. The redder the circles, the heavier their corresponding masses.

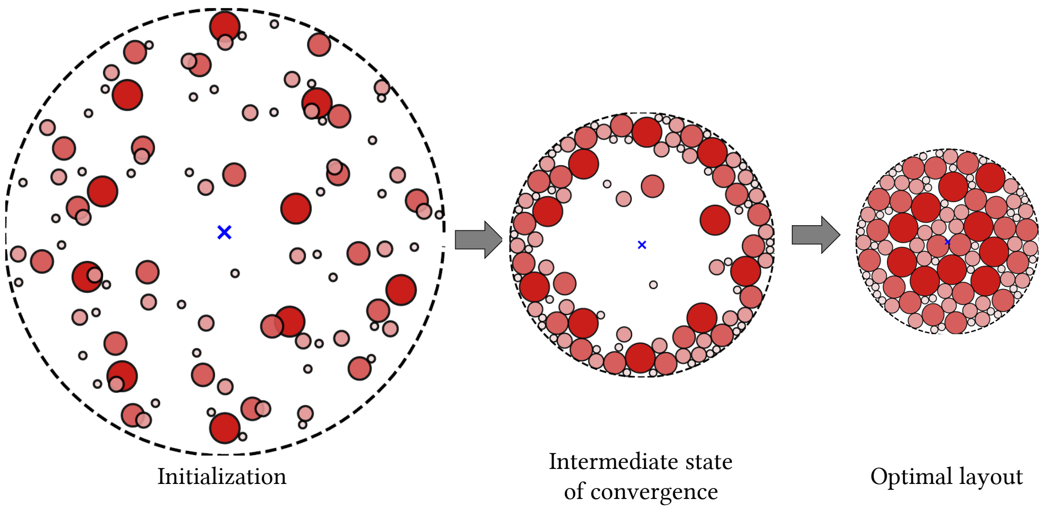


Figure E.3: Evolution between an initialized container, an intermediate layout and the corresponding optimal layout found by the CSO-VF algorithm. The blue cross stands for the center of gravity of the circles. The black dotted circle stands for the container, centered on the center of gravity of the circles. The redder the circles, the heavier their corresponding masses.

E.2.4 The hyperparameters

The 6 hyperparameters of the proposed CSO-VF algorithm are:

- Dynamic features hyperparameters: maximum value of the norm of the force vector F_{max} and speed vector v_{max};

- Balancing force hyperparameter: α;
- Radius law hyperparameters (in Equations E.11, E.12): maximum and minimum radius steps s_{max} and s_{min}, speed evolution hyperparameter c.

Figure E.4 describes the CSO-VF algorithm.

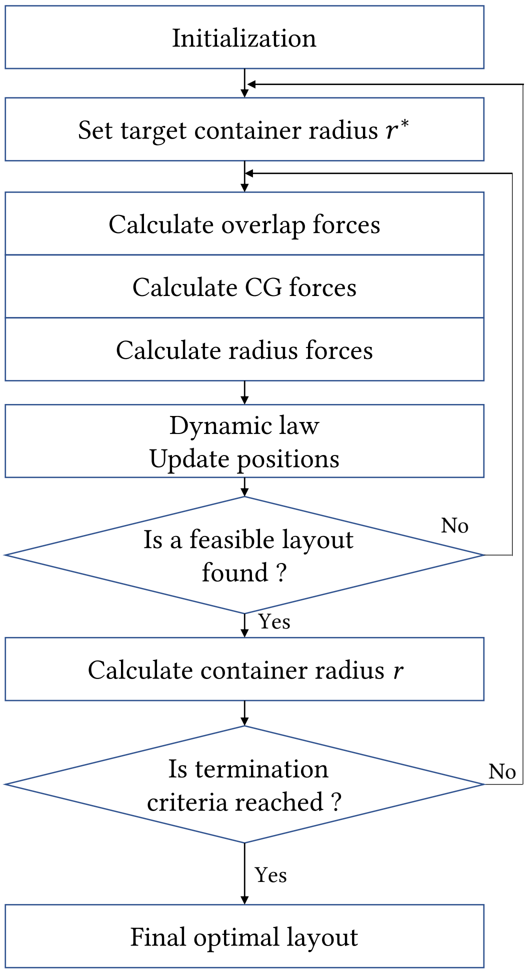


Figure E.4: Steps of the CSO-VF algorithm.

E.3 Experimentations and results

In this section, the CSO-VF algorithm is applied to two benchmarks of balanced circular bin packing problems of growing complexity, with respectively 10 and 3 problems taken from [XXA07a] and [Rom+22]. The best obtained results are shown and analyzed, and a complementary study on the robustness of the CSO-VF algorithm is conducted on the most complex benchmark of problems.

E.3.1 First benchmark of balanced circular bin packing problems

The first benchmark of balanced circular bin packing is initially described in [XXA07a]. This consists of 10 different sets of balanced circular bin packing problems from 10 to 55 circles of various sizes and masses, fully reported in Table E.5 in E.5.1. Table E.1 sums up the maximal and minimal radii and masses of the circles for each problem of the benchmark.

Pb.	Number of circles	Radii	Masses
1	10	8-23	20-93
2	15	8-24	12-98
3	20	8-24	11-94
4	25	6-24	11-96
5	30	6-24	12-96
6	35	7-24	12-99
7	40	6-23	12-99
8	45	6-24	11-99
9	50	5-24	11-99
10	55	6-24	13-99

Table E.1: Minimum and maximum radii and masses of the circles for each problem of the benchmark (Pb.). They are detailed in E.5.1.

The CSO-VF algorithm is applied to the 10 problems of this first benchmark with 100 repetitions each from different initializations using the process described in Section E.2.3. The hyperparameters are set thanks to a parametric study.
The results are outlined as follows:

- The best obtained layouts' radii are reported on Table E.2 and compared with those published in [XXA07a] obtained with the PSO algorithm described in [ZGG05], those obtained in [XXA07a] with a compaction algorithm based on gradient search and SA (Comp.+SA), those obtained in [XXA07a] with a compaction algorithm based on gradient search and PSO (Comp.+PSO) and those from [Liu+15] obtained with a combination of the energy landscape paving method and local search procedure (IELP-LS);

- Figure E.5 characterizes the speed of convergence: it shows the number of iterations necessary to reach 10%, 5%, 1%, 0.5% and 0.1% of the final value for the best run of all the 10 problems;

- Figure E.6 shows the best obtained layout for each of the 10 problems. The redder the circles, the heavier they are;

- In Section E.5.3, Table E.6 reports the CPU time necessary to run 20000 iterations with the allocated numerical resources for each benchmark problem.

Pb.	PSO [ZGG05]	Comp. + SA [XXA07a]	Comp. + PSO [XXA07a]	IELP-LS [Liu+15]	Proposed CSO-VF algorithm	Relative Improvement (%) wrt best method
1	61.32	60.96	59.93	59.92	**59.85**	0.12%
2	76.58	68.77	67.65	67.39	**67.07**	0.47%
3	89.15	83.09	83.06	82.99	**82.58**	0.49%
4	106.31	83.97	84.24	82.98	**82.84**	0.17%
5	136.88	99.58	99.89	98.97	**98.77**	0.20%
6	148.39	102.86	102.71	102.32	**101.52**	0.78%
7	165.79	115.15	115.58	115.00	**113.53**	1.28%
8	172.69	120.63	119.67	119.07	**117.99**	0.90%
9	189.89	125.82	126.19	124.98	**124.30**	0.54%
10	200.82	138.22	138.89	136.13	**135.99**	0.10%

Table E.2: Results for the first benchmark of 10 problems (Pb.): best container's radius obtained in [ZGG05; XXA07a; Liu+15] and with the CSO-VF algorithm. Best results are indicated in bold.

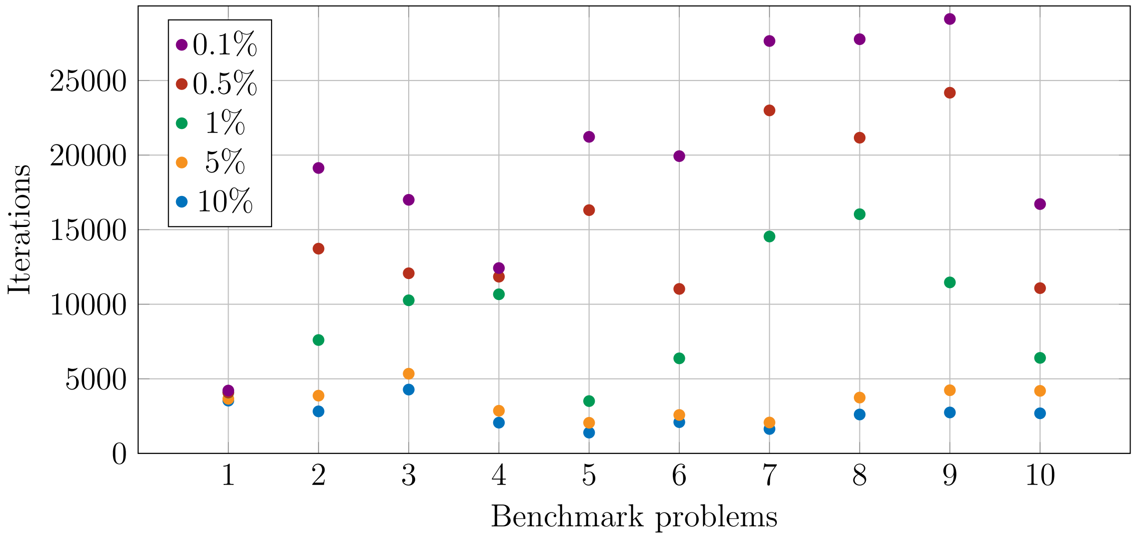


Figure E.5: Number of iterations to reach 10%, 5%, 1%, 0.5% and 0.1% of the final obtained values for the best runs of the 10 benchmark problems.

The proposed strategy allows to solve all the benchmark problems. It enables to improve the final best radius of all the problems, from 0.1% to 1.28% in comparison to the results from the literature obtained with various techniques [XXA07a; Liu+15; ZGG05]. This relative improvement tends to increase with the problem number (*i.e.*, with the number of circles) except for the last one. This might be explained by the fact that the CSO-VF algorithm is provided with dedicated forces to solve each of the constraints and thus, remains consistent in its ability to resolve constraints from 10 to 55 circles.

Figure E.5 shows that for all 10 problems less than 5000 iterations are necessary to reach 5% of the final best value. This represents from 43 seconds (problem 1, 10 circles) to 3.5 minutes (problem 10, 55 circles) thanks to Table E.6. Figure E.5 also highlights that the speed of convergence until 5% of the final value seems not linked to the number of circles of the benchmark problems. However, problems 7, 8 and 9 (*i.e.*, 40, 45 and 50 circles) seem to require more iterations to converge to 0.1% of their final value.

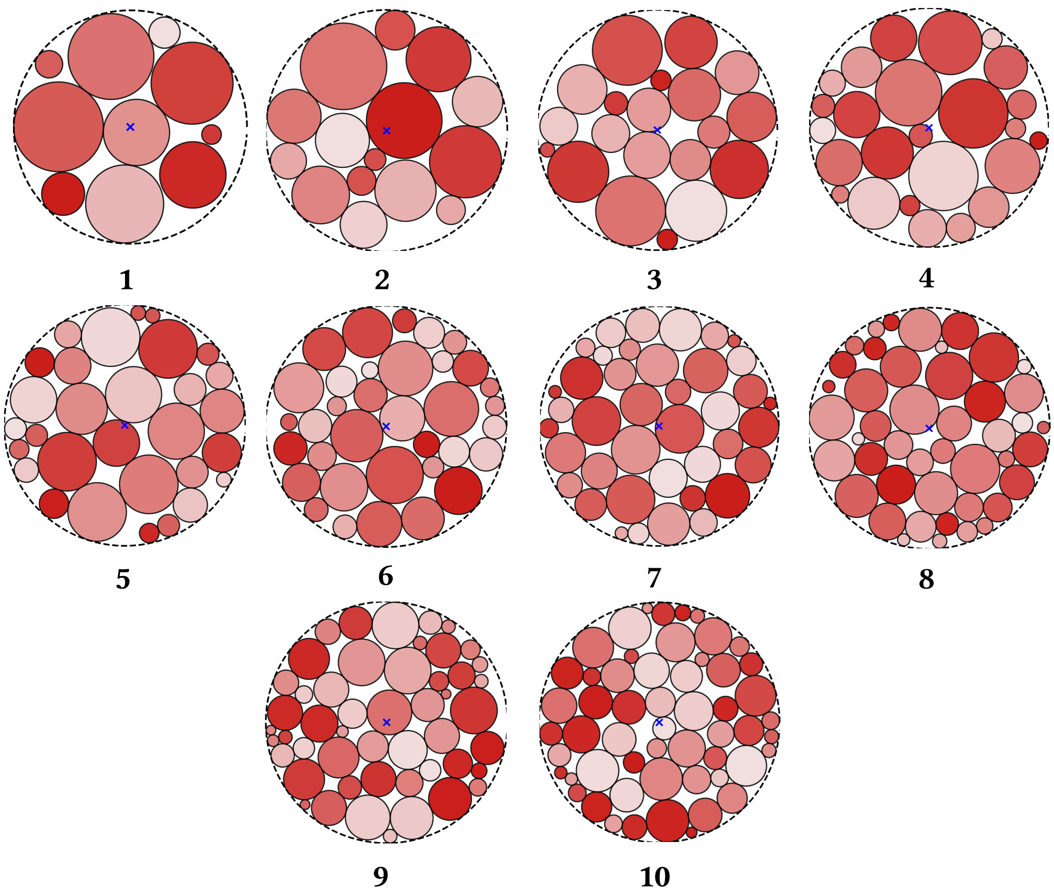


Figure E.6: Obtained final best layout for the first benchmark of problems with the CSO-VF algorithm.

E.3.2 Second benchmark of balanced circular bin packing problems

The second benchmark of problems is initially proposed in [Rom+22]. This consists of 3 problems with 100, 150 and 300 circles, in which the radius of each circles is equal to its mass. Table E.3 lists the number of circles and dimensions for each problem of the benchmark.

