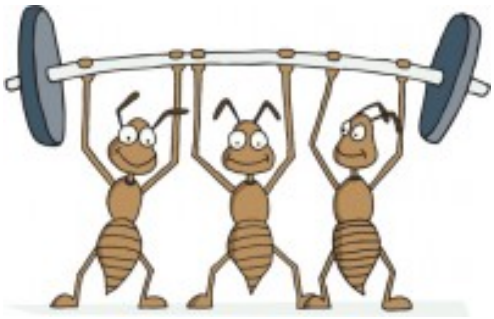
CONTRIBUTIONS TO SINGLE- AND MULTI-OBJECTIVE OPTIMIZATION

TOWARDS DISTRIBUTED AND AUTONOMOUS MASSIVE OPTIMIZATION

Habilitation à Diriger des Recherches
Spécialité : Informatique

BILEL DERBEL



CONTRIBUTIONS TO SINGLE- AND MULTI-OBJECTIVE OPTIMIZATION: TOWARDS DISTRIBUTED AND AUTONOMOUS MASSIVE OPTIMIZATION

Univ. Lille, France
http://www.cristal.univ-lille.fr/~derbel

Habilitation à Diriger des Recherches (HDR)
Defended **December 11, 2017**, Lille, France

Evaluation committee - jury:
- Dr. Hernan E. AGUIRRE, HDR, Shinshu Univ., Japan, *jury member*
- Prof. Pascal BOUVRY, Univ. of Luxembourg, Luxembourg, *reviewer*
- Prof. José A. LOZANO, Univ. of the Basque Country, Spain, *reviewer*
- Prof. Nouredine MELAB, Univ. of Lille, France, *mentor*
- Prof. Frédéric SAUBION, Univ. of Angers, France, *reviewer*
- Prof. Pierre SENS, Univ. of Paris 6 (UPMC, Pierre and Marie Curie), France, *jury member*
- Prof. Lionel SENTURIER, Univ. of Lille, France, *jury chair*

Bilel Derbel: *Contributions to single- and multi- objective optimization,* Towards distributed and autonomous massive optimization, © Defended **December 11, 2017**, Lille, France.

PAGE WEB:
http://www.cristal.univ-lille.fr/~derbel

COURRIEL:
bilel.derbel@univ-lille1.fr

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to the reviewers and the members of the jury who accepted to read this document and who attended the defense. In deed, this habilitation would not have been possible without the expert feedback and the insightful comments of Hernan Aguirre, Jose Lozano, Nouredine Melab, Frédéric Saubion, Pierre Sens and Lionel Senturier.

This habilitation was a unique opportunity to provide a unified overview of my research trajectory during the last few years. In this respect, I would like to warmly thank all my colleagues in the DOLPHIN research group in Lille; in particular, Nouredine Melab and El-Ghazali Talbi for their kind support. Thanks so much to the Master, PhD students and the post-docs I had the chance to collaborate with; especially, Trong-Tuan Vu and Christopher Jankee, for their hard work and the pleasurable experience we had together. Besides, I would like to thank all my colleagues in the computer science department of the University of Lille, in the CRIStAL research center, and in Inria. They highly contributed making my research and teaching activities very much enjoyable.

I am very much grateful to all the colleagues I had the chance and the honor to collaborate with; in particular, Kiyoshi Tanaka, Hernan Aguirre, Cyril Fonlupt and Qingfu Zhang for their friendship and their strong commitment to the development of our common scientific ideas and research collaborations. There is in fact evidence that, due to their continuous support, the next coming years will be again very intensive in research collaborations and in scientific exchanges between France, Japan, Hong Kong, and beyond.

I am also so much grateful to my friends Arnaud Liefooghe and Sébastien Verel. Without their big support and their precious advice, it would not be possible to conduct most of the research described in this document.

Last but not least, I would like to thank all my family. To my dear wife Yosra and my little diamond Ela - thanks so much for your patience and your love.

CONTENTS

1 INTRODUCTION

This document aims at providing an overview of my scientific activities and contributions in the field of single- and multi- objective optimization in general, with a particular interest in distributed and parallel optimization, and general-purpose search algorithms.

Before going into a more focused description of the research topics addressed in this document, I would like to first share with the reader some milestones in my research developments showing how my (re)search trajectory has been evolving from theoretical distributed algorithms to optimization. It is also the opportunity for me to highlight very briefly some of my research contributions in the theory of distributed algorithms (which is otherwise not discussed elsewhere in this document), and to give a short overview of my initial scientific background.

1.1 FROM THEORETICAL DISTRIBUTED ALGORITHMS TO OPTIMIZATION

FROM SOUTH TO NORTH. During my PhD thesis at the University of Bordeaux (LaBRI) from 2002 to 2006, and later as an assistant professor (ATER) at the University of Aix-Marseille 1 in 2006-2007, I was mainly working on the theoretical aspects of distributed and graph algorithms. After I moved to the University of Lille in September 2007 as an associate professor (Maître de Conférences) within the DOLPHIN (now BONUS) research group of the CRIStAL (previously LIFL) Laboratory and Inria, I continued working on those aspects and in the same time started to progressively work in the optimization field, as discussed in the following paragraphs.

LOCALITY IN DISTRIBUTED ALGORITHMS. I was, and to some extent, I am still, interested in the local aspects of distributed computing [Pelooa]. In a distributed or decentralized system, a number compute entities interact in order to compute some task. Interaction among compute entities is in general enabled through some kind of communication and a fully distributed algorithm is a protocol to be executed locally by each compute entities. A compute entity typically has a very local view of the global state of the distributed system, since only a snapshot of the state of the other entities with whom it is interacting directly is available at some moment of the system time-life. Thus, each distributed compute entity has to make decisions based on this local view, hopefully leading to a correct and efficient evolution of the global state of the whole distributed system. In this context, my initial background is on the (time) complexity analysis of distributed graph algorithms from a very theoretical perspective and relating directly to the locality of the problem to be solved.

THEORETICAL DISTRIBUTED GRAPH ALGORITHMS. Let us consider a standard graph-based message passing distributed model, where the nodes of the graph model the distributed entities, and the edges of the graph model the fact that two nodes can communicate together by sending and receiving messages. Let us also assume a very simple setting where the system can operate in rounds, where each round costs one time unit, and a node can send and receive a message to each of his neighbors in this time unit. It is not difficult to see that if a distributed algorithm has a time complexity $O(\tau)$, that is every node terminates after at most $O(\tau)$ time units/rounds, then all the (topological) information used by every node to compute the intended task lies exactly in the ball of radius $O(\tau)$. When it comes

to compute distributively some graph structure, like a spanning tree or a coloring, this means that if initially every node knows the *local* topology of the graph up to distance $O(\tau)$, then it can compute the intended task in one round. Of course, this reasoning holds in a very theoretical model, known as the "Linial's free model" or the "Local model" [Lin92; Peloob]. This kind of theoretical model allows one to study a fundamental question which is tightly related to the (time and message) complexity of distributed algorithms: "what local information does every node need in order to compute some task?" or equivalently "what is the *locality* degree of a distributed task?". In this context, during the very first years of my carrier as an associate professor, I continued working on different theoretical models (e.g., [DMZ10; Der+09; DG08; Der+08; Der08; DMG08]); where I was mainly interested in computing distributively in a fully decentralized manner a number of graph structures appearing at the bottleneck of a number of networking and telecommunication applications, and are often considered in graph theory as core problems. These aspects are not addressed in this document. They are only discussed very briefly in the following two paragraphs highlighting how my scientific interests shifted to the optimization field.

DISTRIBUTED GRAPH SPANNERS. I was firstly interested in the locality of computing the so-called graph spanners. More formally, an (α, β)-spanner of a graph G is a subgraph H that approximates distances in G within a multiplicative factor α and an additive error β, ensuring that for any two nodes u, v, $d_H(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta$. For this kind of structures, we are interested in optimizing another criteria: the weight (or the number of edges) of the subgraph H, i.e., not only preserve distance in the skeleton H, but make it as sparse as possible. It is well-known for instance that every n-node graph has a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges, k is an integer parameter, which can be obtained by a modification of the Kruskal's minimum spanning tree algorithm [Alt+93]. This is also related to the Erdös-Simonovits Girth Conjecture [Erd64; ES82], where it is believed that every (α, β)-spanner with $\alpha + \beta < 2k + 1$ must have $\Omega(n^{1+1/k})$ edges for some worst-case graphs. During my PhD, I was mainly concerned with spanners having a purely multiplicative stretch factor ($\beta \to 0$), and continued studying purely additive spanners ($\alpha \to 0$) as an associate professor. For instance, we were able to derive new lower and upper bounds for the distributed construction of nearly additive graph spanners [Der+09]; where we provide a generic and deterministic distributed algorithm that in constant number of rounds constructs, for every n-node graph and integer $k \geqslant 1$, an (α, β)-spanner of $O(\beta n^{1+1/k})$ edges, where α and β are constants depending on k, e.g., a $(1+\varepsilon, O(1/\varepsilon)^{k-2})$-spanner of $O(\varepsilon^{-k+1} n^{1+1/k})$ edges for any $\varepsilon \in (0, 4]$. Such a result is actually based on some pre-processing steps involving advanced tools from graph theory such as the distributed computation of sparse clusters and independent/dominated sets.

RANDOMIZED DISTRIBUTED SYMMETRY BREAKING. The previous considerations are purely theoretical in the sense that the runtime analysis is derived with respect to a purely theoretical message passing model which informs about the locality of a distributed task in an ideal scenario. Nonetheless, such a model might not hold in practice and I was interested in a more realistic setting where communications between neighboring nodes are subject to collisions and/or interferences [GW13]. Computing distributed scheduling schemes enabling collision-free message transmission is hence mandatory for such a distributed setting, e.g., in multi-hop radio networks, wireless ad-hoc networks, etc. From an optimization perspective, such a scheduling can 'simply' be viewed as a graph problem. From a distributed perspective, this turns out to be a symmetry breaking problem where the major challenge is to avoid that two nodes transmit simultaneously and given that nothing about the surrounding and/or global distributed environment is available locally for each node. In this context, I was interested in the distributed self-organization of nodes using two standard techniques for distributed symmetry breaking in graphs: randomized algorithms and graph-based structures such as node coloring, independent sets and edge matchings,

and contributed some theoretical and applied results [Jem+15; Jem+13; Ghr+13; DT10a; DT10b]. For instance, in [DT10a], we are able to prove that under the harsh SINR (Signal-to-Interference-plus-Noise Ratio) physical model, and for every n-node unit disk graph with maximum degree Δ, there exists a distributed algorithm computing with high probability and in at most $O(\Delta \log n)$ time slots a $(1, O(\Delta))$-coloring, that is a coloring where nodes at distance 1 have different colors and at most $O(\Delta)$ different colors are used overall. The time bound is actually optimal up to a logarithmic factor and is proved by combining graph properties with conventional 'theory' and analysis tools in randomized algorithms.