The hyperparameters are set using a parametric analysis and the CSO-VF algorithm is applied to the 3 problems of this second benchmark with 100 repetitions

Pb.	1		2		3	
	Number of circles	Radii, masses	Number of circles	Radii, masses	Number of circles	Radii, masses
Circles	40	10	50	10	100	10
	30	20	40	20	80	20
	20	30	30	30	60	30
	10	40	20	40	40	40
			10	50	20	50
Total number of circles	100		150		300	

Table E.3: Configuration of the 3 benchmark problems (Pb.): Number of circles, dimensions (radii equal to masses) and total number of circles for each problem.

each from different initializations using the process described in Section E.2.3. The results are outlined as follows:

- The best obtained layouts are reported on Table E.4 and compared with those obtained in [Rom+22] with a sequential algorithm based on a variant of the Shor's r-algorithm [Sho+03; Ste17];
- Figure E.7 characterizes the speed of convergence: it shows the number of iterations necessary to reach 10%, 5%, 1%, 0.5% and 0.1% of the final value for the best run of all the benchmark problems;
- Figure E.8 characterizes the robustness of the algorithm on each problem: it shows the numbers of repetitions that reach 10%, 5%, 1% and 0.5% of the best layout for all the benchmark problems;
- Figure E.9 shows the best obtained layout for each problem of this benchmark. The redder the circles, the heavier their corresponding masses;
- In Section E.5.4, Table E.7 reports the CPU time necessary to run 15000 iterations with the allocated numerical resources for all the benchmark problems.

The results are reported in Table E.4 and compared with the results obtained in [Rom+22]. Figure E.9 shows the best obtained layout for each problem of the benchmark.

Pb.	Results from [Rom+22]	Proposed CSO-VF algorithm	Relative Improvement (%)
1	257.19	**247.93**	3.60%
2	368.40	**357.97**	2.83%
3	520.56	**504.11**	3.16%

Table E.4: Results for the second benchmark of problems (Pb.): best container's radius obtained in [Rom+22] and with the CSO-VF algorithm.

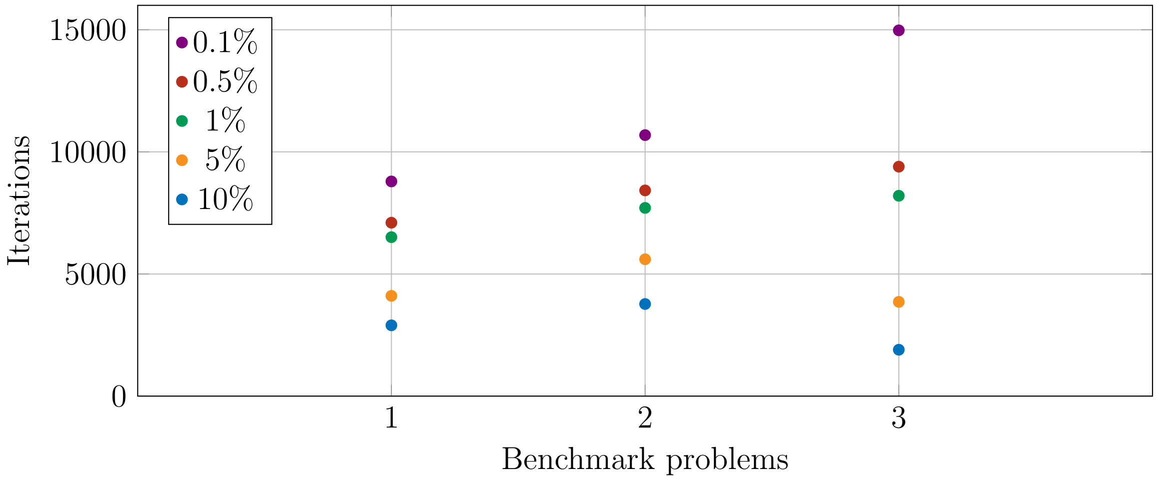


Figure E.7: Number of iterations to reach 10%, 5%, 1%, 0.5% and 0.1% of the obtained value for the best run of the 3 problems of the benchmark.

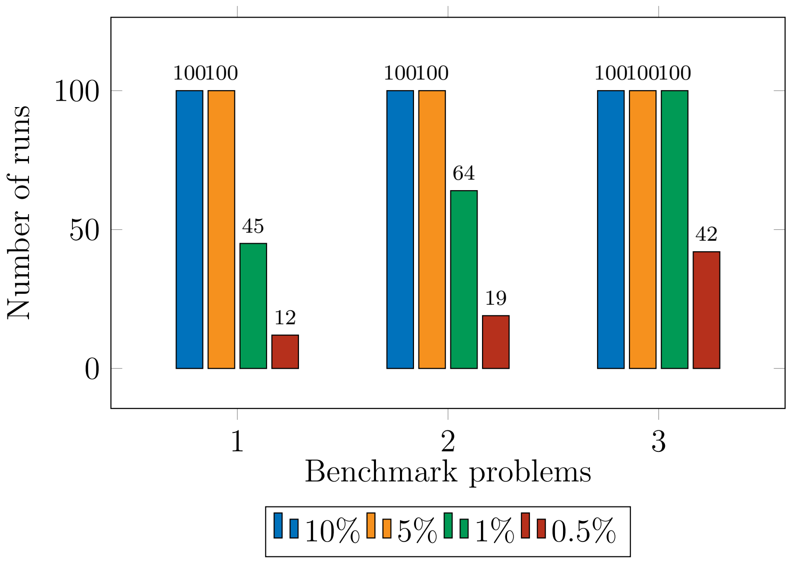


Figure E.8: Number of runs (amongst 100) to reach 10%, 5%, 1% and 0.5% of the obtained value for the best run of the 3 problem of the benchmark.

Table E.9 and Figure E.9 show that the algorithm succeeds in finding feasible layouts for balanced circular bin packing problems up to 300 circles. It provides better container's radii from 2.83% to 3.60% compared to the best known layouts proposed in [Rom+22]. Figure E.7 shows that for all 3 problems of the benchmark, around 5000 iterations are necessary to reach 5% of the final value. Moreover, Figure E.7 shows that the number of iterations needed to reach less than 1% of the final best value increases along with the total number of circles in each benchmark problem. However, for all three problems, 5% of their best final value is reached in about 5000 iterations. Figure E.8 characterizes the robustness of the CSO-VF algorithm. It shows that for the 3 bin packing problems of the benchmark, all the runs manage to converge to at least 5% of the final best layouts. Furthermore, the robustness of the algorithm increases along with the total number of circles of each problem. Indeed, for the 300-circles benchmark problem (*i.e.*, the third one), the CSO-VF algorithm achieves 100% of repetitions reaching 1% of the best radius obtained in Table E.4, and more than 40% of repetitions reaching 0.5% of the best radius obtained in Table E.4.

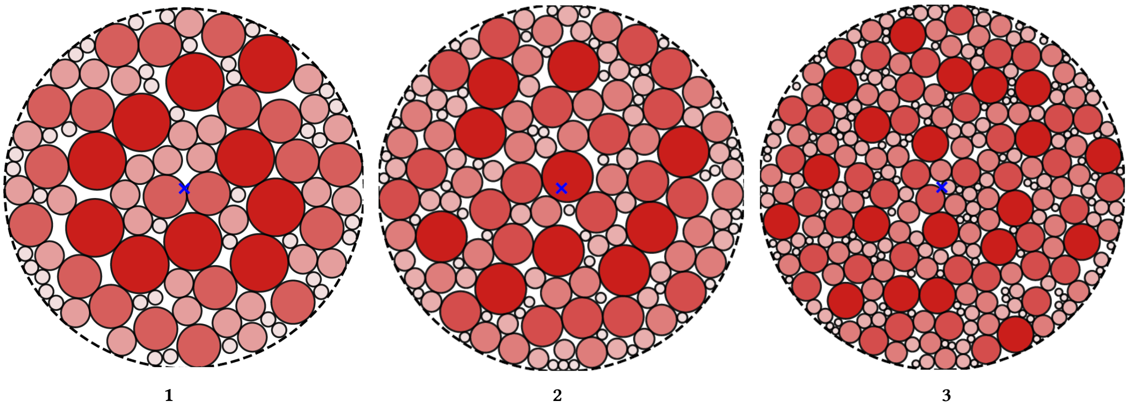


Figure E.9: Obtained final best layout for the second benchmark of problems with the CSO-VF algorithm.

E.3.3 Summary of the results for the two benchmarks

The CSO-VF algorithm provides better optimal layouts for all 13 problems where 10 to 300 circles had to be positioned in comparison with several techniques from the literature. It has been shown that the CSO-VF algorithm solves all the benchmarks problems with similar convergence speed. As the number of circles increases from 100 to 300 circles, the algorithm appears to be more robust and able to achieve up to 0.5% of the best final value. It might be due to the fact that problems with fewer circles are more sensitive to the initialization than problems with several hundreds of circles and especially if they are uniform in terms of dimensions and masses. The virtual-force system of the CSO-VF algorithm uses dedicated operators to solve each of the constraints as well as minimize the objective function. Therefore, the CSO-VF algorithm provides constant and efficient optimization capabilities regardless of the considered number of circles.

E.4 Conclusions and future works

This paper describes a quasi-physical approach in order to solve balanced circular bin packing problems which aim is to pack weighted circles into as smallest balanced container as possible. The proposed algorithm consists in a swarm intelligence based on a virtual-force system which provides dedicated operators to solve each of the constraints as well as optimize the objective function. It has been successfully applied on various benchmark problems from 10 to 300 circles. The reported results show that this strategy allows to improve the results obtained for the considered benchmark problems compared to other various counterparts from the literature. In the future, the following perspectives will be investigated. Firstly, the evolution of the container's radius step law will be improved to increase the robustness of the method. Secondly, the algorithm will be extended in order to tackle 3-dimensional bin packing problems.

E.5 Further materials

E.5.1 First benchmark problems configurations

Table E.5 details the dimensions of the circles of the first benchmark of problems addressed in Section E.3.1.

Inst	Size	Radii	Masses
I1	10	{20, 22, 17, 17, 7, 21, 11, 5, 23, 8}	{35, 61, 49, 89, 68, 80, 93, 82, 70, 20}
I2	15	{8, 14, 8, 15, 11, 17, 21, 16, 6, 18, 24, 13, 20, 10, 15}	{75, 29, 36, 58, 75, 32, 98, 52, 76, 85, 59, 18, 85, 36, 12}
I3	20	{20, 24, 8, 11, 13, 7, 7, 15, 24, 18, 15, 17, 17, 14, 16, 18, 5, 21, 21, 13}	{86, 72, 81, 54, 29, 94, 92, 41, 57, 77, 40, 67, 31, 47, 39, 61, 73, 83, 11, 20}
I4	25	{24, 16, 19, 7, 14, 24, 15, 6, 16, 16, 23, 10, 9, 10, 18, 22, 7, 9, 7, 13, 14, 8, 18, 6, 8}	{16, 80, 52, 21, 42, 86, 67, 96, 61, 79, 57, 62, 32, 38, 20, 75, 80, 11, 53, 32, 41, 68, 85, 53, 71}
I5	30	{14, 15, 11, 19, 9, 6, 23, 9, 23, 13, 24, 12, 24, 24, 10, 8, 9, 8, 6, 11, 6, 16, 24, 12, 9, 19, 13, 24, 21, 18}	{24, 52, 37, 17, 12, 19, 51, 67, 23, 46, 14, 96, 55, 84, 21, 92, 69, 65, 72, 36, 73, 83, 83, 97, 73, 81, 30, 46, 49, 51}
I6	35	{10, 20, 13, 19, 19, 10, 14, 14, 24, 11, 20, 15, 7, 18, 22, 10, 13, 12, 21, 14, 9, 10, 9, 7, 8, 18, 8, 8, 23, 14, 13, 21, 23, 16, 10}	{44, 46, 14, 32, 70, 31, 95, 24, 75, 99, 99, 79, 10, 79, 69, 64, 12, 47, 41, 62, 17, 85, 43, 70, 43, 63, 44, 57, 62, 20, 17, 80, 47, 68, 19}
I7	40	{6, 12, 20, 6, 14, 19, 9, 20, 10, 13, 12, 14, 23, 17, 16, 19, 15, 10, 12, 18, 21, 6, 20, 17, 13, 20, 17, 6, 21, 15, 12, 9, 14, 20, 23, 16, 23, 9, 23, 18}	{74, 48, 16, 35, 19, 58, 87, 90, 17, 29, 32, 63, 46, 76, 26, 88, 71, 49, 89, 14, 68, 94, 41, 53, 36, 67, 14, 88, 99, 46, 66, 14, 21, 44, 73, 72, 72, 37, 82, 12}
I8	45	{13, 8, 11, 21, 9, 20, 24, 20, 17, 21, 7, 13, 24, 7, 6, 8, 18, 15, 12, 18, 17, 21, 8, 23, 22, 15, 10, 17, 24, 8, 14, 6, 16, 14, 6, 10, 19, 21, 20, 6, 16, 14, 6, 19, 11}	{91, 95, 96, 47, 63, 37, 56, 96, 84, 70, 36, 41, 48, 12, 86, 43, 70, 71, 56, 89, 52, 49, 53, 82, 42, 35, 11, 82, 88, 58, 74, 16, 91, 57, 26, 39, 48, 68, 72, 69, 27, 44, 25, 99, 96}
I9	50	{9, 17, 5, 15, 24, 23, 12, 9, 5, 13, 7, 18, 19, 21, 7, 18, 18, 24, 12, 23, 22, 13, 5, 6, 17, 21, 7, 18, 14, 17, 10, 15, 18, 8, 8, 16, 7, 18, 24, 6, 20, 10, 21, 11, 22, 24, 12, 7, 14, 11}	{19, 85, 60, 19, 88, 18, 28, 55, 66, 47, 49, 69, 93, 94, 35, 43, 93, 34, 27, 61, 20, 52, 51, 41, 98, 85, 82, 89, 54, 43, 54, 94, 80, 99, 41, 41, 63, 28, 19, 53, 11, 78, 65, 10, 98, 43, 78, 24, 84, 16}
I10	55	{17, 23, 17, 13, 18, 21, 23, 22, 7, 9, 8, 13, 20, 11, 10, 19, 10, 14, 12, 22, 19, 10, 17, 11, 21, 8, 15, 16, 19, 21, 17, 19, 8, 6, 13, 13, 14, 19, 18, 23, 20, 24, 24, 13, 13, 19, 7, 6, 10, 8, 8, 10, 24, 19, 24}	{97, 62, 28, 36, 97, 58, 13, 21, 40, 97, 79, 90, 62, 47, 64, 23, 23, 95, 99, 44, 71, 79, 52, 59, 47, 60, 41, 47, 90, 95, 81,98, 70, 47, 90, 13, 93, 50, 21, 80, 17, 52, 96, 73, 88, 16, 91, 97, 40, 52, 50, 90, 19, 69, 14}

Table E.5: Geometrical configuration of each problem: total number of circles (size), their radii and masses.

E.5.2 CPU time of resolution

The routines are implemented in Python language, and executed on a PC with an Intel Core i7, 16 GB of RAM and Windows operating system. The CPU time of resolution for each benchmark problem is reported and analyzed in the following sections.

E.5.3 First benchmark of problems

Pb.	1	2	3	4	5	6	7	8	9	10
Time (min)	1.75	3.62	4.05	5.37	6.12	8.43	10.6	12.35	13.14	13.89

Table E.6: Mean of CPU time for 20000 iterations for the ten benchmark problems (Pb.). Implementation: Python language, Execution: PC with an Intel Core i7, 16 GB of RAM, Operating system: Windows.

E.5.4 Second benchmark problems

Benchmark Problems	1	2	3
CPU time (min)	20.9	42.1	98.7

Table E.7: Mean of CPU time for 15000 iterations for the three problems. Implementation: Python language, Execution: PC with an Intel Core i7, 16 GB of RAM, Operating system: Windows.

E.5.5 Analysis

Table E.6 and E.7 show that the CPU time needed to solve the problems of each benchmark evolves linearly with respect to the number of circles to pack. Indeed, the more circles to position, the more forces to calculate within the virtual-force system of the CSO-VF algorithm. The required CPU time with the allocated numerical resources remains lower than 13.89 minutes for the first benchmark of problems (obtained for 55 circles) and lower than 98.7 minutes for the second benchmark of problems (obtained for 300 circles). It must be noted that the CPU time strongly depends on the implementation language used as well as the numerical resources allocated and is therefore hardly comparable with other CPU time from the literature. Furthermore, the CSO-VF algorithm is compatible with GPU programming that would considerably improve the computing time. However, from an industrial point of view, the effective computing times remain reasonable.

Bibliography

[Abd13] O. Abdelkhalik. "Hidden genes genetic optimization for variable-size design space problems". In: *Journal of Optimization Theory and Applications* 156 (2013), pp. 450–468.

[AG12] O. Abdelkhalik and A. Gad. "Dynamic-size multiple populations genetic algorithm for multigravity-assist trajectory optimization". In: *Journal of Guidance, Control, and Dynamics* 35(2) (2012), pp. 520–529.

[AZT20] F. Abdessamia, W.Z. Zhang, and Y.C. Tian. "Energy-efficiency virtual machine placement based on binary gravitational search algorithm". In: *Cluster Computing* 23 (2020), pp. 1577–1588.

[AAD07] M.A. Abramson, C. Audet, and J.E Dennis. "Filter pattern search algorithms for mixed variable constrained optimization problems". In: *Pacific Journal of Optimization* 3(3) (2007), pp. 477–500.

[Abr+09] M.A. Abramson et al. "Mesh adaptive direct search algorithms for mixed variable optimization". In: *Optimization Letters* 3 (2009), pp. 35–47.

[Agu+04] A.H. Aguirre et al. "Handling constraints using multiobjective optimization concepts". In: *International Journal for Numerical Methods in Engineering* 59(15) (2004), pp. 1989–2017.

[AJ16] A. Ahmadi and M.R.A Jokar. "An efficient multiple-stage mathematical programming method for advanced single and multi-floor facility layout problems". In: *Applied Mathematical Modelling* 40(9-10) (2016), pp. 5605–5620.

[APJ17] A. Ahmadi, M.S. Pishvaee, and M.R.A. Jokar. "A survey on multi-floor facility layout problems". In: *Computers & Industrial Engineering* 107 (2017), pp. 158–170.

[ASE12] G. Aiello, G. La Scalia, and M. Enea. "A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding". In: *Expert Systems with Applications* 39(12) (2012), pp. 10352–10358.

[ASN14] M. Alamuri, B.R. Surampudi, and A. Negi. "A survey of distance/similarity measures for categorical data". In: *In 2014 International joint conference on neural networks (IJCNN) IEEE* (2014), pp. 1907–1914.

[AL21] H. Alibrahim and S.A. Ludwig. “Hyperparameter optimization for machine learning models based on Bayesian optimization”. In: *In 2021 IEEE Congress on Evolutionary Computation (CEC)* (2021), pp. 1551–1559.

[Alo+19] M.T. Alonso et al. “Mathematical models for multi container loading problems with practical constraints”. In: *Computers & Industrial Engineering* 127 (2019), pp. 722–733.

[ARL12] M.A. Alvarez, L. Rosasco, and N.D. Lawrence. “Kernels for vector-valued functions: A review”. In: *Foundations and Trends in Machine Learning* 4(3) (2012), pp. 195–266.

[APT08] R. Alvarez-Valdés, F. Pareño, and J.M. Tamarit. “Reactive GRASP for the strip-packing problem”. In: *Computers & Operations Research* 35(4) (2008), pp. 1065–1083.

[AKB13] J.S. Angelo, E. Krempser, and H.J.C Barbosa. “Differential evolution for bilevel programming”. In: *In Proceedings of the Congress on Evolutionary Computation (CEC), Cancún, Mexico, 2013* (2013), pp. 470–477.

[AB63] G.C. Armour and E.S. Buffa. “A heuristic algorithm and simulation approach to relative location of facilities”. In: *Management science* 9(2) (1963), pp. 294–309.

[Arn11] D.V. Arnold. “Analysis of a repair mechanism for the $(1, \lambda)$-ES applied to a simple constrained problem”. In: *In Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), pp. 853–860.

[AH12] D.V. Arnold and N. Hansen. “A (1+1)-CMA-ES for constrained optimisation”. In: *In Proceedings of the 14th annual conference on Genetic and evolutionary computation* (2012), pp. 297–304.

[Aro50] N. Aronszajn. “Theory of reproducing kernels”. In: *Transactions of the American Mathematical Society* 68(3) (1950), pp. 337–404.

[AWT16] A. Derakhshan Asl, K.Y. Wong, and M.K. Tiwari. “Unequal-area stochastic facility layout problems: solutions using improved covariance matrix adaptation evolution strategy, particle swarm optimisation, and genetic algorithm”. In: *International Journal of Production Research* 54(3) (2016), pp. 799–823.

[AD01] C. Audet and J.E. Dennis. “Pattern search algorithms for mixed variable programming”. In: *SIAM Journal on Optimization* 7(1) (2001), pp. 573–594.

[AD06] C. Audet and J.E. Dennis. “Mesh Adaptive Direct Search Algorithms for Constrained Optimization”. In: *SIAM Journal on Optimization* 17(1) (2006), pp. 188–217.

[AD09] C. Audet and J.E. Dennis. “A Progressive Barrier for Derivative-Free Nonlinear Programming”. In: *SIAM Journal on Optimization* 20(1) (2009), pp. 445–472.

[ADT19] C. Audet, S. Le Digabel, and C. Tribes. “The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables”. In: *SIAM Journal on Optimization* 29(2) (2019), pp. 1164–1189.

[AHD23] C. Audet, E. Hallé-Hannan, and S. Le Digabel. “A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables”. In: *In Operations Research Forum. Cham: Springer International Publishing.* 4(1) (2023), p. 12.

[Aud+00] C. Audet et al. “A surrogate-model-based method for constrained optimization”. In: *8th Symposium on Multidisciplinary Analysis and Optimization* (2000), p. 4891.

[BAS12] M. Baghel, S. Agrawal, and S. Silakari. “Survey of metaheuristic algorithms for combinatorial optimization”. In: *International Journal of Computer Applications* 58(19) (2012).

[BMN02] A.P. Barbosa-Póvoa, R. Mateus, and A.Q. Novais. “Optimal 3D layout of industrial facilities”. In: *International Journal of Production Research* 40(7) (2002), pp. 1669–1698.

[Bar13] J.F. Bard. “Practical bilevel optimization: algorithms and applications”. In: *Springer Science & Business Media* 30 (2013).

[BZ17] T. Bartz-Beielstein and M. Zaefferer. “Model-based methods for continuous and discrete global optimization”. In: *Applied Soft Computing* 55 (2017), pp. 154–167.

[BG01] A. Baykasoğlu and N.N. Gindy. “A simulated annealing algorithm for dynamic layout problem”. In: *Computers & Operations Research* 28(14) (2001), pp. 1403–1426.

[El-04] M.A. El-Baz. “A genetic algorithm for facility layout problems of different manufacturing environments”. In: *Computers & Industrial Engineering* 47(2-3) (2004), pp. 233–246.

[Ben+11] J. Benabes et al. “An interactive-based approach to the layout design optimization”. In: *In Global Product Development: Proceedings of the 20th CIRP Design Conference, Ecole Centrale de Nantes, Nantes, France, 19th-21st April 2010. Springer Berlin Heidelberg.* (2011), pp. 511–520.

[Ber+11] J. Bergstra et al. “Algorithms for hyper-parameter optimization”. In: *Advances in neural information processing systems* 24 (2011).

[BA12] S. Bernardi and M.F. Anjos. “A two-stage mathematical-programming method for the multi-floor facility layout problem”. In: *Journal of the Operational Research Society* 64(3) (2012), pp. 352–364.

[BT04] J. Berntsson and M. Tang. “A slicing structure representation for the multi-layer floorplan layout problem”. In: *In Applications of evolutionary computing* (2004), pp. 188–197.

[BT97] D. Bertsimas and J.N. Tsitsiklis. “Introduction to linear optimization”. In: *Belmont, MA: Athena scientific* 6 (1997), pp. 479–530.

[BM85] E.L. Blair and S. Miller. “An Interactive Approach to Facilities Design Using Microcomputers”. In: *Computers and Industrial Engineering* 9(1) (1985), pp. 91–102.

[Bli00] T. Blickle. “Tournament selection”. In: *Evolutionary Computation* 1 (2000), pp. 181–186.

[BBL14] C. Bliek1ú, P. Bonami, and A. Lodi. “Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report”. In: *In Proceedings of the twenty-sixth RAMP symposium* (2014), pp. 16–17.

[Blo78] T.E. Block. “FATE: A New Construction Algorithm for Facilities Layout”. In: *Journal of Engineering Production* 2(2) (1978), pp. 111–120.

[Bor06] A. Bortfeldt. “A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces”. In: *European Journal of Operational Research* 172(3) (2006), pp. 814–837.

[BME94] Y.A. Bozer, R.D. Meller, and S.J. Erlebacher. “An improvement-type layout algorithm for single and multiple-floor facilities”. In: *Management Science* 40(7) (1994), pp. 918–932.

[BAZ15] N. Bozorgi, M. Abedzadeh, and M. Zeinali. “Tabu search heuristic for efficiency of dynamic facility layout problem”. In: *The International Journal of Advanced Manufacturing Technology* 77 (2015), pp. 689–703.

[Bra+13] M. Brambilla et al. “Swarm robotics: a review from the swarm engineering perspective”. In: *Swarm Intelligence* 7(1) (2013), pp. 1–41.

[Bre01] L. Breiman. “Random forests”. In: *Machine Learning* 45(1) (2001), pp. 5–32.

[BM05] A.H. Brie and P. Morignot. “Genetic Planning Using Variable Length Chromosomes”. In: *In ICAPS* (2005), pp. 320–329.

[BCf10] E. Brochu, V.M. Cora, and N. De freitas. “A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning”. In: *arXiv preprint arXiv:1807.02811* (2010).

[BHS20] F. Budianto, J. Halim, and A.C. Sembiring. “Redesigning Furniture Production Floors Using Systematic Layout Planning and ALDEP Method to Minimize Material Handling Costs”. In: *In 2020 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT).IEEE.* (2020), pp. 84–90.

[Bue13] P. Von Buelow. “Improving Generative Design through Selective Breeding”. In: *In International Association of Shell and Spatial Structures (IASS) Symposium 'Beyond the Limits of Man'* (2013).

[BWH21] P. Burggraäf, J. Wagner, and B. Heinbach. “Bibliometric study on the use of machine learning as resolution technique for facility layout problems”. In: *IEEE Access* 9 (2021), pp. 22569–22586.

[BR84] R.E. Burkard and F. Rendl. “A thermodynamically motivated simulation procedure for combinatorial optimization problems”. In: *European Journal of Operational Research* 17(2) (1984), pp. 169–174.

[Bus+21] J.H. Bussemaker et al. “Effectiveness of surrogate-based optimization algorithms for system architecture optimization”. In: *In AIAA AVIATION 2021 FORUM* (2021).

[BM04] M.R. Bussieck and A. Meeraus. “General algebraic modeling system (GAMS)”. In: *Modeling languages in mathematical optimization* (2004), pp. 137–157.

[CSY02] J. Cagan, K. Shimada, and S. Yin. “A survey of computational approaches to three-dimensional layout problems”. In: *Computer-Aided Design* 34(8) (2002), pp. 597–611.

[CE91] K.E. Cambron and G.W. Evans. “Layout design using the analytic hierarchy process”. In: *Computers & Industrial Engineering* 20(2) (1991), pp. 211–229.

[CCC14] F. Carrabs, C. Cerrone, and R. Cerulli. “A tabu search approach for the circle packing problem”. In: *In 2014 17th International conference on network-based information systems. IEEE.* (2014), pp. 165–171.

[CP13] D. Cazzaro and D. Psisinger. “Variable neighborhood search for large offshore wind farm layout optimization”. In: *Computers & Operations Research* 8 (2013), pp. 1–12.

[CD14] T. Chai and R.R. Draxler. “Root mean square error (RMSE) or mean absolute error (MAE)”. In: *Geoscientific model development discussions* 7(1) (2014), pp. 1525–1534.

[CK20] K. Chaiyota and T. Krityakierne. “A comparative study of infill sampling criteria for computationally expensive constrained optimization problems”. In: *Symmetry* 12(10) (2020), p. 1631.