OPTIMIZATION BEGINS... In parallel to the previously-mentioned research, my scientific interests started shifting progressively to combinatorial optimization and high-performance parallel computing. This was motivated from two scientific perspectives. Firstly, distributed algorithms can be considered as a fundamental building-block in many massively parallel and large-scale compute environments. Tackling large-scale or hard optimization problems often requires to organize the communications between the available processing units in a smart manner. As such, and given the strong involvement of my research team in the first national large-scale experimental grid established in France [Gri], I started looking at the application of different graph based structures, e.g., small world graph, distributed hash-tables, etc, with the aim of efficiently applying them in the design of parallel and fully distributed optimization algorithms. Some fundamental aspects related to termination detection, fault-tolerance, formal verification and complexity analysis, which are not described in this document, were in particular studied within the first PhD Thesis [Dja13] I had the opportunity to co-supervise. Later, designing parallel and distributed optimization algorithms [AT02; Alb05; Tal09] became one of my main scientific activities as discussed all along this document. Secondly, locality in distributed algorithms have, in some sense, a number of common aspects with general-purpose search heuristics. For instance, and specifically with respect to parallel search heuristics, there exist a number of widely-used models and algorithmic frameworks, e.g., the island model, cellular genetic algorithms, ant colony metaheuristics, where local cooperation is shown to play a crucial role. Hence, I start being naturally interested in designing high-level (distributed) search algorithms which would rely on simple local rules that would imply a powerful global behavior of the underlying (distributed) system. Later, designing autonomous, high-level and general-purpose optimization algorithms [HMS12] became one of my favorite research topics, together with the systematic analysis and the fundamental understanding of their behaviors.

1.2 CURRENT RESEARCH TOPICS

Generally speaking, my current research activities are related to a number of interrelated topics including combinatorial single-objective optimization, evolutionary multi-objective optimization, general-purpose and autonomous optimization algorithms, fitness landscape analysis and of course parallel and distributed algorithms. *My main focus is on the design, analysis and understanding of high-level search algorithms, where parallelism and distributed computations play a crucial role*, both to take full benefit from the ever-increasing growth of high performance and large scale compute facilities, and to improve the robustness of the underlying optimization procedures with respect to the ever-increasing complexity of optimization problems and their specific features and application domains. Accordingly, my research can be viewed following four interrelated aspects as summarized in the following paragraphs providing a more focused description of my current scientific interests and objectives. This shall also constitute the main concern of the rest of this document.

LARGE-SCALE PARALLELISM. Optimization algorithms for solving optimization problems are time consuming when applied to difficult, expensive and/or large real-life problem in-

stances. Distributed and massively parallel modern compute environments (see e.g., [Hus+13]) harnessing huge amount of computational resources (e.g., super-computers [Top], grids [Gri], clouds [Ama; Zha+11]) are a key option and a unique opportunity to speed-up the optimization process. Designing efficient and scalable parallel and distributed algorithms for that purpose is a challenging task, not only because of several technological issues such as the hardware characteristics of a target compute platform and the underlying programming paradigms (shared memory, message exchange, virtualization, etc), but also because the optimization flow might itself be difficult to think and to design distributively in parallel. In this context, I am interested in designing fully distributed optimization algorithms, as well as their effective deployment in large-scale and massively parallel compute environments.

LOCALITY AND COOPERATION. Metaheuristics and evolutionary algorithms have been proved to be extremely efficient in solving hard optimization problems [Hol75; Gol89; HS04; Talo9]. Although a large body of the literature is devoted to the parallelization of such algorithms [AT02; Alb05], there still remain a wide field to be explored when it comes to set up novel distributed and parallel search heuristics both at the design level and at the implementation and deployment level. In this context, I am interested in adopting a principled approach where the algorithm is thought and designed in parallel till the beginning eventually fitting a real compute platform. This includes the use of a purely local information when designing the different components of an algorithm and the design of novel local coordination mechanisms that do not rely on any global information but rather on a cooperative information acquired during the optimization process. The objective is two-fold: (i) design effective optimization techniques, and (ii) run them efficiently on large-scale compute environments without much additional design efforts.

ADAPTIVENESS AND LEARNING. The performance of an optimization algorithm depends heavily on the accurate configuration and combination of its several algorithmic components and their respective parameters. Besides, there might be a plethora of approaches that one can adopt to tackle a given optimization problem. An algorithm that is well suited for a particular instance type might fail when executed on another one. It might even be the case that different algorithms have to be combined at different stages of the optimization process in order to achieve optimal performances. This is especially the case when designing general purpose solvers that are intended to tackle problem instances or problem domains having different characteristics. In this respect, I am interested in the design of autonomous optimization techniques [HMS12] in order to improve the robustness of existing optimization methods and to enable the automatic design of novel and high-level search algorithms. This is also related to the on-line and off-line tuning and selection of algorithms, as well as other algorithmic techniques coming from the learning and computational intelligence fields, such as portfolio design, performance prediction, surrogates and meta-models, etc. Generally speaking, being able to adapt the optimization process depending on the problem being considered, the instance one wants to tackle, the current state of the search, the variable setting, the search trajectory, the possible components or parameters, etc, requires learning mechanisms that I am interested in designing, analyzing and understanding.

MULTI-OBJECTIVE OPTIMIZATION. In real-life applications, one has to deal with a number of objective functions to be optimized simultaneously although being by essence pairwise conflicting [Deb01; CLV07]. In contrast to single-objective problems, where the goal is to compute one solution optimizing one cost functions, multi-objective problems imply a whole set of solutions providing the best possible compromises. Finding such a set, or even an approximation of it, is usually a challenging task for which specific solving methodologies and approaches are needed. In this context, I am interested in the development of computationally efficient evolutionary algorithms, with a particular focus on decomposition based techniques [ZL07] that consists in transforming the original problem in a number of

(smaller) sub-problem using some scalarizing function, and solving the so-obtained subproblems cooperatively. This kind of approaches offers in fact a high degree of flexibility, e.g., when leveraging existing single-objective evolutionary algorithms, and enable parallelism in a rather natural manner. Additionally, it is worth-noticing that the research issues discussed in the previous paragraphs still hold in the multi-objective setting, and I also consider to address them while coping with the multi-dimensional nature of the objective space.

1.3 DOCUMENT OUTLINE

The rest of this document provides an overview of some of the research I conducted in collaboration with different colleagues and postdoctoral student, as well as PhD and Master students I had the chance to co-supervise or to work with. For clarity, some contributions are not included in this document. In particular, my early contributions on the locality and theory of distributed algorithms are not presented and some of my recent contributions are also omitted. My goal is in fact to provide the reader a consistent overview of my main current research interests and activities and, it is my hope, a clear idea about the research path I would like to follow in the future.

The rest of the document is organized following a simple classification of the optimization problems and algorithms I am interested in, namely, whether an *exact* or a *heuristic* approach is considered and whether the target optimization problem is *single-* or *multi-* objective. The described contributions are there-by structured into three chapters, each one eventually dealing with one or more particular facets of the research topics mentioned in the previous section.

In Chapter 2, we consider exact (single-objective) optimization algorithms where we are mainly concerned with the design of parallel and scalable high performance load-balancing algorithms to cope with the dynamic and irregular tree search work-load in a heterogeneous compute environment. Our general goal is two-fold: (i) provide a systematic investigation of the design of parallel B&B in a large-scale and massively parallel environment, and (ii) improve and leverage the existing dynamic load-balancing protocols from the High Performance Computing (HPC) community.

In Chapter 3, we consider evolutionary (single-objective) optimization algorithms, where we are mainly concerned with the design and analysis of online distributed adaptive operator selection techniques. Other related contributions in the sequential setting are also highlighted with respect to hyperheuristics and offline algorithm configuration. Our general goal is three-fold: (i) leverage existing operator selection techniques from reinforcement learning in the distributed setting, (ii) enhance our fundamental understanding of the behavior and dynamics of existing techniques in lights of the properties of the considered optimization scenario, and (iii) strengthen the existing methodologies in attempt to establish more automated and autonomous optimization algorithms.

In Chapter 4, we consider (evolutionary) multi-objective optimization algorithms, where we are mainly concerned with the design of sequential and distributed decomposition-based techniques. Other related contributions on the design and incorporation of evolutionary operators based on local search and adaptive (machine learning based) stochastic sampling are also discussed; respectively for discrete and continuous domains. Our general goal is three-fold: (i) reduce the curse of dimensionality by adopting a divide-and-conquer approach allowing to incorporate a high level parallelism, (ii) contribute to the strengthening of the main design components of such an approach and the analysis of their combined effect on the search behavior and dynamics, and (iii) enhance our understanding of the main challenges and bottlenecks for a high quality and computationally efficient multi-objective optimization process.

Besides some standard and well understood optimization paradigms and algorithmic concepts that the reader is assumed to be familiar with, the chapters are self-contained and pair-

wise independent. Each chapter starts with a short abstract summarizing the corresponding contributions, and providing a general outline, together with a summary of related scientific projects and collaborations. It then follows a general introduction providing the necessary background, definitions, related work, etc, and stating the general challenges and the adopted methodology. A description of our contributions, organized in different sections, is then provided and some selected experimental results are discussed. We in fact choose to go briefly into some empirical findings, since our work is essentially based on a systematic and comprehensive empirical analysis. The last section of each chapter provides a conclusion, and a number of general perspectives are discussed as well.

In Chapter 5, we provide a general conclusion where our main research perspectives are discussed from a very general perspective in light of our current and on-going projects. Notice also that an extended CV including our research projects, research and teaching responsibilities, student supervision, personal bibliography etc, is provided in Appendices A and B.

2 DISTRIBUTED AND PARALLEL BRANCH-AND-BOUND

In this chapter, we describe our main contributions to the design of efficient parallel and distributed Branch-and-Bound (B&B) algorithms in both homogeneous and heterogeneous compute environments, where heterogeneity can occur either at the computing-node level or at the communication-link level. This piece of research was mainly conducted in the context of Trong Tuan Vu PhD Thesis [Vu14]; where the focus is on balancing B&B workload accurately and on leveraging state-of-the-art dynamic load-balancing algorithms coming from the High Performance Computing (HPC) literature. The following aspects will be addressed:

- In the introductory section, a general overview of B&B and the different types of compute environments that we shall consider is first given. The design challenges, as well as some related work, are then highlighted. The goal is to provide the reader with a brief overview of some critical aspects to effectively set up a parallel B&B algorithm; while abstracting as much as possible the technical, yet important, implementation and technological issues.