[CL06] C.H. Chang and H.J. Lin. “Multiple-floor facility layout design with aisle construction”. In: *IEMS* 5(1) (2006), pp. 1–10.

[CLS95] C.S. Chen, S.M. Lee, and Q.S. Shen. “An analytical model for the container loading problem”. In: *European Journal of Operational Research* 80(1) (1995), pp. 68–76.

[Che+18] X. Chen et al. “The hybrid algorithms based on differential evolution for satellite layout optimization design”. In: *In 2018 IEEE Congress on Evolutionary Computation (CEC).IEEE* (2018), pp. 1–8.

[CY21] R. Cheng and J. Yan. “On joint learning for solving placement and routing in chip design”. In: *arXiv preprint arXiv:2111.00234* (2021).

[CP06] D.M. Cherba and W. Punch. “Crossover gene selection by spatial location”. In: *In GECCO’06 (ACM, Seattle, 2006)* (2006), pp. 1111–1116.

[CGM21] X. Chou, L.M. Gambardella, and R. Montemanni. “A tabu search algorithm for the probabilistic orienteering problem”. In: *Computers & Operations Research* 126 (2021), p. 105107.

[Cin+08] G.F. Cintra et al. “Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation”. In: *European journal of operational research* 191(1) (2008), pp. 61–85.

[CC99] C. A. Coello Coello and A. D. Christiansen. “MOSES: A multiobjective optimization tool for engineering design”. In: *Engineering Optimization* 31(3) (1999), pp. 337–368.

[CM02] C.A.C. Coello and E.M. Montes. “Constraint-handling in genetic algorithms through the use of dominance-based tournament selection”. In: *Advanced Engineering Informatics* 16(3) (2002), pp. 193–203.

[CK04] P. Corry and E. Kozan. “Ant colony optimisation for machine layout problems”. In: *Computational Optimization and Applications* 28 (2004), pp. 287–310.

[ChL13] A. Costa, P/ hansen, and L. Liberti. “On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square”. In: *Discrete Applied Mathematics* 161(1-2) (2013), pp. 96–106.

[CW78] P. Craven and G. Wahba. “Smoothing noisy data with spline functions”. In: *Numerische mathematik* 31(4) (1978), pp. 377–403.

[CML19] L. Cruz-Piris, I. Marsa-Maestre, and M.A. Lopez-Carmona. “A variable-length chromosome genetic algorithm to solve a road traffic coordination multipath problem”. In: *IEEE Access* 7 (2019), pp. 111968–111981.

[CXW18] F.Z. Cui, Z.Z Xu, and X.K. Wang. “A dual-system cooperative co-evolutionary algorithm for satellite equipment layout optimization”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 232 (2018), pp. 2432–2457.

[CZa19] Feng-Zhe Cui, Chong-Quan Zhong, and Xiu-Kun Wang et al. “A collaborative design method for satellite module component assignment and layout optimization”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233(15) (2019), pp. 5471–5491.

[CH06] Y. Cui and L. Huang. “Dynamic programming algorithms for generating optimal strip layouts”. In: *Computational Optimization and Applications* 33 (2006), pp. 287–301.

[DCI13] G. Damblin, M. Couplet, and B. Iooss. “Numerical studies of space-filling designs: optimization of Latin Hypercube Samples and subprojection properties”. In: *Journal of Simulation* 7(4) (2013), pp. 276–289.

[Dan+16] D.C. Dang et al. "Escaping local optima with diversity mechanisms and crossover". In: *In Proceedings of the Genetic and Evolutionary Computation Conference 2016* (2016), pp. 645–652.

[DA18] S. Darani and O. Abdelkhalik. "Convergence analysis of hidden genes genetic algorithms in space trajectory optimization". In: *Journal of Aerospace Information Systems* 15(4) (2018), pp. 228–238.

[Deb00] K. Deb. "An efficient constraint handling method for genetic algorithms". In: *Computer Methods in Applied Mechanics and Engineering* 186(2-4) (2000), pp. 311–338.

[Deb01] K. Deb. "Nonlinear goal programming using multi-objective genetic algorithms". In: *Journal of the Operational Research Society* 52 (2001), pp. 291–302.

[DA95] K. Deb and R.B. Agrawal. "Simulated binary crossover for continuous search space". In: *Complex Systems* 9 (1995), pp. 115–148.

[DJ02] K. Deb and D. Joshi. "A computationally efficient evolutionary algorithm for real parameter optimization". In: *Technical Report 003, KanGal* (2002).

[Deb+02] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions Evolution Computation* 6(2) (2002), pp. 182–197.

[Dee92] N. Deeb. "Simulated annealing in power systems". In: *In [Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics* (1992), pp. 1086–1089.

[DGR04] X. Delorme, X. Gandibleux, and J. Rodriguez. "GRASP for set packing problems". In: *European Journal of Operational Research* 153(3) (2004), pp. 564–580.

[Dem02] S. Dempe. "Foundations of bilevel programming". In: *Springer Science & Business Media* (2002).

[Der+08] Y Deremaux et al. "Environmental MDO and uncertainty hybrid approach applied to a supersonic business jet". In: *In 12th AIAA/ISSMO multidisciplinary analysis and optimization conference* Victoria, British Columbia Canada (2008), p. 5832.

[Des+15] S. Desale et al. "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey". In: *International Journal of Computer Engineering in Research Trends* 351(5) (2015), pp. 2349–7084.

[Des+16] V. Deshpande et al. "Plant layout optimization using CRAFT and ALDEP methodology". In: *Productivity Journal by National Productivity Council* 57(1) (2016), pp. 32–42.

[Des+22] A. Deshwal et al. "Bayesian optimization over permutation spaces". In: *In Proceedings of the AAAI Conference on Artificial Intelligence* 36(6) (2022), pp. 6515–6523.

[DO91] C.R. Dietrich and M.R. Osborne. “Estimation of covariance parameters in kriging via restricted maximum likelihood”. In: *Mathematical Geology* 23(1) (1991), pp. 119–135.

[Dig83] P.J. Diggle. “Statistical Analysis of Spatial Point Patterns”. In: *Academic Press* (1983).

[DTS08] L. Dimitriou, T. Tsekeris, and A. Stathopoulos. “Genetic Computation of Road Network Design and Pricing Stackelberg Games with Multiclass Users”. In: *Applications of Evolutionary Computing: EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings. Springer, Heidelberg.* (2008), pp. 669–678.

[DKB18] P. Diwedi, V. Kant, and K.K. Bharadwaj. “Learning path recommendation based on modified variable length genetic algorithm”. In: *Education and Information Technologies* 23 (2018), pp. 819–836.

[DP22] V. Dogan and S. Prestwich. “Bayesian Optimization with Multi-objective Acquisition Function for Bilevel Problems”. In: *In Irish Conference on Artificial Intelligence and Cognitive Science. Cham: Springer Nature Switzerland* (2022), pp. 409–422.

[DBS06] M. Dorigo, M. Birattari, and T. Stutzle. “Ant colony optimization”. In: *IEEE computational intelligence magazine* 1(4) (2006), pp. 28–39.

[Dow93] K.A. Dowsland. “Some experiments with simulated annealing techniques for packing problems”. In: *European Journal of Operational Research* 68(3) (1993), pp. 389–399.

[DRW03] T. Dunker, G. Radons, and E. Westkämper. “A coevolutionary algorithm for a facility layout problem”. In: *International Journal of Production Research* 41(15) (2003), pp. 3479–3500.

[DMB16] C. Durantin, J. Marzat, and M. Balesdent. “Analysis of multi-objective Kriging-based methods for constrained global optimization”. In: *Computational Optimization and Applications* 63(3) (2016), pp. 903–926.

[Edd04] S.R. Eddy. “What is dynamic programming?” In: *Nature Biotechnology* 22(7) (2004), pp. 909–910.

[EGH70] H.K. Edwards, B.E. Gillett, and M.E. Hale. “Modular Allocation Technique (MAT)”. In: *Management Science* 17(3) (1970), pp. 161–169.

[ER66] M. Efroymson and T.L. Ray. “A branch-bound algorithm for plant location”. In: *IOperations Research* 14(3) (1966), pp. 361–368.

[Ege03] J. Egeblad. “Placement Techniques for VLSI Layout Using Sequence-Pair Legalization”. In: *Mémoire de D.E.A., Department of Computer Science, University of Copenhagen* (2003).

[EHG05] E. Elbeltagi, T. Hegazy, and D. Grierson. “Comparison among five evolutionary-based optimization algorithms”. In: *Advanced Engineering Informatics* 19(1) (2005), pp. 43–53.

[EAE22] A. Ellithy, O. Abdelkhalik, and J. Englander. "Multi-objective hidden genes genetic algorithm for multigravity-assist trajectory optimization". In: *Journal of Guidance, Control, and Dynamics* 45(7) (2022), pp. 1269–1285.

[EN10] A. Emami and P. Noghreh. "New approach on optimization in placement of wind turbines within wind farm by genetic algorithms". In: *Renewable Energy* 35(7) (2010), pp. 1559–1564.

[Emm+92] H. Emmons et al. "Storm: Personal Version 3.0/3.5, Quantitative Modeling for Decision Support". In: *Prentice Hall* (1992).

[EMS09] H. Jair Escalante, M. Montes, and L.E. Sucar. "Particle swarm model selection". In: *Journal of Machine Learning Research* 10(2) (2009).

[Esp+11] A. Espinal et al. "Comparison of PSO and DE for training neural networks". In: *In 2011 10th Mexican International Conference on Artificial Intelligence* (2011), pp. 83–87.

[FS22] F.Guerriero and F. Saccomanno. "A hierarchical hyper-heuristic for the bin packing problem". In: *Soft Computing* (2022), pp. 1–14.

[FT16] M. Fakoor and M. Taghinezhad. "Layout and configuration design for a satellite with variable mass using hybrid optimization method". In: *In Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 230 (2016), pp. 360–377.

[Fan+19] Z. Fan et al. "Push and pull search for solving constrained multi-objective optimization problems". In: *Swarm Evolutionary Computation* 44 (2019), pp. 665–679.

[FLS06] K.T. Fang, R. Li, and A. Sudjianto. "Design and modeling for computer experiments". In: *Chapman & Hall CRC* (2006).

[FS14] J. Feng and W.Z. Shen. "Wind farm layout optimization in complex terrain: A preliminary study on a Gaussian hill". In: *In Journal of Physics: Conference Series. IOP Publishing.* 524 (2014), p. 012146.

[FR95] T.A. Feo and M.G.C. Resende. "Greedy randomized adaptive search procedures". In: *Journal of Global Optimization* 6 (1995), pp. 109–133.

[Fes14] P. Festa. "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems". In: *In 2014 16th International Conference on Transparent Optical Networks (ICTON)* (2014), pp. 1–20.

[FH19] M. Feurer and F. Hutter. "Hyperparameter optimization". In: *Automated machine learning: Methods, systems, challenges* (2019), pp. 3–33.

[FR78] L.R. Foulds and D.F. Robinson. "Graph theoretic heuristics for the plant layout problem". In: *Journal of Production Research* 16(1) (1978), pp. 27–37.

[Fra+18] C.P. Frank et al. "Evolutionary multi-objective multi-architecture design space exploration methodology". In: *Optimization and Engineering* 19 (2018), pp. 359–381.

[Fra18] P.I. Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[FHL13] Z.H. Fu, W.Q. Huang, and Z.P. Lü. "Iterated tabu search for the circular open dimension problem". In: *European Journal of Operational Research* 225(2) (2013), pp. 236–243.

[FLS17] F. Furini, I. Ljubić, and M. Sinnl. "An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem". In: *European journal of operational research* 262(2) (2017), pp. 438–448.

[GA11] A. Gad and O. Abdelkhalik. "Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization". In: *AIAA Journal of Spacecraft and Rockets* 48(4) (2011), pp. 629–641.

[Gal+21] T. Galanos et al. "ARCH-Elites: Quality-diversity for urban design". In: *In Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2021), pp. 313–314.

[Gam+22] J. Gamot et al. "Component Swarm Optimization Using Virtual Forces for Solving Layout Problems". In: *In Swarm Intelligence: 13th International Conference, ANTS 2022, Málaga, Spain, November 2–4, 2022, Proceedings* Cham: Springer International Publishing. (2022), pp. 292–299.

[GH20] E.C. Garrido-Merchán and D. Herández-Lobato. "Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes". In: *Neurocomputing* 380 (2020), pp. 20–35.

[GC02] S.S. Ge and Y.J. Cui. "Dynamic motion planning for mobile robots using potential field method". In: *Autonomous Robots* 13 (2002), pp. 207–222.

[Gen+19] L. Gentile et al. "Structured-chromosome GA optimisation for satellite tracking". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic* (2019), pp. 1955–1963.

[Geo+99] M.C. Georgiadis et al. "A general mathematical programming approach for process plant layout". In: *Computers & Chemical Engineering* 23(7) (1999), pp. 823–840.

[GA16] N.S. Ghaisas and C.L. Archer. "Geometry-based models for studying the effects of wind farm layout". In: *Journal of Atmospheric and Oceanic Technology* 33(3) (2016), pp. 481–501.

[GB10] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *In Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256.

[Glo86] F. Glover. “Future paths for integer programming and links to artificial intelligence”. In: *Computers & operations research* 13(5) (1986), pp. 533–549.

[Gol89] D.E. Goldberg. “Genetic Algorithms in Search, Optimization, and Machine Learning”. In: *Addison-Wesley Professional* (1989).

[Gon+11] J.S. González et al. “Overall design optimization of wind farms”. In: *Renewable Energy* 36(7) (2011), pp. 1973–1982.

[Gow71] J.C. Gower. “A general coefficient of similarity and some of its properties”. In: *Biometrics* 27(4) (1971), p. 857.

[Gre00] J. Grefenstette. “Rank-based selection”. In: *Evolutionary Computation* 1 (2000), pp. 187–194.

[GL17] Z. Guo and B. Li. “Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system”. In: *Frontiers of Architectural Research* 6(1) (2017), pp. 53–62.

[HB97] A. Ben Hadj-Alouane and J. C. Bean. “A genetic algorithm for the multiple-choice integer program”. In: *Operational Research* 45(1) (1997), pp. 92–101.

[HA21] S.H. Haji and A.M. Abdulazeez. “Comparison of optimization techniques based on gradient descent algorithm: A review”. In: *PalArch’s Journal of Archaeology of Egypt/Egyptology* 18(4) (2021), pp. 2715–2743.

[HI15] I.M. Hakim and V. Istiyanti. “Improvement of layout production facilities for a secondary packaging area of a pharmaceutical company in Indonesia using the CORELAP method”. In: *International Journal of Technology* 6(6) (2015), pp. 1006–1016.

[Hal16] M. Halstrup. “Black-Box Optimization of Mixed Discrete-Continuous Optimization Problems”. In: *Ph.D. Thesis, TU Dortmund* (2016).

[HM06] W. Halter and S. Mostaghim. “Bilevel optimization of multi-component chemical systems using particle swarm optimization”. In: *In: World Congress on Computational Intelligence (WCCI 2006)* (2006), pp. 1240–1247.

[Ham+22] R. Hamano et al. “CMA-ES with margin: Lower-bounding marginal probability for mixed-integer black-box optimization”. In: *In Proceedings of the Genetic and Evolutionary Computation Conference* (2022), pp. 639–647.

[HO01] N. Hansen and A. Ostermeier. “Completely Derandomized Self-Adaptation in Evolution Strategies”. In: *Evolutionary Computation* 9(2) (2001), pp. 159–195.

[Has+17] R.A. Hasan et al. “A comprehensive study: Ant colony optimization (ACO) for facility layout problem”. In: *16th RoEduNet conference: networking in education and research (RoEduNet) IEEE* (2017), pp. 1–8.

[HHS86] M.M. Hassan, G.L. Hogg, and D.R. Smith. "SHAPE: a construction algorithm for area placement evaluation". In: *International Journal of Production Research* 24(5) (1986), pp. 1283–1295.

[Has+05] R. Hassan et al. "A comparison of particle swarm optimization and the genetic algorithm". In: *In 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference* (2005), p. 1897.

[HKY17] E. Hazan, A. Klivans, and Y. Yuan. "Hyperparameter optimization: A spectral approach". In: *arXiv preprint arXiv:1706.00764* (2017).

[He+13] K. He et al. "A coarse-to-fine quasi-physical optimization method for solving the circle packing problem with equilibrium constraints". In: *Computers & Industrial Engineering* 66(4) (2013), pp. 1049–1060.

[He+15] K. He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *In Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.

[He+18] K. He et al. "An efficient quasi-physical quasi-human algorithm for packing equal circles in a circular container". In: *Computers & Operations Research* 92 (2018), pp. 26–36.

[HU17] J. Herman and W. Usher. "SALib: An open-source Python library for sensitivity analysis". In: *Journal of Open Source Software* 2(9) (2017), p. 97.

[HCD21] A. Herrán, J.M. Colmenar, and A. Duarte. "An efficient variable neighborhood search for the space-free multi-row facility layout problem". In: *European Journal of Operational Research* 295(3) (2021), pp. 893–907.

[Hil63] F.S. Hillier. "Quantitative tools for plant layout analysis". In: *Journal of Industrial Engineering* 1 (1963), pp. 33–40.

[HC66] F.S. Hillier and M.M. Connors. "Quadratic assignment problem algorithms and the location of indivisible facilities". In: *Management Science* 13(1) (1966), pp. 42–57.

[HGR21] J.G. Hobbie, A.H. Gandomi, and I. Rahimi. "A comparison of constraint handling techniques on NSGA-II". In: *Archives of Computational Methods in Engineering* (2021), pp. 1–16.

[Hol92] J.H. Holland. "Genetic Algorithms". In: *Scientific american* 267(1) (1992), pp. 66–73.

[HQL94] A. Homaifar, C. X. Qi, and S. H. Lai. "Constrained optimization via genetic algorithms". In: *Simulation* 62(4) (1994), pp. 242–253.

[Hor+19] D. Horn et al. "Surrogates for hierarchical search spaces: the wedge-kernel and an automated analysis". In: *In Proceedings of the genetic and evolutionary computation conference* (2019), pp. 916–924.

[Hos+14] S.S. Hosseini et al. "Multi-floor facility layout improvement using systematic layout planning". In: *Advanced Materials Research* 845 (2014), pp. 532–537.

[Hou+19] P. Hou et al. "A review of offshore wind farm layout optimization and electrical system design methods". In: *Journal of Modern Power Systems and Clean Energy* 7(5) (2019), pp. 975–986.

[HL04] B. Huang and N. Liu. "Bilevel programming approach to optimizing a logistic distribution network with balancing requirements". In: *Transportation Research Record* 1894(1) (2004), pp. 188–197.

[HH09] W. Huang and K. He. "A caving degree approach for the single container loading problem". In: *European Journal of Operational Research* 196(1) (2009), pp. 93–101.

[HY11a] W. Huang and T. Ye. "Global optimization method for finding dense packings of equal circles in a circle". In: *European Journal of Operational Research* 210(3) (2011), pp. 474–481.

[HY11b] W. Huang and T. Ye. "Global optimization method for finding dense packings of equal circles in a circle". In: *European Journal of Operational Research* 210(3) (2011), pp. 474–481.

[HLG21] J. Huo, J. Liu, and H. Gao. "An NSGA-II algorithm with adaptive local search for a new double-row model solution to a multi-floor hospital facility layout problem". In: *Applied Sciences* 11(4) (2021), p. 1758.

[HT09] J.Z. Huo and H.F. Teng. "Optimal layout design of a satellite module using a coevolutionary method with heuristic rules". In: *Journal of Aerospace Engineering* 22 (2009), pp. 101–111.

[HW07] B. Hutt and K. Warwick. "Synapsing variable-length crossover: meaningful crossover for variable-length genomes". In: *IEEE Transaction Evolution Computation* 11(1) (2007), pp. 118–131.

[HHK11] F. Hutter, H. Hoos, and K.Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *In Coello, C. (ed.) Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION'11). Lecture Notes in Computer Science. Springer.* 6683 (2011), pp. 507–523.

[HO13a] F. Hutter and M.A. Osborne. "SA kernel for hierarchical parameter spaces". In: *arXiv preprint arXiv:1310.5738* (2013).

[HO13b] F. Hutter and Michael A. Osborne. "A Kernel for Hierarchical Parameter Spaces". In: *Technical Report arXiv:1310.5738. arXiv* (2013).

[Ili+17] I. Ilievski et al. "Efficient Hyperparameter Optimization for Deep Learning Algorithms Using Deterministic RBF Surrogates". In: *In Proceedings of the AAAI conference on artificial intelligence* 31(1) (2017).

[IES14] N. Izadinia, K. Eshghi, and M.H. Salmani. "A robust model for multi-floor layout problem". In: *Computers & Industrial Engineering* 78 (2014), pp. 127–134.

[Jac10] G. Jacquenot. "Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel". In: *Doctoral dissertation, Ecole Centrale de Nantes (ECN) (in french)* (2010).

[Jac+09] G. Jacquenot et al. “2d multi-objective placement algorithm for free-form components”. In: *In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* 49026 (2009), pp. 239–248.

[Jak96] S. Jakobs. “On genetic algorithms for the packing of polygons”. In: *European Journal of Operational Research* 88(1) (1996), pp. 165–181.

[JRS20] N.P. Jati, A.D.I. Rahayu, and S.E. Salsbila. “Facility layout design with CORELAP algorithm for Educational Tour”. In: *In IOP Conference Series: Materials Science and Engineering. IOP Publishing.* 982(1) (2020), p. 012060.

[Jen+17] R. Jenatton et al. “Bayesian optimization with tree-structured dependencies”. In: *In International Conference on Machine Learning* (2017), pp. 1655–1664.

[Ji+22] K. Ji et al. “A virtual force interaction scheme for multi-robot environment monitoring”. In: *Robotics and Autonomous Systems* 149 (2022), p. 103967.

[JV15] Y. Jiao and J.-P. Vert. “The Kendall and Mallows kernels for permutations”. In: *In International Conference on Machine Learning* (2015), pp. 1935–1944.

[Joh82] R.V. Johnson. “SPACECRAFT for multi-floor layout planning”. In: *Management Science* 28(4) (1982), pp. 407–417.

[JH94] J. A. Joines and C. R. Houck. “On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA’s”. In: *In Proceedings of the first IEEE conference on evolutionary computation. IEEE World Congress on Computational Intelligence* (1994), pp. 579–584.

[JMW98] D.R. Jones, S. Matthias, and J. Welch William. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global Optimization* 13 (1998), pp. 455–492.

[JSW98] D.R. Jones, M. Schonlau, and W.J. Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global optimization* 14(4) (1998), p. 455.

[KŠ18] P. Kadlec and V. Šeděnka. “Particle swarm optimization for problems with variable number of dimensions”. In: *Engineering Optimization* 50(3) (2018), pp. 382–399.

[KSE17] A. El Kady, S.A. Sami, and A.M. Eldeib. “A Two Stage Heuristics for Improvement of Existing Multi Floor Healthcare Facility Layout”. In: *In ICBBT’17: 9thInternational Conference on Bioinformatics and Biomedical Technology, Lisbon Portugal* (2017), pp. 97–101.

[KLM96] L.P. Kaelbling, M.L. Littman, and A.W. Moore. “Reinforcement learning: A survey”. In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.

[KM97] B.K. Kaku and J.B. Mazzola. "A tabu-search heuristic for the dynamic plant layout problem". In: *INFORMS Journal on Computing* 9(4) (1997), pp. 374–384.

[KC17] S. Kang and J. Chae. "Harmony search for the layout design of an unequal area facility". In: *Expert Systems with Applications* 79 (2017), pp. 269–281.

[Kap+19] M.S. Kapadia et al. "Impact of scheduling policies on the performance of an additive manufacturing production system". In: *Procedia Manufacturing* 39 (2019), pp. 447–456.

[KSN22] H. Karateke, R. Sahin, and S. Niroomand. "A hybrid Dantzig-Wolfe decomposition algorithm for the multi-floor facility layout problem". In: *Expert Systems with Applications* 206 (2022), p. 117845.

[KE95] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *In Proceedings of ICNN'95-international conference on neural networks* 4 (1995), pp. 1942–1948.

[KE97] J. Kennedy and R.C. Eberhart. "A discrete binary version of the particle swarm algorithm". In: *In 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation* 5 (1997), pp. 4104–4108.

[KM84] R.L. Ketcham and E.M. Malstrom. "Computer assisted facilities layout algorithm using graphics". In: *In 1984 Fall Industrial Engineering Conference– Integrating People and Technology* (1984), pp. 88–95.

[Kha73] T.M. Khalil. "Facilities relative allocation technique (FRAT)". In: *International Journal of Production Research* 11(2) (1973), pp. 183–194.

[Kha+13] B. Khandelwal et al. "Hydrogen powered aircraft: The future of air transport". In: *Progress in Aerospace Sciences* 60 (2013), pp. 45–59.

[KB86] O. Khatib and J. Burdick. "Motion and force control of robot manipulators". In: *In Proceedings. 1986 IEEE international conference on robotics and automation* 3 (1986), pp. 1381–1386.

[Kie+21] E. Kieffer et al. "Bayesian optimization approach of general bi-level problems". In: *In Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2021), pp. 1614–1621.

[KK00] J.G. Kim and Y.D. Kim. "Layout planning for facilities with fixed shapes and input and output points". In: *International Journal of Production Research* 38(18) (2000), pp. 4635–4653.

[KB] D.P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* ().

[KJV83] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. "Optimization by simulated annealing". In: *Science* 220 (1983), pp. 671–680.

[Koc+15] P. Koch et al. "A new repair method for constrained optimization". In: *In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), pp. 273–280.

[Koh07] A. Koh. “Solving transportation bi-level programs with differential evolution”. In: *In: IEEE Congress on Evolutionary Computation (CEC 2007)* (2007), pp. 2243–2250.

[KYS08] D. Kohara, H. Yamamoto, and A. Suzuki. “Efficient algorithms based on branch and bound methods for multi floor facility layout problems”. In: *In Proceedings of the 9th Asia pacific industrial engineering & management systems conference, Bali, Indonesia* (2008), pp. 387–395.

[KAD01] M. Kokkolaras, C. Audet, and J.E. Dennis. “Mixed Variable Optimization of the Number and Composition of Heat Intercepts in a Thermal Insulation System”. In: *Optimization and Engineering* 2(1) (2001), pp. 5–29.

[Kon85] S. Konz. “Facility Design”. In: *Wiley, NY* (1985).

[KB57] T.C. Koopmans and M. Beckmann. “Assignment problems and the location of economic activities”. In: *Econometrica: journal of the Econometric Society* (1957), pp. 53–76.

[KG14] R. Kothari and D. Ghosh. “A scatter search algorithm for the single row facility layout problem”. In: *Journal of Heuristics* 20 (2014), pp. 125–142.

[KS05] N. Krasnogor and J. Smith. “A tutorial for competent memetic algorithms: model, taxonomy, and design issues”. In: *IEEE transactions on Evolutionary Computation* 9(5) (2005), pp. 474–488.

[KAA10] H. Küçükaydın, N. Aras, and I.K. Altinel. “A hybrid tabu search heuristic for a bilevel competitive facility location model”. In: *In Hybrid Metaheuristics: 7th International Workshop, HM 2010, Vienna, Austria, October 1-2, 2010. Springer Berlin Heidelberg.* 7 (2010), pp. 31–45.

[KH87] A. Kusiak and S.S. Heragu. “The facility layout problem”. In: *European Journal of Operational Research* 29(3) (1987), pp. 229–251.

[Lag+23] N.D. Lagaros et al. “Constraint handling techniques for metaheuristics: a state-of-the-art review and new variants”. In: *Optimization and Engineering* (2023), pp. 1–48.

[LM03] M. Laguna and R.C. Martí. “Scatter search: methodology and implementations in C”. In: *Springer Science & Business Media* (2003).

[LBE18] C. Lamini, S. Benhlima, and A. Elbekri. “Genetic algorithm based approach for autonomous mobile robot path planning”. In: *Procedia Computer Science* 127 (2018), pp. 180–189.

[Law93] J. Lawrence. “Introduction to neural networks”. In: *California Scientific Software* (1993).

[LC12a] A. Layeb and S. Chenche. “A novel grasp algorithm for solving the bin packing problem”. In: *International Journal of Information Engineering and Electronic Business* 4(2) (2012), p. 8.