- In Section 2.2, a tree-based load-balancing protocol is described and its efficiency is studied in a large-scale node-homogeneous compute environment. Our goal is to highlight the effectiveness of carefully structuring compute nodes using a distributed tree like architecture, compared to a (centralized) Master/Worker architecture, and more importantly compared to the well-established random work stealing based dynamic load-balancer.

- In Section 2.3, a heterogeneous multi-core multi-CPU multi-GPU approach is then described and its parallel efficiency is demonstrated. To the best of our knowledge, the performance obtained when applying the underlying parallel and distributed protocols for the permutational flowshop problem, constitute the state-of-the-art both in terms of parallel efficiency and distributed scalability.

- In Section 2.4, an adaptive distributed load-balancing protocol based on the work stealing paradigm and dedicated to link-heterogeneous compute environments is described and its behavior is studied using an emulation-based approach. We show in particular that, independently of parallel B&B, our distributed protocol is able to outperform existing algorithms for unbalanced tree search workload dynamic scheduling.

RELATED PUBLICATIONS, PROJECTS AND COLLABORATORS.

COLLABORATORS. A. Ali (Postdoctoral Supervision), A. Bendjoudi, M. Djamai (PhD Supervision), N. Melab, T.T. Vu (PhD Supervision)

PUBLICATIONS. [Dja13; Vu14; VD16; VD14; VDM13; Vu+12]

PROJECTS. HEMERA Inria large wingspan project (Responsible for Challenge A - COPS) (2010-2014), BQR Emergent Research (Coordinator) (2012-2013). (See Appendix A.3)

2.1 INTRODUCTION AND BACKGROUND

2.1.1 B&B in a nutshell

SERIAL B&B. Branch-and-Bound (B&B) is a universal search algorithm [LD10] that can be used to find the optimal solution(s) with respect to a given optimization problem. The general idea of B&B is to represent the search space as a tree, where the root of the tree represents the whole search space, and the intermediate nodes represent smaller sub-problems where typically the range of few variables has been fixed or restricted. Generally speaking, a sequential B&B algorithm uses four operators as illustrated in the high level template of Algorithm 1. The algorithm maintains a list of subproblems which constitutes the nodes of the search tree. At each iteration, one specific subproblem, that is one intermediate node in the search tree, is selected to be processed. Processing a subproblem first consists in computing its cost. If the subproblem is a leaf in the tree, then this means that all the variables of the subproblem have been determined, and a complete solution is obtained. In this case, the quality of the solution is evaluated using the cost function of the original problem to be optimized. The newly computed solution is retained if its quality is better than the best one found so far in previous iterations. Otherwise, the cost of the subproblem is computed using a problem-dependent method. Without loss of generality, this corresponds to the computation of a lower bound in case minimization is considered. If the computed lower-bound is worst than the quality of the best solution found so far (i.e., the best known upper bound), the subproblem and all its potential descendants are discarded and not expanded further in the tree. This is known as the pruning phase of B&B. If the lower-bound does not allow to prune, the branching operation is activated and the current subproblem is further decomposed into two or more smaller subproblems. The newly computed subproblems represent new intermediate node in the tree, with the current subproblem originating them being their parent, and so on until all the search tree is fully explored.

SOURCES OF PARALLELISM. Pruning the B&B nodes can significantly reduce the size of the search space by exploring only those parts that exhibit promising costs. However, B&B is a computing intensive algorithm that requires a relatively huge computational effort especially when dealing with large-scale and difficult problem instances. Parallel and distributed computing is among the classical alternatives that are used in order to speed up the computations of serial B&B. This has been the object of abundant work and a relatively rich literature can be found about the subject. In particular, the sources of parallelism in B&B are now well-identified and one can find several taxonomies and classifications on the subject B&B [Tri86; TB92; GC94; BCG00]. All these classifications share the following simple observation. The problem-dependent bounding operation is many often the most time consuming part of a serial B&B algorithm and much gain can be obtained when parallelizing this step. This type of parallelism is known as low level or node-based. We can distinguish two basic variants. In the first variant, the bounding operation is parallelized when executed for one single subproblem. In the context of our study, this is of limited interest since the bounding operation is problem-dependent. In the second variant, many bounding operations are carried out for different subproblems which constitutes a much more generic and standard source of parallelism in parallel B&B. Tightly related to this source of parallelism, another important type of parallelism in B&B consists in exploring different subproblems in parallel. This is known as high level or tree-based parallelism. It typically consists in exploring different B&B subtrees in parallel, that is distributing the computed subproblems over the available computing processes and performing the serial B&B in parallel. This type of parallelism enables to implicitly perform the bounding operation in parallel when concurrently exploring different subtrees, but it might also imply different tree explorations strategies than the original serial B&B (in the same manner that different branching strategies can impact the search [Cer+17]).

Algorithm 1: A simplified template of serial B&B for minimization problems

```
Input: r : root node (representing the whole problem to be solved);
       f : objective function vector, to be minimized;
       uy* : initial upper bound (e.g., obtained with a heuristic);
Output: x* : optimal solution; y* = f(x*) : optimal objective value;
1  T ⟵ {r} ; // BB active tree
2  while T ≠ ∅ do
       // Select the next node to be explored in the B&B tree (e.g., DFS, BFS)
3      N ⟵ Select(T) ;
4      T ⟵ T \ {N} ;  // node N will now be explored
5      𝒩 := (N_1, ..., N_k) ⟵ Decompose_Branch(N) ;
6      for ℓ ∈ {1, ··· , k} do
           // if N_ℓ is a final node (feasible complete solution)
7          if N_ℓ is a leaf then
8              if f(N_ℓ) ⩽ y* then
9                  x* ⟵ N_ℓ; y* ⟵ f(N_ℓ); uy* ⟵ y*;
10             𝒩 ⟵ 𝒩 \ {N_ℓ} ;
11         else
               // Pruning by infeasibility
12             if X(N_ℓ) is not feasible then
13                 𝒩 ⟵ 𝒩 \ {N_ℓ} ;
14             else
                   // Compute lower bound (problem-dependent)
15                 LB(N_ℓ) ⟵ a lower bound with respect to partial solution N_ℓ ;
                   // Pruning by bounds
16                 if y* < LB(N_ℓ) then
17                     𝒩 ⟵ 𝒩 \ {N_ℓ} ;
       // Updating the tree with the newly created (not pruned) nodes
18     T ⟵ T ∪ 𝒩 ;
```

In practice, computing resources with different characteristics are nowadays increasingly available in the form of public and private clouds, grids, aggregated clusters and personal computers scattered over possibly large-scale distributed platforms connected via a network. This huge number of parallel an distributed resources offers an impressive compute power which is in theory sufficient to tackle large problem instances. On the other hand, exploiting such a compute power when parallelizing B&B is challenging mainly due to the complexity coming from the considered system architectures. In our work, we consider to use computing resources coming from both shared and distributed memory systems. For completeness, we first sketch the characteristics of the computing environments and the underlying compute architecture considered in this work, and then we state the underlying challenges with respect to parallel B&B.

2.1.2 Compute Environments and Paradigms

SHARED MEMORY SYSTEMS. A shared memory system refers to a computing environment where all threads/processors share some common memory space. The interconnection of processors to the main memory defines the particular type of the architecture in shared memory systems (e.g., UMA, NUMA, etc). The most common shared memory systems use one or more multicore processors in which a multicore processor has multiple CPUs or cores

on a single chip. Such systems assume that processors are able to access directly any part of the main memory thanks to a logical address space. From a programming or engineering perspective, writing parallel programs for shared memory systems requires to coordinate the work of the different threads. This is not difficult *per se*; however, achieving parallel efficiency is generally a challenging task. One of the most critical issue is the synchronization of the communications among the threads via some local operations on the shared variables. For instance, in parallel B&B, an idle thread running on a core can perform load balancing to migrate some subproblems from another working thread. The idle thread has to remove the subproblems out of the work pool of the working thread and to write them into its own pool. This can cause data race issues. Synchronization techniques are hence borrowed to transform simultaneous accesses to a sequence of several single access. Obviously, this comes with a price and reduces the potential parallelism of a system. Besides, when dealing with particular types of devices such as GPUs (Graphics Processing Units), memory access is perhaps the most severe issue that might impact parallel performance.

DISTRIBUTED MEMORY SYSTEMS. A distributed memory system indicates that the underlying processing units (PUs) do not share a common memory but are connected through some kind of networks, allowing them to communicate through the message passing paradigm. Although the quality of the distributed interconnect is getting better and better, and allows for a relatively fast delivery of messages, communications in distributed message passing systems are in general much expensive compared to the local communication enabled by shared memory. This constitutes the first challenge when dealing with parallel algorithms in such a compute environment especially if high performance and scalability are the main objectives. In this work, we are interested in large-scale distributed systems where PUs could be geographically distributed and connected through different types of physical networks (i.e Local area networks or Wide area networks). Such systems are generally built as an aggregation of several types of PUs connected by different types of networks; thus, exposing different levels of hardware and network hierarchies, and leading to a complex system and network architecture. This raises several challenging issues for setting up large-scale distributed protocols. *Heterogeneity* is one critically important aspect in nowadays modern platforms [Top]. It is also the main focus of the work described in this chapter. For the parallel B&B algorithms that we are interested-in, we shall distinguish two main types of heterogeneity in accordance with the general classification commonly used in the high performance and distributed computing community [BR10a]: *node-heterogeneity* and *link-heterogeneity*. Node-heterogeneous systems use more than one kind of processors with possibly different compute powers and abilities according to their hardware/memory architectures. For example, a node can be simply a single CPU, multi-core CPU or a complex one with a multi-core CPU equipped with a many-core GPUs of different potential computing capabilities. Link-heterogeneous systems refers to the use of different physical networks connecting the underlying PUs. The PUs can thus communicate with different network speeds and bandwidths: ethernet, infiniband, LANs, WANs. etc.

CENTRALIZED *VS.* DECENTRALIZED COMPUTING. A centralized computing architecture, often termed as Master/Worker (MW), is widely used when setting up parallel and distributed computing systems and algorithms. PUs are split into one (or possibly few) master(s) and several workers. The master is the central point of the system which usually manages all the control and coordination/synchronization operations. The workers are in turn responsible for most of the computations. In parallel B&B, the master usually controls the assignment of subproblems to workers and the workers perform B&B computation like branching, bounding, selecting and pruning. In our experience, we observed that this model can lead to rather efficient protocols (especially in small scales); however, for large scales, the performance drops dramatically mainly because of the bottleneck created at the master level. In contrast, a decentralized computing architecture is intended to leverage

the communication bottleneck that might occur at any PU. It refers to a fully distributed management of PUs and does not count on any centralized master. Generally speaking, the communication between PUs is organized following some deterministic or probabilistic patterns, hence implying some kind of abstract communication topology called overlay. This usually allows one to distinguish between two classes of decentralized architectures; those organized following (i) an unstructured overlay, where no fixed topology is imposed, typically when messages are exchanged on a random basis; and those using (ii) a structured overlay, typically following a tree, a hypercube or a distributed hash table as in peer-to-peer computing. It is worth-noticing that some architectures referred to as Hierarchical Master Worker (HMW) are simply to be viewed as using a decentralized tree-like overlay topology in which some PUs are specialized to play some specific actions, hence offering a kind of compromise between a fully decentralized and a fully centralized architecture.