[LRJ05] K.Y. Lee, M.I. Roh, and H.S. Jeong. “An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages”. In: *Computers & Operations Research* 32(4) (2005), pp. 879–899.

[Lee67] R.C. Lee. “CORELAP-computerized relationship layout planning”. In: *Journal of Industrial Engineering* 8(3) (1967), pp. 195–200.

[LL02] Y.H. Lee and M.H. Lee. “A shape-based block layout approach to facility layout problems using hybrid genetic algorithm”. In: *Computers & Industrial Engineering* 42(2-4) (2002), pp. 237–248.

[LS11] J. Lehman and K.O. Stanley. “Evolving a diversity of virtual creatures through novelty search and local competition”. In: *In Proc. of the Thirteenth Annual Conference on Genetic and Evolutionary Computation (GECCO’11), Dublin, Ireland* (2011), pp. 211–218.

[LKM17] O.K. Lekan, O.I. Kayode, and A. Abdulrazak Morenikeji. “Analysis of plant layout design for operational efficiency with craft algorithms”. In: *Acta Universitatis Danubius. Œconomica* 13(4) (2017).

[LD04] J. Levine and F. Ducatelle. “Ant colony optimization and local search for bin packing and cutting stock problems”. In: *Journal of the Operational Research society* 55(7) (2004), pp. 705–716.

[LI+18] C. LI et al. “High dimensional Bayesian optimization using dropout”. In: *arXiv preprint arXiv:1802.05400* (2018).

[Li+18] L. Li et al. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *Journal of Machine Learning Research* 18(185) (2018), pp. 1–52.

[Li+17] W. Li et al. “A novel extension algorithm for optimized Latin hypercube sampling”. In: *Journal of Statistical Computation and Simulation* 87(13) (2017), pp. 2549–2559.

[LZW18] Y. Li, P. Zhang, and Y. Wu. “Public recharging infrastructure location strategy for promoting electric vehicles: a bi-level programming approach”. In: *Journal of Cleaner Production* 172 (2018), pp. 2720–2734.

[LC08] L.Y. Liang and W.C. Chao. “The strategies of tabu search technique for facility layout optimization”. In: *Automation in Construction* 17(6) (2008), pp. 657–669.

[Lia+13] T. Liao et al. “Ant colony optimization for mixed-variable optimization problems”. In: *IEEE Transactions on Evolutionary Computation* 18(4) (2013), pp. 503–518.

[LR21] D. Lieber and G. Reinhart. “A bi-level optimisation approach for assembly line design using a nested genetic algorithm”. In: *International Journal of Production Research* 59(24) (2021), pp. 7560–7575.

[LC12b] L.C. Lien and M.Y. Cheng. “A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization”. In: *Expert systems with applications* 39(10) (2012), pp. 9642–9650.

[LPK17] Zhen Yang Lim, S.G. Ponnambalam, and et Kazuhiro Izui. "Multi-objective hybrid algorithms for layout optimization in multi-robot cellular manufacturing systems". In: *Knowledge-Based Systems* 120 (2017), pp. 87–98.

[LDD18] Y. Lin, W. Du, and W. Du. "Multi-objective differential evolution with dynamic hybrid constraint handling mechanism". In: *Soft Computing* (2018), pp. 1–15.

[LL12] A. Lipowski and D. Lipowska. "Roulette-wheel selection via stochastic acceptance". In: *Physica A: Statistical Mechanics and its Applications* 391(6) (2012), pp. 2193–2196.

[Liu12] J Liu. "Constrained Layout Optimization in Satellite Cabin Using a Multiagent Genetic Algorithm". In: *Asia-Pacific Conference on simulated evolution and learning* Springer, Berlin, Heidelberg, Deutschland (2012), pp. 440–449.

[Liu+15] J. Liu et al. "Heuristic-based energy landscape paving for the circular packing problem with performance constraints of equilibrium". In: *Physica A: Statistical Mechanics and its Applications* 431 (2015), pp. 166–174.

[Liu+16a] J. Liu et al. "A heuristic quasi-physical algorithm with coarse and fine adjustment for multi-objective weighted circles packing problem". In: *Computers & Industrial Engineering* 101 (2016), pp. 416–426.

[Liu+18] J. Liu et al. "Multi-objective particle swarm optimization algorithm based on objective space division for the unequal-area facility layout problem". In: *Expert Systems with Applications* 102 (2018), pp. 172–192.

[LLC10] J.F. Liu, G. Li, and D.B. Chen. "Two-dimensional equilibrium constraint layout using simulated annealing". In: *Computers & Industrial Engineering* 59(4) (2010), pp. 530–536.

[Liu+16b] J.F. Liu et al. "Multi-objective layout optimization of a satellite module using the Wang-Landau sampling method with local search". In: *Frontiers of Information Technology & Electronic Engineering* 17(6) (2016), pp. 527–542.

[LT08] Z.W. Liu and H.F. Teng. "Human–computer cooperative layout design method and its application". In: *Computers & Industrial Engineering* 55(4) (2008), pp. 735–757.

[Loh96] W.L. Loh. "On Latin hypercube sampling". In: *The annals of statistics* 24(5) (1996), pp. 2058–2080.

[LHC09] M. Lozano, F. Herrera, and J.R. Cano. "Replacement strategies to preserve useful diversity in steady-state genetic algorithms". In: *Information sciences* 178(23) (2009), pp. 4421–4433.

[LVM05] S. Lucidi, P. Veronica, and S. Marco. "An algorithm model for mixed variable programming". In: *SIAM Journal of Optimization* 15(4) (2005), pp. 1057–1084.

[LP04] S. Luicidi and V. Piccialli. “A Derivative-Based Algorithm for a Particular Class of Mixed Variable Optimization Problems”. In: *Optimization Methods and Software* 14(3-4) (2004), pp. 317–387.

[LPS05] S. Luicidi, V. Piccialli, and M. Sciandrone. “An Algorithm Model for Mixed Variable Programming”. In: *SIAM Journal on Optimization* 15(4) (2005), pp. 1057–1084.

[LRP22] Q. Luo, Y. Rao, and D. Peng. “GA and GWO algorithm for the special bin packing problem encountered in field of aircraft arrangement”. In: *Applied Soft Computing* 114 (2022), p. 108060.

[Ma+22] M. Ma et al. “Optimal sizing and operations of shared energy storage systems in distribution networks: A bi-level programming approach”. In: *Applied Energy* 307 (2022), p. 118170.

[MLa18] X. Ma, X. Li, and Q. Zhang et al. “A survey on cooperative co-evolutionary algorithms”. In: *IEEE Transactions on Evolutionary Computation* 23 (2018), pp. 421–441.

[Mah92] S.W. Mahfoud. “Crowding and preselection revisited”. In: *In R. Männer and B. Manderick Eds, Parallel problem solving from nature. Elsevier.* 2 (1992), pp. 27–36.

[Mah94] S.W. Mahfoud. “Crossover interactions among niches”. In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* (1994), pp. 188–193.

[MTK96] K.F. Man, K.S. Tang, and S. Kwong. “Genetic algorithms: concepts and applications [in engineering design]”. In: *IEEE transactions on Industrial Electronics* 43(5) (1996), pp. 519–534.

[MBS18] I.B. Mansour, M. Basseur, and F. Saubion. “A multi-population algorithm for multi-objective knapsack problem”. In: *Applied Soft Computing* 70 (2018), pp. 814–825.

[MLG06] R. Martí, M. Laguna, and F. Glover. “Principles of scatter search”. In: *European Journal of Operational Research* 169(2) (2006), pp. 359–372.

[MPA94] R. Mathieu, L. Pittard, and G. Anandalingam. “Genetic algorithm based approach to bi-level linear programming”. In: *Operational Research* 28 (1994), pp. 1–21.

[MPA00] R. Mathieu, L. Pittard, and G. Anandalingam. “Genetic-algorithms-based approach for bilevel programming models”. In: *Journal of Transportation Engineering* 126(2) (2000), pp. 115–120.

[MIY14] K. Matsuzaki, T. Irohara, and K. Yoshimoto. “Heuristic algorithm to solve the multi-floor layout problem with the consideration of elevator utilization”. In: *Computers & Industrial Engineering* 36(2) (2014), pp. 487–502.

[Mec+01] M. Meckesheimer et al. “Metamodeling of combined discrete/continuous responses”. In: *AIAA Journal* 39(10) (2001), pp. 1950–1959.

[MB96] R.D. Meller and Y.A. Bozer. "A new simulated annealing algorithm for the facility layout problem". In: *The International Journal of Production Research* 34(6) (1996), pp. 1675–1692.

[MB97] R.D. Meller and Y.A. Bozer. "Alternative approaches to solve the multi-floor facility layout problem". In: *Journal of Manufacturing Systems* 16(3) (1997), pp. 192–203.

[MSK21] X. Meng, H. Sun, and J. Kang. "Equipment layout optimization based on human reliability analysis of cabin environment". In: *Journal of Marine Science and Engineering* 9(11) (2021), p. 1263.

[MC11] E. Mezura-Montes and C.A.C Coello. "Constraint-handling in nature-inspired numerical optimization: past, present and future". In: *Swarm and Evolutionary Computation* 1(4) (2011), pp. 173–194.

[MMT03] K. Miettinen, M.M. Mäkelä, and J. Toivanen. "Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms". In: *Journal of Global Optimization* 27(4) (2003), pp. 427–446.

[MPV13] A. Migdalas, P.M. Pardalos, and P. Värbrand. "Multilevel Optimization: Algorithms and Applications". In: *Springer Science & Business Media, London, Heidelberg, 2013* (2013).

[Mir+21] A. Mirhoseini et al. "A graph placement methodology for fast chip design". In: *Nature* 594(7862) (2021), pp. 207–212.

[MO04] N. Mithulananthan and T. Oo. "Distributed generator placement in power distribution system using genetic algorithm to reduce losses". In: *Science & Technology Asia* (2004), pp. 55–62.

[MH97] N. Mladenović and P. Hansen. "Variable neighborhood search". In: *Computers & operations research* 24(11) (1997), pp. 1097–1100.

[Moh+21] A. Mohammadi et al. "Design and modeling of adaptive IIR filtering systems using a weighted sum-variable length particle swarm optimization". In: *Applied Soft Computing* 109 (2021), p. 107529.

[MVV13] M. Montemurro, A. Vinceti, and P. Vannucci. "The automatic dynamic penalization method (ADP) for handling constraints with genetic algorithms". In: *Computer Methods in Applied Mechanics and Engineering* 1256 (2013), pp. 70–87.

[Mon90] B. Montreuil. "A Modelling Framework for Integrating Layout Design and Flow Network Design". In: *In Proceedings from the Material Handling Research Colloquium, Hebron, Kentucky* (1990), pp. 43–58.

[MRG87] B. Montreuil, H.D. Ratcliff, and M. Goetschalckx. "Matching based interactive facility layout". In: *IIE Transactions* 19(3) (1987), pp. 271–279.

[MC15] J.B. Mouret and J. Clune. "Illuminating search spaces by mapping elites". In: *arXiv preprint arXiv:1504.04909* (2015).

[ND02] P.K.S. Nain and K. Deb. "A computationally effective multiobjective search and optimization technique using coarse-to-fine grain modeling". In: *In Proceedings on Parallel Problem Solving Nature Workshop Evolutionary Multiobjective Optimization* (2002), pp. 2081–2088.

[NDN22] A. Nikfarjam, A. Viet Do, and F. Neumann. "Analysis of Quality Diversity Algorithms for the Knapsack Problem". In: *In Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part II. Cham: Springer International Publishing* (2022), pp. 413–427.

[Nin+17] W. Ning et al. "Constrained multi-objective optimization using constrained non-dominated sorting combined with an improved hybrid multi-objective evolutionary algorithm". In: *Engineering Optimization* 49(10) (2017), pp. 1645–1664.

[NAO15] H.M. Nyew, O. Abdelkhalik, and N. Onder. "Structured-chromosome evolutionary algorithms for variable-size autonomous interplanetary trajectory planning optimization". In: *Journal of Aerospace Information Systems* 12(3) (2015), pp. 314–328.

[OK97] I. Ono and S. Kobayashi. "A real-coded genetic algorithm for functional optimization using unimodal normal distribution crossover". In: *In ICGA-7, 7th International Conference on Genetic Algorithms* (1997), pp. 246–253.

[OMS19] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda. "Mobile robot path planning using membrane evolutionary artificial potential field". In: *Applied Soft Computing* 77 (2019), pp. 236–251.

[ONY22] D. Otaki, H. Nonaka, and N. Yamada. "Thermal design optimization of electronic circuit board layout with transient heating chips by using Bayesian optimization and thermal network model". In: *International Journal of Heat and Mass Transfer* 184 (2022), p. 122263.

[Oua+20] A. Ouali et al. "Variable neighborhood search for graphical model energy minimization". In: *Artificial Intelligence* 278 (2020), p. 103194.

[PSa+23] P.Saves et al. "A mixed-categorical correlation kernel for Gaussian process". In: *Neurocomputing* (2023), p. 126472.

[Pat+19] B.K. Patle et al. "A review: On path planning strategies for navigation of mobile robot". In: *Defence Technology* 15(4) (2019), pp. 582–606.

[PP02] D.I. Patsiatzis and L.G. Papageorgiou. "Optimal multi-floor process plant layout". In: *Computers & Chemical Engineering* 26(4-5) (2002), pp. 575–583.

[PC11] Y. Pei and Y. Ci. "Study on bi-level planning model and algorithm optimizing highway network layout". In: *In: Eastern Asia Society for Transportation Studies* 38(1) (2011), pp. 320–327.

[Pel20] J. Pelamatti. "Mixed-variable Bayesian optimization: application to aerospace system design". In: *Doctoral dissertation, Université de Lille* (2020).

[Pel+20] J. Pelamatti et al. "Overview and comparison of gaussian process-based surrogate models for mixed continuous and discrete variables: Application on aerospace design problems". In: *High-Performance Simulation-Based Optimization* (2020), pp. 189–224.

[Pel+21] J. Pelamatti et al. "Bayesian optimization of variable-size design space problems". In: *Optimization and Engineering* 22 (2021), pp. 387–447.

[PLG17] C. Peng, H.L. Liu, and F. Gu. "An evolutionary algorithm with directed weights for constrained multi-objective optimization". In: *Applied Soft Computing* 60 (2017), pp. 613–622.