2.1.3 Parallel and Distributed B&B: The Main Challenges

Abstracting away from the characteristics of the distributed environment and the programming paradigm, a parallel B&B algorithm is essentially a *parallel tree search* algorithm where the tree is constructed *dynamically at runtime* as a consequence of the branching and pruning operations. At every node of the tree, a bound has to be computed before determining whether the tree can be expanded/explored or not. At a first glance, it is straightforward to set up a parallel a B&B algorithm. In our opinion, however, the main point is not on how to parallelize the B&B on a specific platform, but how to achieve high performance and scalability with respect to that specific platform. On the one hand, as the parallel exploration is carried out (either to compute bounds or to explore subtrees), the workload has to be distributed fairly among the available PUs. On the other hand, considering that most target platforms are heterogeneous with possibly different levels of hierarchy and compute ability of the underlying PUs, different tightly coupled issues for a parallel efficient B&B have to be addressed. We summarize them in the following.

MAPPING B&B PARALLELISM. The main types of parallelism exposed in a generic B&B is (i) at the node level to evaluate (bound) several B&B tree nodes (subproblems), and, (ii) at the tree level, to explore different B&B subtrees in parallel (bound, prune and branch) [Tri86; TB92; GC94; BCG00]. However, one still have to decide how to generate B&B work units and to which PUs they should be assigned. For instance, the optimal traversal strategy adopted when expanding new active nodes (e.g., DFS, BFS, etc), and subsequently assigning them to some available PUs can depend on the particular optimization problem at hand. Moreover, since some PUs can be better suited to deal with a particular type of parallel B&B operations, mapping these well-established sources of parallelism in B&B to the underlying hardware can constitute a difficulty. In our work, we are mainly concerned with CPUs or GPUs. In contrast to CPU cores, the GPU cores can suffer from thread divergence due B&B work execution. Hence, we shall simply restrict the GPU side to deal with the B&B node-parallelism, that is the bounding of a number of B&B active nodes. Since the bounding operation is problem-specific, B&B node-parallelism can it-self imply some issues, but which can only be investigated in the context of a particular optimization problem. This is of limited interest in the context of our work since our goal is to gain insights in the design of general-purpose parallel B&B algorithms.

WORKLOAD IRREGULARITY. The dynamic and irregular nature of the B&B tree constitutes a major source of workload unbalance that can dramatically prevent high performance and scalability. Processing the search tree distributively in parallel is trivial, for instance, by iteratively generating and distributing subproblems over the available PUs. However, such a naive approach often fails to achieve high parallel efficiency because (i) the shape of the search tree is unknown in advance, i.e., it is difficult to predict in advance the nodes of the

tree that would generate enough work for subsequent iterations, and (ii) the actual explored subtrees differ in shape and have very unbalanced structures, e.g., the depth and the number of nodes that could be attained when following different branches are highly variable. In this respect, we argue that the main challenge is the design of an accurate load balancing mechanism where B&B workload can unfold *dynamically at runtime*.

PLATFORM HETEROGENEITY. The heterogeneity of the compute environment is the other critically important issue when balancing workload across different PUs. For instance, the compute power of PUs of the same type could be substantially different when considering different clusters in a grid-like platform. The computational capability of different PUs can also be very substantial. According to the optimization problem being tackled, the relative performance can for instance range from few to hundreds orders of magnitude in favor of a modern GPU device compared to the most recent CPU cores. Similarly, the communication cost can vary very substantially depending on the different types and means of the communication medium used by the different PUs. This impacts the time PUs stay idle, and hence, can strongly decrease scalability and parallel efficiency. In brief, besides dealing with the unpredictable B&B workload, addressing the heterogeneity of large-scale compute environments is mandatory for high performance.

DISTRIBUTED GLOBAL OPERATIONS. The cost of the best known feasible solution needs to be maintained in order to prune subproblems efficiently. This information is hence to be shared distributively between all B&B parallel processes. Moreover, we have to deal with termination detection, that is to determine whether the search process is terminated or some PUs are still processing some nodes in the B&B trees, in which case the underlying tasks still have to be shared. Although termination detection is often not as critical as the previously mentioned aspects, and can be managed using standard techniques [Dij87], it appears to have a non-negligible impact if not well embedded in the underlying protocols.

2.1.4 Parallel and Distributed B&B: Literature Overview

The previously discussed challenges are in general at the core of several studies. Given the abundant literature on the subject, it is beyond the scope of this document to go through a detailed discussion of related work. A non exhaustive set of representative approaches are nonetheless discussed in the following.

SHARED AND DISTRIBUTED MEMORY B&B. One can find several studies dealing with shared memory parallel B&B, see e.g., [JS89; SD12; Sil+14; Dro+12; BB10; KJL13a; BDW11; ECG11; OD10; OD12]. In such a setting, work pool(s) management is a common issue which is shown to play an important role. In fact, we can see from the generic template of Algorithm 1 that an iteration of a B&B consists of a procedure to pop a subproblem from a pool, perform a set of operations (branching, bounding and pruning), and then insert one or several generated subproblems into the pool. Work pool management then refers to a specific data structure (e.g. array, list, map, stack, queue, etc) placed in a memory location where PUs access generated subproblems. Obviously, this is tightly related to distributing, and thus balancing, the B&B workload. Simultaneous I/O operations to the same work pool(s) pose several issues. In practice, there are two common strategies. In single pool based algorithms, only one memory location is used to maintain a single global pool shared among PUs. Synchronization techniques are then unavoidable when popping/pushing a B&B node from/to the single global pool. For instance, a single pool approach is presented in [MMT13], where the authors reported a relatively big gap between their implementation and the ideal linear speed up. In multiple pool based algorithms, a set of different memory locations are used to handle B&B work units. There are some variants depending on the number of pools used in the system, namely collegial, grouped and mixed. In the first case,

each PU has its own pool. In the second one, all PUs are partitioned in several groups and each group shares the same work pool. The choice of the number of pools depends on the number of PUs as well as their accessing frequency. The last one is a mixed between collegial and grouped.

In this context, we can cite the work of Casado et al. [Cas+08] who proposed two multi-threaded schemes for parallel B&B. In the first scheme, all threads share a global pool of generated subproblems therefore a synchronization mechanism is designed in a master-worker style. In the second scheme, each thread manages a local pool to avoid synchronization overheads and a dynamic load balancing is proposed to deal with the B&B irregularity. At each iteration, if a certain condition is satisfied, a thread creates a new one and migrates work from its local pool to the pool of the new one. The condition for new thread creation is described as follows: the number of running threads are less than the total number of available cores and there is more than one subproblem in the local pool of the thread. In the same spirit, Evtushenko et al [EPS09] presented a B&B solver that allows to deal with both shared and distributed memory environments. At the shared memory level, a global pool and one local pool per compute thread are implemented. When a thread becomes idle it picks a subproblem from the shared pool, or it stays blocked unit the shared pool is fulfilled. After a number of steps, a thread migrates some problems from its local pool to the shared pool which stands for load balancing. At the distributed memory level, they use a master-worker like paradigm to coordinate the distributed threads. Similar load-balancing considerations are addressed in [Her+13b; Her+13a], where a thread can create a new one if the maximum number of threads is not reached. Barreto *et al.* [BB10] conducted a comparison of parallel B&B approaches on shared and distributed memory systems using respectively OpenMP and MPI. A good speedup of both implementations compared to the sequential one is reported; however, the speedup using MPI was found to be slightly better than with OpenMP. This might appear counter-intuitive at the first sigh since it is in fact well understood that inter-communication is much costly in MPI where exchanged messages have to pass through standard networks connecting computers. However, this result also enlightens the negative impact of using synchronization mechanisms in shared memory systems. Most of previously mentioned approaches and more recent ones, e.g., [Mez+14; Ler+14; Sil+14; SD12], try to address the (shared and distributed) memory hierarchy mapping and the pool management issues with respect to B&B workload. Despite their skillful design, previous approaches suffer scalability issues when considering large distributed environments.

CENTRALIZED MASTER-WORKER (MW) B&B. In (message-passing) distributed environments, the classical Master-Worker (MW) paradigm is widely used for parallel B&B. Most often, the distribution and scheduling of B&B work units run only on the master which tries to maintain a faithful global view of work being processed. Among others [Gou+00; GLY00; CF01], we can cite the B&B@GRID approach described in [MMT07b; Mez07], where an interval representation of B&B work units for the permutation-like problems is investigated in attempt to reduce the communication latency and the cost of synchronizing and updating the workers. Otten *et al.* [OD10; OD12] introduced a scheme to predict the complexity of a subproblem so that subproblems can be evenly distributed among workers based on the prediction scheme. Initially, the master node explores the tree up to a parallelization frontier for having enough subproblems, then send them to its workers and wait for the result. Balancing the B&B workload is then ensured by the prediction scheme which estimates the size of the explored space of a subproblem (i.e. the number of explored nodes to solve a subproblem). In [KJL10] a Master-Worker algorithm called GAUUB is proposed, and then improved by proposing another version called GALB [KJL12; KJL13b] that reduces load unbalancing among the available processors. The algorithm GALB is composed of two steps: initialization and distribution. The master executes the initialization step by performing the sequential B&B algorithm until reaching a fixed level L of the search tree in order to generate a large amount of work to distribute among the slaves processors and therefore to

ensure load balancing for all processors in the second distribution step. Generally speaking, it is well understood that MW-based approaches are only suitable at small or intermediate scales, which is also confirmed by our own investigations. Some hierarchical master-worker (HMW), as well as fully decentralized, schemes are proposed to solve the scalability of MW.

HIERARCHICAL MASTER-WORKER (HMW) AND DECENTRALIZED B&B. Despite the many variations that one could find, the key idea of HMW B&B is simply to use several masters organized hierarchically following a tree overlay. As such, there are in general two types of layers that are considered: a control layer comprising one or more levels of masters and a work layer composing workers. For example, Aida *et al.* [AF02; AO05] proposed a three-tier tree structure comprising a supervisor, masters and workers. The supervisor is the root node of the tree and manages all masters. Workers are grouped into sets and each set is managed by a master. The supervisor and the masters are in charge of all communications among workers belonging to different sets such as load balancing, and broadcasting upper bound values. Similarly, Xu *et al.* [Xu+05] proposed a framework to implement parallel search algorithms called APLS. The framework uses the following entities: Master, Hub and Workers. A hub controls a fixed number of workers and the number of hubs increases proportionally with the number of workers. Load balancing is also taken into account at two levels: intra-cluster and inter-cluster. In the first level, the hub manages dynamic load balancing and the workers periodically update their workload to the hub, hence helping the hub to detect and to correct an unbalanced situation. This design principle is also shared in other studies and frameworks, e.g., [EPH01; Dro+12; Dic07]. In [BMT12b; BMT12a], the authors suggest a MHW architecture which allows workers to additionally communicate directly together after receiving a task from the master hence exposing less communication bottleneck to the masters. This actually implies a kind of communication overlay, and is inline with other fully distributed or peer-to-peer existing algorithms, e.g., [FM87; IF00; Dic07; Dja13; Meh+09; Meh+08]. Each peer in such systems has in general a local pool storing all work units for processing. When the local pool of a peer is empty, the peer basically sends work requests. Upon receiving a work request, a peer shares some work units from its local pool to the requesting peer. This is actually performed according to some load balancing policy. To cite a few, in [LM92; TLM95], the authors proposed to balance the workload of a peer with its neighbors by means of a weight function which takes into account the quality and the quantity of generated subproblems. The quantity is the number of subproblems in a local pool. The quality is the minimum cost of generated subproblems of a local pool.

GPU BASED PARALLEL B&B. The previous approaches and related issues (pool management, synchronization issues, memory mapping, hierarchical and distributed architectures, load balancing, etc), are continuously revisited with respect to the devices and hardwares that can compose a large-scale compute environment. In particular, one can find a number of studies dealing with the integration of GPU devices in parallel B&B, e.g., see [Cha13] and the references there-in. A single GPU device can offer an impressive compute power when bounding different tree nodes in parallel, which is the main type of parallelism that a GPU device might offer. In fact, implementing the whole B&B process inside the GPU device is a difficult task, mainly because of thread divergence issues and the difficulty in handling efficiently the different memory levels. As mentioned before, the obtained speed-ups with respect to a single CPU are hence problem-depend since the bounding operation depends on the problem one wants to tackle. In the context of our work, we rather focus on the scalability issues. In fact, when multiple GPU devices are combined with multiple multi-core CPUs, most of the existing approaches fail to be highly scalable which is mainly attributed to the heterogeneity of the underlying compute systems and the unbalanced and fine-grained workload of parallel B&B, e.g., [Gmy+17; CM16; CM15].

2.2 CONTRIBUTION #1: DYNAMIC LOAD-BALANCING IN B&B

The main contribution of the work described in the rest of this chapter is to bring high performance on the scene of parallel optimization, by cross-fertilizing parallel B&B with general-purpose dynamic load balancing algorithms; thus bridging the gap between the solving of difficult optimization problems and the establishment of novel large-scale distributed and parallel algorithms.

As discussed previously in Section 2.1.3 about the main challenges that one has to face, and as witnessed by the bench of studies described in the previous section, the most recurrent issue in the design of parallel B&B is how to balance the workload dynamically at runtime. This is since the parallelism exposed by generic B&B is straightforward in our opinion; but, keeping all PUs busy doing useful work at any time of the execution is the main point that may prevent high performance and scalability.

Although being critically important, balancing workload is not always addressed in an explicit manner. This can mainly be attributed to the technical complexity of setting up an effective B&B parallelization on top of a specific compute environment, especially when dealing with a new type of parallel hardwares or parallel technologies. Interestingly, there exist a relatively well-established body of research, coming form high performance and parallel computing community, that deals explicitly with dynamic load balancing. In this section, we will in particular focus on the first steps we made in leveraging such techniques for parallel B&B in a large-scale compute environment. To better position our first contribution, we start providing a brief overview of state-of-the-art general purpose (i.e., independent from B&B) load balancing techniques.

2.2.1 Work Stealing: the Basics in a Nutshell

Load balancing is tightly related to the more general problem of task scheduling which is a well known problem that, besides B&B, can occurs in a different number of situations and application fields, e.g., [Kum+94; XL97; WR93; Lar+17]. In our setting, we are interested in the *Dynamic* load balancing of the workflow generated *at runtime* by a parallel application, that is, when the work units or the tasks of the corresponding application are generated during the course of the computations, and nothing about what each PU is executing, neither the cost or the number of work units, can be assumed beforehand. As such, the main goal of a dynamic load balancing method is to offload work units from overloaded PUs to underloaded/idle PUs dynamically at runtime in order to ensure that all PUs have approximately the same workload at any time of the application execution.

Different general-purpose methods have been developed so-far [BS81; ELZ86; KGV94; SKS92]. They are often fully distributed and implemented using multiple work pools. More specifically, each PU is associated with a single pool for storing and sharing the tasks generated at runtime. Existing methods can then be distinguished in three types, depending on who is the main initiator of the load balancing operations. In *work pushing* [ELZ86], it is the role of *overloaded* PUs to balance workload. In this class of methods, a PU with a non empty work pool automatically offloads tasks to an appropriately chosen PU, typically when it has more tasks than a threshold value. Every PU has then to decide on how to select a target PU, as well as on how many tasks to offload and how often. In *work stealing* [BS81], it is the role of *idle* PUs to start a load balancing operation. When a PU finds no task in its local pool, it simply requests another PU; which then offloads tasks if its own work pool is not empty, or rejects the request. The idle PU repeats this process until effectively fetching some work somewhere. Similarly, a PU has to decide on how to select a target PU whenever it is idle and how many tasks to offload when receiving a request. The third class of approaches is without surprise *hybrid* [SKS92], in the sense that it combines the previous two approaches by both offloading tasks from overloaded PUs, or stealing on demand when PUs are idle.

In work-pushing, good load balancing strongly depends on how often tasks are offloaded in a system, which is generally controlled using some predefined thresholds. If these thresholds are not well optimized to the correct value depending on the target application, tasks will be moved ahead either so often or so rarely. Furthermore, work pushing could be unstable when the granularity of workload is high and the application highly irregular, i.e., even when the system load is high at some stage of the execution and all PUs are actually busy, task offloading will still occur. Hence, it can be argued that work pushing is not a good choice for highly irregular applications like parallel B&B. In contrast, work stealing appears to be more suited, and it is actually chosen as a scheduler for load balancing in several well-established softwares and frameworks (e.g Cilk [FLR98a], Intel TBB [Int], OpenMP 3.0 [Ope11] and Javelin [Nea+00]).

In work-stealing, a PU running out of work is called *thief*, whereas those that service thieves' work requests are called *victims*. The most important factors for an efficient work-stealing based protocol are: (i) the selection strategy adopted by a thief to chose its victims and (ii) the work sharing strategy which is the amount of work offloaded from a victim to a thief. This was addressed in a number of studies and in different application settings, e.g., [MIY11; Din+07; Din+09; Cyb89; FLR98b; Oli+07; OP08; Sar+11; MIY11; QW10a; RLP11].

The main issue when designing an appropriate selection strategy is to minimize the time the thieves stay idle searching for work. The most simple selection strategy is a random one, i.e., a thief chooses one victim uniformly at random. The so-obtained *random work stealing protocol* is actually at the corner stone of several other variants. The work sharing strategy is in turn responsible for balancing the load evenly between a victim and a thief; which will subsequently becomes a potential victim. The most common strategy is the *steal-half* strategy which consists in offloading half of the work units available at the victim local pool. The goal behind coupling an appropriate victim selection with an appropriate work-sharing, is to flood the whole idle PUs in the system with fresh work as quick as possible, and hence to maximize the global parallel efficiency of the system.

From a purely theoretical perspective, Blumofe and Leiserson proved in a seminal paper [BL99] that random work stealing is essentially optimal (up to some constants) in terms of time, and communication. To be more precise, a random work-stealing algorithm for scheduling the so-called *fully strict (well-structured) multithreaded computation* model is presented and analyzed in the so-called *atomic-access* model. Under those theoretical assumptions, which are not detailed in this document for clarity, it is proved that the expected time of random work-stealing when executed by p identical parallel processors is $T_p = T_1/p + O(T_\infty)$, where T_1 is the minimum serial execution time of the multithreaded computation and T_∞ is the minimum execution time with an infinite number of processors. Similar tight bounds (for divide-and-conqueror computations) are also provided for the amount of communication required between the p processors, which is actually better than the bounds previously known for work-sharing. Such results, even restricted to some theoretical assumptions, comfort us in the idea that random work stealing is a highly effective approach to dynamic load balancing, especially in our target parallel B&B.

2.2.2 Tree-based Dynamic Load-Balancing

Our first contribution is on the design of a work stealing distributed protocol improving the performance of basic random work stealing in a large-scale distributed environment [Vu+12]. The main idea is that structuring computing nodes in a specific overlay can help addressing the issue of fetching work efficiently. Based on our previous investigations on the behavior of peer-to-peer B&B algorithms [Dja13], we describe in the following the core idea of our tree-based work stealing protocol for B&B dynamic load balancing.

STEALING ON TREE EDGES. Let us assume that PUs are organized logically following a rooted tree. The target application to be parallelized is initially pushed at the root PU. An

idle PU steals synchronously downwards and upwards in the tree. In the down phase, every idle PU first requests its children. The steals are sent sequentially by choosing a child uniformly at random at each step. Then, if and only if all children are idle, a steal is sent at last upwards to the parent. Conceptually, this corresponds to a random work stealing strategy, but considering only the set of children that have not sent a request upwards yet.

STEALING ON BRIDGE EDGES. The previous protocol exhibits much locality, since tasks inside a subtree will always be completely finished before load balancing requests are sent to the parent. This property leads to some drawbacks, basically because a PU only steals upward when its subtree becomes entirely idle. Therefore the whole subtree remains idle during the round trip of the parent upward steal request. To handle this issue, while maintaining a low communication overhead, we introduce what we call Bridge-based work steals with the aim of speeding up work flow from overloaded subtrees to under-loaded ones. Apart from the tree edges, we propose to connect PUs being far away each other using random *bridge* edges. Those bridge edges are to be viewed as logical shortcuts that can be traveled by work to reach under-loaded subtrees more quickly. In this approach, every PU v *further* requests work from *one* PU r through a bridge edge $b_{v\to r}$ chosen uniformly at random among PUs *being neither children nor parent*. More precisely, *in parallel while requesting its neighbors in the tree*, every idle PU v *asynchronously* sends a steal request over $b_{v\to r}$. Such an asynchronous steal request does not block v waiting for a response from r. Instead, v is allowed to concurrently search for work from its neighboring PUs in the tree, as described previously. If the remote neighbor r owns work, then it immediately services v. If r is idle, then this means that r has already sent an asynchronous work request through its bridge edge, and it is also requesting its respective direct neighbors. Thus, whenever an idle node, say p, gets work from its neighbors or through its bridge, then it immediately services all nodes from which a steal request was received. Let us remark that this distributed strategy operates in a recursive manner, implicitly building up a logical cluster of idle PUs. Consequently, all idle PUs are more likely to cooperate efficiently in searching for fresh work.