[Pen+18] Y. Peng et al. "An improved genetic algorithm based robust approach for stochastic dynamic facility layout problem". In: *Discrete Dynamics in Nature and Society* (2018), pp. 1–8.

[PAO18] J. Peralta, M. Andretta, and J.F. Oliveira. "Packing Circles and Irregular Polygons using Separation Lines". In: *International Conference on Operations Research and Enterprise Systems* (2018), pp. 71–77.

[Pha+17] N. Phan et al. "Coverage optimization of LTE networks based on antenna tilt adjusting considering network load". In: *China Communications* 14(5) (2017), pp. 48–58.

[PWG13] V. Picheny, T. Wagner, and D. Ginsbourger. "A benchmark of kriging-based infill criteria for noisy optimization". In: *Structural and Multidisciplinary Optimization* 48 (2013), pp. 607–626.

[Pot+94] Potter et al. "A cooperative coevolutionary approach to function optimization". In: *International Conference on Parallel Problem Solving from Nature* Springer, Berlin, Heidelberg (1994), pp. 249–257.

[PRR14] N.H. Prasad, G. Rajyalakshmi, and A.S. Reddy. "A typical manufacturing plant layout design using CRAFT algorithm". In: *Procedia Engineering* 97 (2014), pp. 1908–1814.

[PBD19] R. Priem, N. Bartoli, and Y. Diouane. "On the Use of Upper Trust Bounds in Constrained Bayesian Optimization Infill Criteria". In: *AIAA Aviation 2019 Forum* (2019), p. 2986.

[PM12] L. Pronzato and W.G. Müller. "Design of computer experiments: space filling and beyond". In: *Statistics and Computing* 22 (2012), pp. 681–701.

[PSS16] J.K. Pugh, L.B. Soros, and K.O. Stanley. "Quality diversity: A new frontier for evolutionary computation". In: *Frontiers in Robotics and AI* (2016), p. 40.

[QL16] H. Qiu and Y. Liu. "Novel heuristic algorithm for large-scale complex optimization". In: *Procedia Computer Science* 80 (2016), pp. 744–751.

[QS11] B.Y. Qu and P.N. Suganthan. “Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods”. In: *Engineering Optimization* 43 (2011), pp. 403–416.

[Qui+15] M. Quiroz-Castellanos et al. “A grouping genetic algorithm with controlled gene transmission for the bin packing problem”. In: *Computers & Operations Research* 55 (2015), pp. 52–64.

[Rah+23] I. Rahimi et al. “A Review on Constraint Handling Techniques for Population-based Algorithms: from single-objective to multi-objective optimization”. In: *Archives of Computational Methods in Engineering* 30(3) (2023), pp. 2181–2209.

[RSS96] D.J. Ram, T.H. Sreenivas, and K.G. Subramaniam. “Parallel simulated annealing algorithms”. In: *Journal of parallel and distributed computing* 37(2) (1996), pp. 207–212.

[Ran+17] S. Rana et al. “High dimensional Bayesian optimization with elastic Gaussian process”. In: *In International conference on machine learning.PMLR.* (2017), pp. 2883–2891.

[RNS09] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi. “GSA: a gravitational search algorithm”. In: *Information Sciences* 179(13) (2009), pp. 2232–2248.

[RW06] C.E. Rasmussen and C.K.I Williams. “Gaussian Processes for Machine Learning”. In: *The MIT Press* (2006).

[RCM20] M. Ravikumar, H. Cheng, and M. McCourt. “Bayesian Optimization 101”. In: *Progress in Aerospace Sciences* https://sigopt.com/blog/bayesian-optimization-101/ (2020).

[Ray+09] T. Ray et al. “Infeasibility Driven Evolutionary Algorithm for Constrained Optimization”. In: *Springer Berlin Heidelberg* (2009), pp. 145–165.

[Rip+13] K.S.N Ripon et al. “Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities”. In: *Swarm and Evolutionary Computation* 8 (2013), pp. 1–12.

[Rom+22] T. Romanova et al. “Balanced Circular Packing Problems with Distance Constraints”. In: *Computations* 10(7) (2022), p. 113.

[Rou+20] O. Roustant et al. “Group kernels for Gaussian process metamodels with categorical inputs”. In: *SIAM/ASA Journal on Uncertainty Quantification* 8(2) (2020), pp. 775–806.

[RR16] M. Rout and R. Roy. “Dynamic deployment of randomly deployed mobile sensor nodes in the presence of obstacles”. In: *Ad hoc networks* 46 (2016), pp. 12–22.

[Roy10] R K Roy. “A primer on the Taguchi method”. In: *Society of Manufacturing Engineers* (2010).

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. "Learning internal representations by error propagation". In: *Volume 1: Foundation. MIT Press.* (1986).

[RY00] T.P. Runarsson and X. Yao. "Stochastic ranking for constrained evolutionary optimization". In: *IEEE Transactions on evolutionary computation* 4(3) (2000), pp. 284–294.

[Rye+17] M.M. Ryerkerk et al. "Solving metameric variable-length optimization problems using genetic algorithms". In: *Genetic Programming and Evolvable Machines* 18 (2017), pp. 247–277.

[SG06] A. Saffari and I. Guyon. "Quickstart guide for CLOP". In: *Technical report, Graz University of Technology* (2006).

[SR17] H. Sajedi and S.F. Razavi. "DGSA: discrete gravitational search algorithm for solving knapsack problem". In: *Operational Research* 17 (2017), pp. 563–591.

[SL22] S. Sajedi and X. Liang. "Deep generative Bayesian optimization for sensor placement in structural health monitoring". In: *Computer-Aided Civil and Infrastructure Engineering* 37(9) (2022), pp. 1109–1127.

[Sal09] S. Salcedo-Sanz. "A survey of repair methods used as constraint handling techniques in evolutionary algorithms". In: *Computer Science Review* 3(3) (2009), pp. 175–192.

[SAA02] A. Salman, I. Ahmad, and S. Al-Madani. "Particle swarm optimization for task assignment problem". In: *Microprocessors and Microsystems* 26(8) (2002), pp. 363–371.

[Sal02] A. Saltelli. "Making best use of model evaluations to compute sensitivity indices". In: *Computer Physics Communications* 145(2) (2002), pp. 280–297.

[Sal+10] A. Saltelli et al. "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index". In: *Computer Physics Communications* 181(2) (2010), pp. 259–270.

[Sam13] M. Samorani. "The wind farm layout optimization problem". In: *Springer Berlin Heidelberg* (2013), pp. 21–38.

[SFT23] T.O.Q. Saraf, N. Fuad, and N.S.A.M. Taujuddin. "Framework of Meta-Heuristic Variable Length Searching for Feature Selection in High-Dimensional Data". In: *Computers* 12(1) (2023), p. 7.

[Sas02] M.J. Sasena. "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations". In: *PhD thesis* (2002).

[Sav+22] P. Saves et al. "Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design". In: *In AIAA SCITECH 2022 Forum* (2022), p. 0082.

[SJ00] P. Schachinger and H.L. Johannesson. “Computer modelling of design specifications”. In: *Journal of Engineering Design* 11(4) (2000), pp. 317–329.

[SWJ98] M Schonlau, W.J. Welch, and D.R. Jones. “Global versus local search in constrained optimization of computer models”. In: *Lecture Notes-Monograph Series* (1998), pp. 11–25.

[SE67] J.M. Seehof and W. Evans. “Automated Layout Design Program”. In: *Journal of Industrial Engineering* 28(1) (1967), pp. 69–75.

[Sem+18] A.C. Sembiring et al. “An application of CORELAP algorithm to improve the utilization space of the classroom”. In: *In Journal of Physics: Conference Series. IOP Publishing.* 1007(1) (2018), p. 012026.

[SH06] D.Y. Sha and C.Y. Hsu. “A hybrid particle swarm optimization for job shop scheduling problem”. In: *Computers & Industrial Engineering* 51(4) (2006), pp. 791–808.

[Sha+15] B. Shariari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104(1) (2015), pp. 148–175.

[She+07] P.S. Shelokar et al. “Particle swarm and ant colony algorithms hybridized for improved continuous optimization”. In: *Applied Mathematics and Computation* 188(1) (2007), pp. 129–142.

[SLZ11] Y. Shi, B. Li, and Z. Zhang. “Layout design of satellite module using a modified artificial bee colony algorithm”. In: *Advanced Science Letters* 4(8-9) (2011), pp. 3178–3181.

[SSS19] N. Shiono, H. Suzuki, and Y. Saruwatari. “A dynamic programming approach for the pipe network layout problem”. In: *European journal of operational research* 277(1) (2019), pp. 52–61.

[Sho+03] N.Z. Shor et al. “P.I. Algorithms of Nondifferentiable Optimization: Development and Application”. In: *Cybernetics and System Analysis* 39 (2003), pp. 537–548.

[Sia16] P. Siarry. “Metaheuristics”. In: *Berlin/Heidelberg, Germany: Springer.* 71 (2016).

[Sil+19] D. Silberhorn et al. “Assessment of hydrogen fuel tank integration at aircraft level”. In: *In Deutscher Luft-und Raumfahrtkongress* (2019).

[SBL11] E.K. Da Silva, H.J.C. Barbosa, and A.C.C. Lemonge. “An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization”. In: *Optimization and Engineering* 12(1-2) (2011), pp. 31–54.

[SS06] S.P. Singh and R.R. Sharma. “A review of different approaches to the facility layout problems”. In: *International Journal of Advanced Manufacturing Technology* 30 (2006).

[SMD17] A. Sinha, P. Malo, and K. Deb. "A review on bilevel optimization: From classical to evolutionary approaches and applications". In: *IEEE Transactions on Evolutionary Computation* 22(2) (2017), pp. 276–295.

[Sin+13] A. Sinha et al. "Multi-objective stackelberg game between a regulating authority and a mining company: a case study in environmental economics". In: *In: 2013 IEEE congress on evolutionary computation (CEC). IEEE.* (2013), pp. 478–485.

[Sob01] I.M. Sobol. "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates". In: *Mathematics and computers in simulation* 55(1-3) (2001), pp. 271–280.

[Som+19] R. Soman et al. "A study of sensor placement optimization problem for guided wave-based damage detection". In: *Sensors* 19(8) (2019), p. 1856.

[Spa+15] E. Sparks et al. "Automating model search for large scale machine learning". In: *Balazinska, M. (ed.) Proceedings of the Sixth ACM Symposium on Cloud Computing - SoCC '15* (2015), pp. 368–380.

[SJ91] W.M. Spears and K.A. De Jong. "An analysis of multi-point crossover". In: *In Foundations of genetic algorithms* 1 (1991), pp. 301–315.

[Sri+09] N. Srinivas et al. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *arXiv preprint arXiv:0912.3995* (2009).

[SS17] S. Srivastava and S.K. Sahana. "Nested hybrid evolutionary model for traffic signal optimization". In: *Applied Intelligence* 46(1) (2017), pp. 113–123.

[Sta52] H. Von Stackelberg. "The theory of the market economy". In: *Oxford University Press, Oxford* (1952).

[Sta+13] S. Stadler et al. "Conceptual full-vehicle development supported by integrated computer-aided design methods". In: *Computer-Aided Design and Applications* 10(1) (2013), pp. 159–172.

[SNB98] M. Stelmack, N. Nakashima, and S. Batill. "Genetic algorithms for mixed discrete/continuous optimization in multidisciplinary design". In: *In: 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization. Reston, Virigina: American Institute of Aeronautics and Astronautic* (1998).

[Ste17] P.I. Stetsyuk. "Shor's r-Algorithms: Theory and practice". In: *In Optimization Methods and Applications: In Honor of Ivan V. Sergienko's 80th Birthday; Butenko, S., Pardalos, P.M., Shylo, V.* Springer: Berlin/Heidelberg, Germany (2017), pp. 495–520.

[SP97] R. Storn and K. Price. "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11(4) (1997), p. 341.

[SPM12] Q. Sun, B. Pfahringer, and M. Mayo. “Full model selection in the space of data mining operators”. In: *In Proceedings of the 14th annual conference companion on genetic and evolutionary computation* (2012), pp. 1503–1504.

[SCL22] S. Sun, H. Chen, and J. Lin. “A Universal Method for Modeling and Characterizing Non-Circular Packing Systems Based on n-Point Correlation Functions”. In: *Materials* 15(17) (2022), p. 5991.

[ST03] Z.G. Sun and H.F. Teng. “Optimal layout design of a satellite module”. In: *Engineering Optimization* 35(5) (2003), pp. 513–529.

[SR97] P.D. Surry and N.J. Radcliffe. “The COMOGA method: constrained optimisation by multi-objective genetic algorithms”. In: *Control and Cybernetics* 26 (1997), pp. 391–412.

[Swe+14] K. Swersky et al. “Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces”. In: *arXiv preprint arXiv:1409.4011* (2014).

[Swi+14] L.P. Swiler et al. “Surrogate models for mixed discrete-continuous variables”. In: *Constraint Programming and Decision Making* (2014), pp. 181–202.

[TTi+16] T.Tian et al. “The multiple container loading problem with preference”. In: *European Journal of Operational Research* 248(1) (2016), pp. 84–94.

[Tal09] E.G. Talbi. “Metaheuristics: from design to implementation”. In: *John Wiley & Sons* (2009).

[Tam92a] K.Y. Tam. “A simulated annealing algorithm for allocating space to manufacturing cells”. In: *The International Journal of Production Research* 30(1) (1992), pp. 63–87.

[Tam92b] K.Y. Tam. “Genetic algorithms, function optimization, and facility layout design”. In: *European Journal of Operations Research* 63 (1992), pp. 322–346.

[TT99] F. Tang and H. Teng. “A modified genetic algorithm and its application to layout optimization”. In: *Journal of Software* 10(10) (1999), pp. 1096–1102.

[TS95] D.M. Tate and A.E. Smith. “A genetic approach to the quadratic assignment problem”. In: *Computers & Operations Research* 22 (1995), pp. 73–83.

[Ten+09] H.F. Teng et al. “A dual-system variable-grain cooperative coevolutionary algorithm: satellite-module layout design”. In: *IEEE transactions on evolutionary computation* 14(3) (2009), pp. 438–455.