TREE-DEPENDENT WORK SHARING. The previous protocol is further combined with the following work sharing strategy. We propose to dynamically adjust the amount of work transferred from one PU to another one according to the size of the overlay subtrees. Based on the observation that idle PUs should not be selfish when searching for work, but should acquire enough work to serve their neighbors, our work sharing policy is overlay-dependent: a PU simply divides its current work into the ratio of its own tree size and the tree size of the requesting PU; which is to contrast with a standard steal-half strategy.

OTHER B&B SPECIFIC OPERATIONS To gain in generality, some B&B specific issues were hidden in the previous discussion. The work acquired by a PU is in fact assumed to represent a B&B work unit which might be a B&B active node, as well as any alternative encoding of B&B subtrees. Each PU is then responsible for executing a B&B on the work units it acquires, that is, the standard branching, bounding and pruning operations as specified by a serial B&B and with respect to the optimization problem. Moreover, the sharing of the best feasible solution found by a B&B process is managed by sending messages through the overlay tree. Finally, termination detection is also handled in a distributed fashion following tree edges. Notice that without the bridge steals, termination can be detected straightforwardly by the root of the tree; however, we manage to carefully implement a wave-based distributed protocol in order to safely ensure that no work units are still traveling along an asynchronous work-transfer responses.

2.2.3 Selected Experimental Results

We present in the following some selected experimental results from our extensive and comprehensive study [Vu14; Vu+12]. It is worth-noticing that conducting a fair and sound experimental study is not straightforward, especially when dealing with parallel and distributed algorithms applied to optimization. In our work, we put much attention not only in evaluating the *relative* performance of the proposed protocols, but also to elicit their *behavior* in terms of scalability and parallel efficiency. We emphasis that the aim of the following paragraphs is to provide a very brief synthesis of our experimental procedure and its rationale, while focusing on the main lessons that we were able to learn.

BENCHMARKS. We consider the solving of the Flow-Shop optimization problem (i.e., minimize the makespan C_{max}) and the well-known Taillard's instances $\{Ta_{21}, \cdots, Ta_{30}\}$ of the family Ta-$20 * 20$, i.e., 20 jobs and 20 machines [Tai93]. Besides, we consider the so-called Unbalanced Tree Search (UTS) [Oli+07] benchmark, which is the reference general-purpose benchmark from the parallel computing and HPC community. It simply consists in exploring/counting the nodes of a parameterized tree with extreme variation/imbalance in the relative size of its induced subtrees. The tree is actually unknown beforehand and can only be constructed at runtime using a splittable, deterministic random stream generated using the SHA-1 secure hash algorithm. The UTS was actually designed to be *the* representative adversary benchmark to evaluate the *parallel* performance of *irregular* state space exploration and combinatorial search algorithms like parallel B&B.

COMPETING PROTOCOLS AND TESTBED. Besides studying the behavior of our tree-based algorithm with different parameter settings, we consider three algorithms based on the three main paradigms used so-far for B&B: a Master/Worker algorithm as introduced by the B&B@Grid approach [MMT07a; MMT07b; Mez07], a hierarchical Master/Worker algorithm as introduced by the so-called Adaptive Hierarchical Master-Worker (AHMW) B&B approach [BMT12b; BMT12a; Ben12], and a fully decentralized algorithm based on the standard random work-stealing paradigm using a steal-half strategy, considered as a HPC-oriented baseline algorithm. All protocols are experimented on top of the Grid'5000 French national experimental grid [Gri]. Two clusters were involved in our experiments and up to 1200 compute cores are used.

EXPERIMENTAL RESULTS. Our first finding is that our proposed protocol is able to perform at its best when configured with an overlay-tree having a well tuned average degree. This is because most of the work is expected to flow over the tree edges, and hence a small diameter tree is mandatory for work to reach idle process efficiently. Besides, stealing using bridge-edges is proved to provide a significant improvement especially when considering the UTS benchmark. Among the different competing algorithms, the hierarchical approach, which at first sight presents some similarity with ours since it is based on a structuring some masters using a tree overlay, is surprisingly found to perform very poorly. For instance, at the relatively moderate scale of 200 cores, our approach is found to be more than 10 times faster in average over all the considered flowshop instances. This result can actually be explained by the particular B&B work mapping used in [BMT12b; BMT12a]. A BFS-like problem-specific strategy is in fact adopted there-in to explore the B&B tree when solving the flowshop problem, which is sufficient to provide a good parallel efficiency of the hierarchical architecture, but very likely not to provide optimal parallel solving times. Turning to the MW approach, which was actually believed to be a state-of-the-art approach for solving large-scale Flow shop instances, it is found to suffer a scalability issue as illustrated in Fig. 1 showing its relative performance at a large scale of 1200 cores. We actually found that the master worker approach can only be efficient at the small and moderate scales, or when considering huge B&B instances for which distributing workload is anyway not a

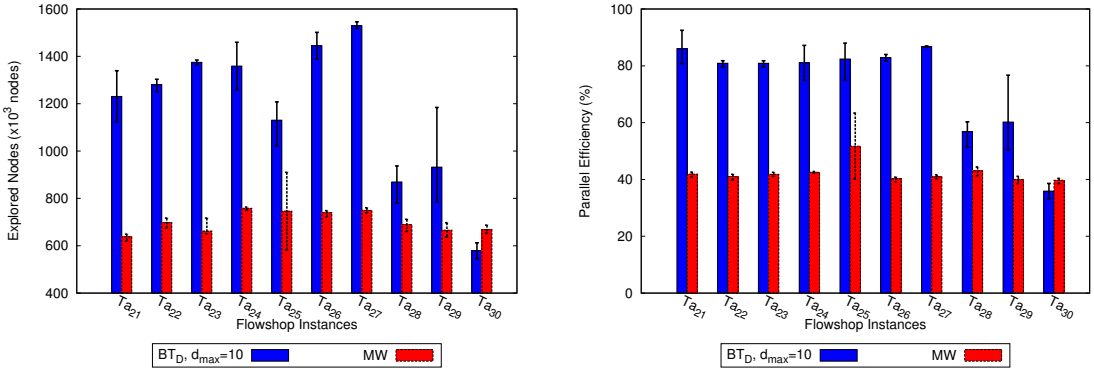


Figure 1: Relative performance when using 1200 compute cores of a MW approach compared to our approach (the notation BT_D refers to stealing with Bridge edges and following a Tree overlay constructed deterministically with maximum degree d_{max} at most 10). **Left**: number of explored nodes per second that is the number of bounding operations performed in parallel. **Right**: B&B parallel efficiency that is the proportion of time devoted to performing B&B useful work against waiting for some work to process.

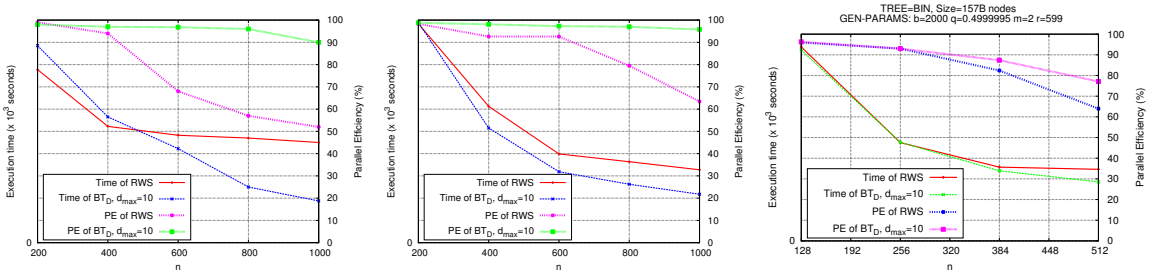


Figure 2: Execution time and Parallel efficiency (PE) when scaling the number of PUs and using respectively our approach (BT_D) and random work stealing (RWS), to solve two representative fine-grained B&B instances: Ta_{21} (**Left**), Ta_{23} (**Middle**) and a reference UTS instance (**Right**). Execution time and Parallel efficiency (PE), as a function of overlay size n for UTS.

challenging issue. From a purely parallel perspective, dealing with fine-grained parallelism in B&B is the main critical issue as we scale up the distributed resources. Lastly, work-stealing based protocols, are found to provide a fairly good performance not only for parallel B&B applied to the flowshop problem, but more generally when applied to the problem-independent UTS benchmark. As illustrated in Fig. 2, standard random work stealing is confirmed to be highly efficient in the moderate scales. However, there is still an opportunity for further improvements as demonstrated by the relative performance of our tree-based protocol. At the largest scales requiring to deal with very fine-grained parallelism, random work stealing reaches its limits, since idle nodes try to catch victims 'blindly' using random requests. In contrast, our tree centric approach tends to minimize communication delays by distributing the load in a more deterministic/cooperative manner and the gain in parallel efficiency, thus in speed-up, is substantial.

2.3 CONTRIBUTION #2: NODE-HETEROGENEOUS WORK STEALING

As a second contribution, we consider the design of parallel B&B for node-heterogeneous platforms harnessing a mixture of shared memory and distributed memory components. In order to achieve high performance, heterogeneity brings further challenges especially when dealing with dynamic load balancing. In this respect, we present in this section a large-scale multi-core multi-CPU multi-GPU approach based on leveraging the work-stealing paradigm and providing near-optimal linear speed-ups under different compute settings. Although some recent attempts on the subject can be reported, the performance of the proposed approach, as well as its scalability, constitute to the best of our knowledge, the state-of-the-art for node-heterogeneous parallel B&B.

2.3.1 A large-scale Multi-core Multi-CPU Multi-GPU B&B Approach (3MBB)

MAPPING B&B WORK FOR CPUS AND GPUS. Motivated by the difference between the compute abilities of CPU and GPU cores, we consider a combination of a parallel DFS and BFS tree traversals when selecting the B&B nodes to be bounded in parallel. Since a GPU can be efficient only when it can bound *many* tree nodes in parallel, we have to prepare enough work (by decomposing enough subproblems) before activating the GPU device computations. A DFS traversal allows us to quickly go deep in the search tree, hopefully finding good complete solutions quickly; However, it does not allow us to infer much parallelism to be handled by the GPU device. In contrast, a BFS traversal makes it possible to generate sufficiently many B&B nodes to push into the GPU device. Hence, a DFS is always performed in the case of a CPU where as a BFS is preferred when dealing with GPUs.