[TZ19] S. Thevenin and N. Zufferey. “Learning variable neighborhood search for a scheduling problem with time windows and rejections”. In: *Discrete Applied Mathematics* 261 (2019), pp. 344–353.

[Tho+21] R.A. Thokar et al. “Multiobjective nested optimization framework for simultaneous integration of multiple photovoltaic and battery energy storage systems in distribution networks”. In: *Journal of Energy Storage* 35 (2021), p. 102263.

[Tho+13] C. Thornton et al. “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms”. In: *In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 847–855.

[Tia+23] S. Tian et al. “A Variable Neighborhood Search Algorithm for Heat Pipe-Constrained Component Layout Optimization”. In: *In 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (2023), pp. 1452–1457.

[TR76] J.A. Tompkins and R.J. Reed. “An applied model for the facilities design problem”. In: *International Journal of Production Research* 14(5) (1976), pp. 583–595.

[TMX09] D. Tong, A. Murray, and N. Xiao. “Heuristics in spatial analysis: a genetic algorithm for coverage maximization”. In: *Annals of the Association of American Geographers* 99(4) (2009), pp. 698–711.

[THU20] V. Tongur, M. Hacibeyoglu, and E. Ulker. “Solving a big-scaled hospital facility layout problem with meta-heuristics algorithms”. In: *Engineering Science and Technology* 23(4) (2020), pp. 951–959.

[Tor97] V. Torczin. “On the convergence of pattern search algorithms”. In: *SIAM Journal on Optimization* 11(3) (1997), pp. 1–25.

[TXZ18] B. Tran, B. Xue, and M. Zhang. “Variable-length particle swarm optimization for feature selection on high-dimensional classification”. In: *IEEE Transactions on Evolutionary Computation* 23(3) (2018), pp. 473–487.

[Tra+22] T.T. Tran et al. “Design of experiments for mixed continuous and discrete variables”. In: (2022).

[TDT15] P.K. Tripathy, R.K. Dash, and C.R. Tripathy. “A dynamic programming approach for layout optimization of interconnection networks”. In: *Engineering Science and Technology, an International Journal* 18(3) (2015), pp. 374–384.

[TBT96] K. Tsuchiya, S. Bhariktar, and Y. Takefuji. “A neural network approach to facility layout problems”. In: *European Journal of Operational Research* 89(3) (1996), pp. 556–5563.

[VTM00] P. Vadakkepat, K.C. Tan, and W. Ming-Liang. “Evolutionary artificial potential fields and their application in real time robot path planning”. In: *In Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512). IEEE.* 1 (2000), pp. 256–263.

[Var+18] K. Varelas et al. “A comparative study of large-scale variants of CMA-ES”. In: *In Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings.Springer International Publishing.* Part I 15 (2018), pp. 3–15.

[Vas+20] D. Vashisht et al. “Placement integrated circuits using cyclic reinforcement learning and simulated annealing”. In: *arXiv preprintarXiv:2011.07577* (2020).

[VC94] L.N. Vicente and P.H. Calamai. “Bilevel and multilevel programming: A bibliography review”. In: *Journal of Global optimization* 5(3) (1994), pp. 291–306.

[VM21] A.H. Victoria and G. Maragatham. “Automatic tuning of hyperparameters using Bayesian optimization”. In: *Evolving Systems* 12 (2021), pp. 217–223.

[Vol68] T.E. Vollmann. “A computerized model for office layout”. In: *Journal of Industrial Engineering* 19(7) (1968), pp. 321–329.

[VB66] T.E. Vollmann and E.S. Buffa. “The facilities layout problem in perspective”. In: *Management Science* 12(10) (1966), pp. 450–468.

[WC87] G. Wa and P. Chamoni. “MICROLAY: an interactive computer program for factory layout planning on microcomputers”. In: *European Journal of Operational Research* 31(2) (1987), pp. 185–193.

[Wag+11] M. Wagner et al. “Optimizing the layout of 1000 wind turbines”. In: *European wind energy association annual event* 205209 (2011).

[WZZ21] F. Wang, H. Zhang, and A. Zhou. “A particle swarm optimization algorithm for mixed-variable optimization problems”. In: *Swarm and Evolutionary Computation* 60 (2021), p. 100808.

[WA19] H. Wang and B. Alidaee. “The multi-floor cross-dock door assignment problem: Rising challenges for the new trend in logistics industry”. In: *Transportation Research Part E: Logistics and Transportation Review* 132 (2019), pp. 30–47.

[Wan+02] H. Wang et al. “An improved algorithm for the packing of unequal circles within a larger containing circle”. In: *European Journal of Operational Research* 141(2) (2002), pp. 440–453.

[Wan+19a] N. Wang et al. “Analysis and design of typical automated container terminals layout considering carbon emissions”. In: *Sustainability* 11(10) (2019), p. 2957.

[WLW22] Y. Wang, Z. Liu, and H. Wang. “Proposal and layout optimization of a wind-wave hybrid energy system using GPU-accelerated differential evolution algorithm”. In: *Energy* 239 (2022), p. 121850.

[Wan+17] Y. Wang et al. “Differential evolution with a new encoding mechanism for optimizing wind farm layout”. In: *IEEE Transactions on Industrial Informatics* 14(3) (2017), pp. 1040–1054.

[Wan+19b] Y. Wang et al. “A stimulus-response-based allocation method for the circle packing problem with equilibrium constraints”. In: *Physica A: Statistical Mechanics and its Applications* 522 (2019), pp. 232–247.

[WST09] Y.S. Wang, Y.J. Shi, and H.F. Teng. “An improved scatter search for circles packing problem with the equilibrium constraint”. In: *Chinese Journal of Computers* 32(6) (2009), pp. 1214–1221.

[WTS09a] Y.S. Wang, H.F. Teng, and Y.J. Shi. “Cooperative co-evolutionary scatter search for satellite module layout design”. In: *Engineering Computations* 26(7) (2009), pp. 761–785.

[WTS09b] Y.S. Wang, H.F. Teng, and Y.J. Shi. “Cooperative co-evolutionary scatter search for satellite module layout design.” In: *Engineering Computations* 26(7) (2009), pp. 761–785.

[War08] E.J. Ward. “A review and comparison of four commonly used Bayesian and maximum likelihood model selection tools”. In: *Ecological Modelling* 211(1-2) (2008), pp. 1–10.

[WW84] R.E Webb and P.M. Wolfe. “CREATE : Interactive Model of the Building Design Process”. In: *Proceedings, Annual Industrial Engineering Conference* (1984).

[Wei+10] X. Wei et al. “A new method for the design of the heliostat field layout for solar tower power plant”. In: *Renewable Energy* 35(9) (2010), pp. 1970–1975.

[WHD18] J. Wilson, F. Hutter, and M. Deisenroth. “Maximizing acquisition functions for Bayesian optimization”. In: *Advances in neural information processing systems* 31 (2018).

[Wu+19] J. Wu et al. “Hyperparameter optimization for machine learning models based on Bayesian optimization”. In: *Journal of Electronic Science and Technology* 17(1) (2019), pp. 26–40.

[XXA07a] R.-B. Xiao, Y.-C. Xu, and M. Amos. “Two hybrid compaction algorithms for the layout optimization problem”. In: *BioSystems* 90(2) (2007), pp. 560–567.

[XXA07b] R.B. Xiao, Y.C. Xu, and M. Amos. “Two hybrid compaction algorithms for the layout optimization problem”. In: *BioSystems* 90(2) (2007), pp. 560–567.

[Xie+19] J. Xie et al. “A sensor deployment approach using improved virtual force algorithm based on area intensity for multisensor networks”. In: *Mathematical Problems in Engineering* (2019).

[XS08] W. Xie and N.V. Sahinidis. “A branch-and-bound algorithm for the continuous facility layout problem”. In: *Computers & Chemical Engineering* 32(4-5) (2008), pp. 1016–1028.

[Xu+10] Y C Xu et al. “Ant colony algorithm for the weighted item layout optimization problem”. In: *arXiv preprint:1001.4099* (2010).

[XXA07c] Y.C. Xu, R.B. Xiao, and M. Amos. “A novel genetic algorithm for the layout optimization problem.” In: *In 2007 IEEE Congress on Evolutionary Computation* (2007), pp. 3983–3943.

[XZT19] Z.Z. Xu, C.Q. Zhong, and H.F. Teng. “Assignment and layout integration optimization for simplified satellite re-entry module component layout”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233(12) (2019), pp. 4287–4301.

[YAJ20] D. Yi, J. Ahn, and S. Ji. “An Effective Optimization Method for Machine Learning Based on ADAM”. In: *Applied Sciences* 10(3) (2020), p. 1073.

[Yin+22] R. Yin et al. “BO-B&B: A hybrid algorithm based on Bayesian optimization and branch-and-bound for discrete network design problems”. In: *Electronic Research Archive* 30(11) (2022), pp. 3993–4014.

[YH09] W. Yishou and T. Hongfei. “Knowledge fusion design method: satellite module layout”. In: *Chinese Journal of Aeronautics* 22(1) (2009), pp. 32–42.

[YK13] Y. Yoon and Y.H. Kim. “An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks”. In: *IEEE Transactions on Cybernetics* 43(5) (2013), pp. 1473–1483.

[YYL21] J. Yu, X. You, and S. Liu. “A heterogeneous guided ant colony algorithm based on space explosion and long–short memory”. In: *Applied Soft Computing* 113 (2021), p. 107991.

[Zae18] M. Zaefferer. “Surrogate models for discrete optimization problems”. Dortmund University, 2018.

[ZH18a] M. Zaefferer and D. Horn. “A First Analysis of Kernels for Kriging-Based Optimization in Hierarchical Search Spaces”. In: *In Parallel Problem Solving from Nature – PPSN XV: 15th International Conference. Springer.* 11102 (2018), pp. 399–410.

[ZH18b] M. Zaefferer and D. Horn. “A first analysis of kernels for kriging-based optimization in hierarchical search spaces”. In: *In Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part II 15* (2018), pp. 8–12.

[Zen+16] Z.Z. Zeng et al. “Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container”. In: *European Journal of Operational Research* 250(2) (2016), pp. 616–627.

[ZTS08] B. Zhang, H.F. Teng, and Y.J. Shi. “Layout optimization of satellite module using soft computing techniques”. In: *Applied Soft Computing* 8(1) (2008), pp. 507–521.

[ZW93] C. Zhang and H.P. Wang. “Mixed-discrete nonlinear optimization with simulated annealing”. In: *Engineering Optimization* 21(4) (1993), pp. 277–291.

[Zha+22a] J. Zhang et al. "Component assignment and layout optimization for multi-module microsatellite considering variable module size". In: *Acta Astronautica* 198 (2022), pp. 36–44.

[Zha+19] Y. Zhang et al. "A latent variable approach to Gaussian process modeling with qualitative and quantitative factors". In: *Technometrics* (2019), pp. 1–12.

[Zha+20] Y. Zhang et al. "A latent variable approach to Gaussian process modeling with qualitative and quantitative factors". In: *Technometrics* 62(3) (2020), pp. 291–302.

[Zha+23] Y. Zhang et al. "Multi-Robot Coordination and Layout Design for Automated Warehousing". In: *arXiv preprint arXiv:2305.06436* (2023).

[ZWT13] Z.H. Zhang, Y.S. Wang, and H.F. Teng. "Parallel dual-system cooperative co-evolutionary differential evolution algorithm with human-computer cooperation for multi-cabin satellite layout optimization". In: *Journal of Convergence Information Technology* 8(4) (2013), pp. 711–720.

[ZW19] L. Zhao and J. Wei. "A nested particle swarm algorithm based on sphere mutation to solve bi-level optimization". In: *Soft Computing* 23(21) (2019), pp. 11331–11341.

[ZL18] M. Zhao and J. Li. "Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators". In: *In 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI). IEEE.* (2018), pp. 613–618.

[Zha+22b] Q. Zhao et al. "Coverage optimization of wireless sensor networks using combinations of PSO and chaos optimization". In: *Electronics* 11(6) (2022), p. 853.

[Zha+17] Y. Zhao et al. "Optimal layout design of obstacles for panic evacuation using differential evolution". In: *Physica A: Statistical Mechanics and its Applications* 465 (2017), pp. 175–194.

[ZL17] Q. Zheng and Y.G. Liang. "A study on the particle swarm optimization with adaptive weight constrained layout optimization". In: *In 2016 8th international conference on intelligent human-machine systems and cybernetics (IHMSC). IEEE.* 1 (2017), pp. 283–287.

[ZXT19] C.Q. Zhong, Z.Z. Xu, and H.F. Teng. "Multi-module satellite component assignment and layout optimization". In: *Applied Soft Computing* 75 (2019), pp. 148–161.

[ZGG05] C. Zhou, L. Gao, and H.B. Gao. "Particle swarm optimization based algorithm for constrained layout optimization". In: *Control Decision* 20(1) (2005), pp. 36–40.

[Zho+17] J. Zhou et al. "An exact penalty function method for optimising QAP formulation in facility layout problem". In: *International Journal of Production Research* 55(10) (2017), pp. 2913–2929.

[ZQZ11] Q. Zhou, P.Z.G. Qian, and S. Zhou. "A Simple Approach to Emulation for Computer Models With Qualitative and Quantitative Factors". In: *Technometrics* 53(3) (2011), pp. 266–273.

[ZLK03] Y. Zhou, Y. Li, and L. Kang. "Multi-objective and MGG evolutionary algorithm for constrained optimization". In: *In The 2003 Congress on Evolutionary Computation, CEC'03.IEEE.* 1 (2003), pp. 1–5.

[ZYW06] X. Zhu, Q. Yu, and X. Wang. "A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints". In: *In Proceedings of the 5th IEEE International Conference Cognitive Informatics (ICCI), Beijing, China* 1 (2006), pp. 126–131.

[ZC03] Y. Zou and K. Chakrabarty. "Sensor deployment and target localization based on virtual forces". In: *In IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)* 2 (2003), pp. 1293–1303.

[ZC04] Y. Zou and K. Chakrabarty. "Sensor deployment and target localization in distributed sensor networks". In: *ACM Transactions on Embedded Computing Systems (TECS)* 3(1) (2004), pp. 61–91.

[Zuo+10] Z. Zuo et al. "A study of both optimal locations and toll levels road pricing using genetic algorithm". In: *Journal of the Eastern Asia Society for Transportation Studies* 8 (2010), pp. 145–156.