Moreover, input data containing several B&B tree nodes has to be transferred from the CPU host core to the GPU memory, then a GPU kernel is executed on the input data, and the outputs (the computed bounds) are copied back from the GPU to the CPU host. In standard CPU host / GPU device execution, the previous operations are done sequentially. In other words, while the CPU host is performing *select*, *branch* or *prune* operations, or even while copying data to and from the device, the GPU is stalled. Similarly, while the evaluation of B&B tree nodes is running inside the GPU device, the CPU host is stalled. To gain in parallel efficiency, the CPU host can in fact dispatch operations into the GPU device asynchronously and continue its computation. Hence, a sequence of operations (namely: copy data from CPU to GPU, perform parallel bounding operations at GPU device and copy results from GPU to CPU) is wrapped into *a stream* which is *asynchronously* dispatched to GPU device for execution. This simple idea, which is straightforwardly enabled using CUDA programming facilities, allows us to significantly speedup the runtime of some existing approaches [Cha13] by up to 40%. Technically speaking, and although being straightforward, this illustrates the importance of properly mapping B&B parallelism with respect to the enabling compute technology.

More importantly, there exist two main levels of parallelism exposed by the distributed environment we are interested in: *intra-node* parallelism which refers to the shared memory computations among CPU cores and the *inter-node* parallelism which refers to the (message-passing) distributed computations. Having in mind that the difference in communication cost between shared memory and distributed memory systems can be very substantial, the main challenge is then to distribute workload efficiently. As discussed next, we adopt a hybrid approach leveraging random work stealing to operate efficiently with respect to the two above mentioned levels of parallelism.

INTRA-NODE PARALLELISM. Let us first zoom in the case of a single shared-memory multi-core component. We basically use asynchronous multiple work pools in our design and

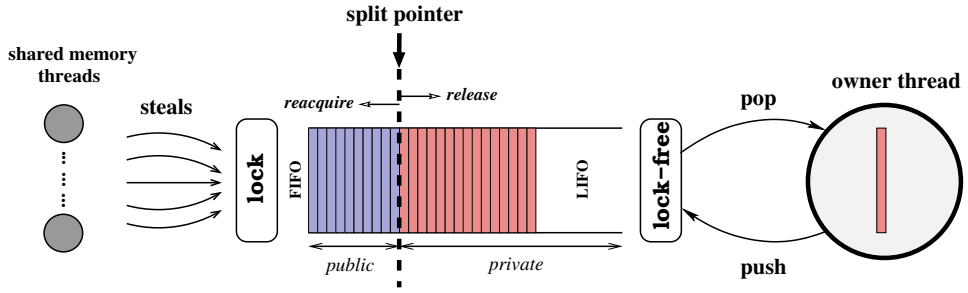


Figure 3: Simplified view of a split work pool

we consider random work stealing for load-balancing. The straightforward approach where each thread manages a fully sharable work pool introduces high communication overhead as locking is required to synchronize the multiple accesses to the work pool of a given thread. We found that the most appropriate approach is to use a work pool split into private and public part as originally described in [Din+09]. The private region is managed by the owner thread and the public one is exposed to other threads. These two regions constitute a single data structure endowed with a *split* pointer (identifying the frontier between the two regions) in order to avoid any data manipulation overheads. The amount of tasks of the private and public regions are then adjusted by moving the *split* pointer forward or backward without any memory copies as illustrated in Fig. 3. The private portion works like a Stack (LIFO) and the public portion works like a Queue (FIFO). The LIFO property of the private region allows threads to perform DFS search on the B&B tree. The FIFO property of the public region allows threads to share coarse grain B&B subproblems that are likely to generate more children subproblems; and thus it encourages the transfer of useful B&B workload. The steal attempts from the public region of a given thread are handled using a standard lock operation; but the victim thread is lock-free with respect to its private region so that it can continue processing B&B subproblems and freely push/pop tasks.

A crucial feature is to adjust the amount of work units in the private and public region at runtime. For this purpose, the split pointer is associated with two operations: *release* and *reacquire*. The *release* operation is handled by the owner thread every time the public part gets empty (because work was stolen so far by other threads), and the split pointer is then moved to *half* of the private queue. The *reacquire* operation refers to the reverse operation which we handle depending the following two rules. *GPU split pointer*: refers to performing the *reacquire* operation in advance before the private region is completely empty. This is only performed when a core is also hosting a GPU device. In fact, it might happen that the private part does not allow to generate enough work to push into the GPU device. *CPU split pointer*: This is with respect to any thread even those not hosting any GPU devices. In this case, the reacquire operation is performed if and only if the private region is empty. The split pointer is always moved to *half* the public region (if not empty).

To summarize, every time a thread in *one single shared memory component* wants to process a B&B subproblem, it takes one from the front of its local queue which actually lies in the private part. Every time a thread generates new subproblems, they are pushed again at the front of the private part. When a thread runs out of work, it attempts to steal work from the public part of another victim thread chosen uniformly at random. If the public part of victim's queue is not empty, then the thread is able to steal some work from the tail of the queue after locking it, and hence it can push stolen work at the front of its own private queue. Otherwise, no work is found and the steal operation is renewed until some work is fetched or global termination is announced.

INTER-NODE PARALLELISM USING HYBRID STEALING. In our approach, stealing across distributed nodes is only enabled when all threads detect that there is no work available in any pool of the shared memory component they belong to. Whenever a shared memory component runs out of work, distributed steals are performed by *solely one thread*, elected initially as *the leader*. A leader functions like the other threads with one main difference: it additionally manages the distributed steals when work is no more available locally. A distributed steal then consists in sending work requests to another remote leader chosen uniformly at random using message passing. For the above protocol to work correctly, we have to manage termination properly, which is done using a tree overlay topology spanning the leaders. Since distributed steals are initiated if and only if all the threads at the leaders are idle, it is easy to see that whenever termination is detected among leaders, the other threads can be informed immediately using a shared memory variable maintained by every leader. In the same way, the exchange of the best solution needed for the B&B pruning is handled distributively using the same tree overlay.

ADAPTIVE AGGREGATED STEAL GRANULARITY. To set the amount of work to be transferred, we consider the following adaptive strategy. Every PU maintains at runtime a measure reflecting its power, say x, which is continuously with respect to the work processed in previous iterations. We simply use the average time needed for processing a B&B subproblem. The amount of work to be transferred is then in the proportion of $x/(x+y)$, where y is the computing power maintained locally by the victim. To be more precise, for a distributed work transfer among two leaders in different shared memory components, the power of a compute node is measured as an aggregated value of all the threads at the shared memory node, and the amount of work is computed with respect to all available tasks in the public regions of the underlying local threads. A thief leader i computes the aggregated power $X_i = \sum_k x_{i,k}$ (where $x_{i,k}$ is the computing power of thread k leaded by i) and sends a work request to a randomly selected victim j while wrapping the value of X_i. Upon receiving a steal request, the victim leader j also measures its aggregated power $X_j = \sum_\ell x_{j,\ell}$. The amount of work to be transferred is then in the proportion of $\frac{X_i}{X_i+X_j}$. Technically speaking, the victim leader j collects $s_{j,\ell} = t_{j,\ell} \cdot \frac{X_i}{X_i+X_j}$ work units from the public region of every work pool of all the local threads ℓ (where $t_{j,\ell}$ is the amount of work at the public region of thread ℓ leaded by j). A total amount of $S = \sum_\ell s_{j,\ell}$ work units are then transferred to the requesting thief i. Notice that this work-sharing strategy is essentially an adaptation of the steal-half strategy designed in order to cope with the heterogeneous compute power of the underlying PUs (CPUs or GPUs).

2.3.2 Selected Experimental Results.

Our approach comes with several components that we manage to experiment in a systematic manner in order to better understand the benefits of intra-node and inter-node parallelism, as well as, the impact of work-stealing as initially described in the extensive study provided in [Vu14; VDM13; VD16]. Here-after, we highlight our most distinguishable findings.

EXPERIMENTAL SETTINGS Three clusters C_1, C_2 and C_3 of the Grid'5000 French national experimental grid [Gri] were involved in our experiments. Cluster C_1 contains 10 nodes, each equipped with 2 CPUs of 2.26Ghz Intel Xeon processors with 4 cores per CPU. Each node is coupled with two Tesla T10 GPUs. Each GPU contains 240 CUDA cores, a 4GB global memory, a 16.38 KB shared memory and a warp size of 32 threads. Cluster C_2 (resp. C_3) is equipped with 72 nodes (resp. 34 nodes), each one equipped with 2 CPUs of 2.27 Ghz Intel Xeon processor with 4 cores per CPU (resp. 2 CPUs of 2.5 Ghz Intel Xeon processor having 4 cores) and a network card Infiniband-40G. We use the standard Taillard's Flowshop instances in the family $20 * 20$. Only the bounding operation has to be executed inside the

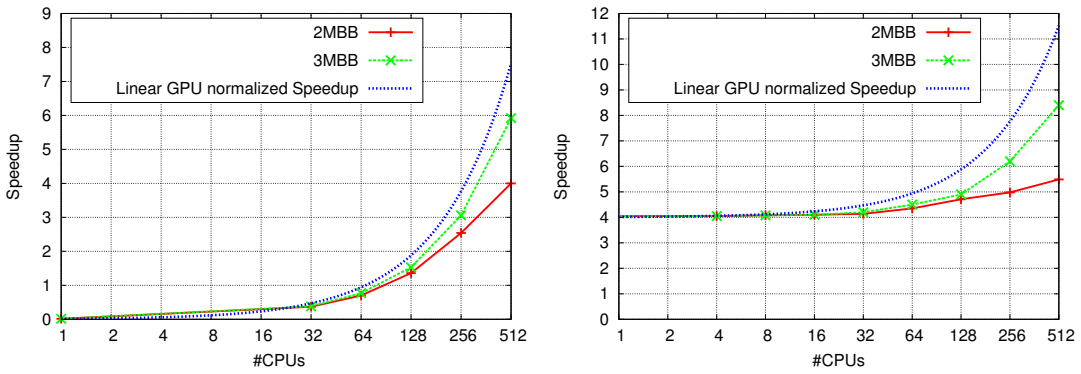


Figure 4: Speedup (GPU-normalized) of the 3MBB approach compared to 2MBB when scaling CPUs and using 0 GPU (**Left**) and 4 GPUs (**Right**). The X-axis is in the log scale and reported speed-ups are w.r.t. one GPU.

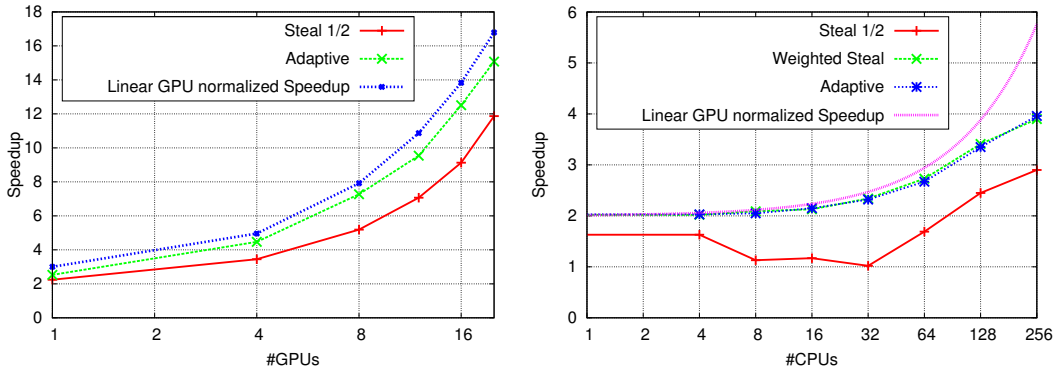


Figure 5: **Left**: Speedup when scaling heterogenous GPUs (1/2 with the maximum compute power $s^\star$, 1/4 with $s^\star/2$, 1/4 with $s^\star/4$), and using 128 heterogenous CPUs (1/2 from cluster C_2, 1/2 from cluster C_3). Speedup is w.r.t. the GPU with the maximum compute power $s^\star$. **Right**: Speedup when scaling CPUs and using 2 (homogeneous) GPUs.

GPU devices. We consider the kernel implementation provided by [CM12; Mel+12] and use it in a blackbox manner. For the sake of analysis, we manage to configure the compute environment so that different GPUs could have different speeds and hence to be able to study the impact of different compute powers of CPUs and GPUs rendering different degree of heterogeneity. Besides, we consider to study our approach when disabling the inter-node parallelism, that is when considering only distributed memory PUs. Accordingly, our approach is denoted by 3MBB in the following while the variant performing only remote (standard random) steals is denoted by 2MBB (multi-CPU multi-GPU B&B).

EXPERIMENTAL RESULTS. As it can be seen in Fig. 4, our approach (3MBB) is able to obtain a near-linear speedup, while continuing to scale rather efficiently compared to the 2MBB approach. This shows that taking into account both inter- and intra- node parallelism, which is inherent to shared and distributed memory heterogeneous systems, is crucially important for scalability. Actually, for moderate scales, the 2MBB is still a fairly good approach which again indicates that random work stealing can be extremely efficient. However, one should pay much attention in setting the amount of work to be transferred between the different PUs especially when their relative compute power is significantly different. This is shown in Fig. 5, where different heterogeneous settings are experimented with the 2MBB approach. In particular, we can see (Fig. 5 Left) that a system containing only GPUs devices, but with different compute powers can only deliver its best performance when the amount of work to be stolen is set proportionally. More interestingly, when considering few (homogenous)

GPUs devices and scaling up the number of CPUs (Fig. 5 Right) from small to large, the overall performance of the system can even downgrade at some intermediate scales; basically because the most powerful PUs are disturbed too often by the other less powerful PUs, hence resulting in a situation where work is traveling back and forth from these two types of PUs. In contrast, our approach can deal with such issues in a consistent manner and allow us to take full benefits from every available PUs, possibly having a relatively restricted compute power.

2.4 CONTRIBUTION #3: LINK-HETEROGENOUS B&B LOAD BALANCING

Continuing our efforts on the design of effective dynamic load balancing algorithms for parallel B&B, our last contribution addresses the challenges behind the *link-heterogeneity* of large-scale platforms. Link-heterogeneity appears as distributed resources are typically connected through different communication networks [Kie+02; PBH99]. Previous studies, coming from the high performance community, have mostly considered the heterogeneity in communication speed by investigating the specific hierarchy proper to each compute environment such as multi-cluster platforms [BBB96], geographically distributed multi-cluster multi-site grids [VKB01; Van+04; JH13], and others [GB10; Pil+12; ABB00]. Skillful design practices have been gained; however, the designed solutions and protocol variants are essentially platform-dependent. In particular, there is still little insights [Pil+12] into how network latency impacts fine-grained parallelism and how distributed communications have to be optimized to face the increasing complexity of such heterogeneity in a portable and unified manner. This makes it more complicated for designers and programmers to deal with different types of distributed environments, which may result in ad-hoc implementations burdening the parallelization process and leading to non-efficient protocols. On the other hand, a knowledge about the computing platform could not be available at the time an application needs to be effectively deployed. For instance, cloud-oriented infrastructures have the distinct characteristic of hiding the actual physical mapping of resources, and recent studies [Zha+11] showed that the interconnection latencies in virtualized environments pose the most severe issue when executing HPC workloads. In the following, we first describe a very simple platform-independent model for link-heterogeneity and some related load-balancing algorithms. We then discuss our main contribution on the design a general purpose link-heterogenous work stealing protocol [VD14].

2.4.1 Dynamic load balancing under Link-Heterogeneity

A SIMPLE DISTRIBUTED MODEL. Generally speaking, modeling the heterogeneity of modern multi-computer distributed systems is a difficult task which is, *per se*, the subject of several dedicated research investigations, e.g. [Cap+05]. We shall consider an abstract model upon which we can easily think and build generic link-heterogenous protocols independently of a particular target platform. In line with previous studies [BR10b], we focus on the case of distributed nodes having identical computing powers and heterogeneous communication resources. The set of computing nodes V are fully connected and form a complete interconnection graph G, i.e., every node can communicate with any other node in the system by message passing. To model the interconnection heterogeneity, we endow the graph G with a function $\omega : V \times V \to \mathbb{R}$; which assigns for each pair of nodes i and j a real-valued weight $\omega_{i,j}$ informing about the cost experienced by node i when communicating with node j. The more the communication over edge (i, j) is costly, the higher is the weight $\omega_{i,j}$. Every node i is assumed to know solely the local weights $\omega_{i,j}$ connecting it with every other nodes $j \neq i$; thus hiding all the architectural characteristics the physical

resources. In our work, we concentrate on link heterogeneity so that the function ω shall simply be viewed as a measure of nodes pairwise latency; that is, the network delay in a point-to-point message exchange. The above model exposes a flat view of the distributed environment which is to contrast to the hierarchical nature of grids, clouds, and more generally large-scale compute platforms. We argue that depending on how the function ω is defined, this is however sufficient to reason about link-heterogeneity

Considering such a distributed model, we adopt a work-stealing based approach. It is worth-noticing that the theoretical results on the optimality of random work stealing (RWS) [BL99] as discussed previously in this document cannot hold unless the weights $\omega_{i,j}$ are pair-wise equal, e.g., a multi-cpu single cluster network-homogenous system. When the $\omega_{i,j}$ are non-uniform, it is not difficult to see that victim selection in work-stealing is still the key for workload to fold efficiently. In the following, we review two state-of-the-art HPC approaches that were specifically designed to deal with non-uniform communication latencies.

PROBABILISTIC WORK STEALING (PWS). The general idea of PWS [QW10b] is to pick up nearby processors in priority. It suggests to use a measure estimating the distance between computing nodes; and to modify the classical RWS algorithm in the following way: "the probability to choose a target computer for steal attempts is not uniform anymore but instead proportional to the inverse of the distance between the thief and the target". This corresponds to every thief i choosing its victim with probability:

$$p_{i,j} = \frac{\frac{1}{\omega_{i,j}}}{\sum_{j \neq i} \frac{1}{\omega_{i,j}}} \quad (1)$$

Such a victim selection procedure enable to privilege the stealing over fast links without discarding the possibility of using slow links, in an attempt to reduce the average latency of steal requests. It also has the nice property of being inherently local and platform-independent – it enables to capture the possibly different levels of hierarchy that might be implied by the computing architecture without loosing in generality nor in efficiency, as experimentally shown in [QW10b] on a hierarchical system of 8 processors. Nevertheless, we found no in-depth investigations on large-scale more complex platforms. As the system scale increases, it is likely that the gap between communication costs over different links increases substantially. This might have the effect of decreasing drastically the probability of stealing over slow links; thus eventually isolating some compute nodes and making work stuck at few regions without being able to flow fairly and quickly in the system.

ADAPTIVE CLUSTER-AWARE RANDOM STEALING (ACRS). This protocol is an improvement of the so-called CRS protocol described originally in [VKB01]. As its name indicates, CRS was designed specifically for two level hierarchical platforms; where every sub-group of nodes running on a whole homogenous single-cluster needs to be explicitly identified, hence being platform-dependent. Given that every node is aware of its cluster, CRS extends on RWS by further allowing each node to steal work asynchronously from a randomly chosen remote clusters. Hence, in CRS, an idle node steals in his own cluster by performing intra-cluster synchronous steals as in RWS, and in parallel, it also sends one additional asynchronous intra-cluster work request. ACRS [Van+04] extends on CRS by adapting the probability of choosing inter-cluster victims: the probability of choosing a remote cluster is inversely proportional to the communication cost between clusters. Although the CRS and ACRS algorithms can be proved to provide good performance, they still suffer from some design limitations. Basically, and besides being platform-dependent, they can lead to situations where tasks might be transferred several times in advance for nothing, e.g., when both inter- and intra- cluster steals are successful, this might result in a ping-pong effect where tasks are moving back and forth between clusters.

Algorithm 2: Link-Heterogenous Work-Stealing (LWS): distributed high level pseudo-code for every node $i \in V$.

Data: $V_{\setminus i} = \{1, 2, \cdots, n\} \setminus \{i\}$: neighbors' identifiers; T: a parameter;

1 flag_asyn_steal_request $\longleftarrow$ false ; flag_syn_steal_request $\longleftarrow$ false ;
2 $\forall j \in V_{\setminus i}$, $c_j \longleftarrow 1$; $Y \longleftarrow 0$;
3 **while** *termination* **do**
4 job $\longleftarrow$ check for work from local pool ;
5 **if** *job* $\neq \varnothing$ **then**
6 process job ;
7 **else**
8 **if** $\sum_{j \in V_{\setminus i}} c_j \ \% \ T = 0$ **then**
9 $V_{asyn} \longleftarrow$ **Partition_Victims()** ;
10 $\forall j \in V_{asyn}$, $q_j \longleftarrow \frac{1/\omega_{i,j}}{\sum_{\ell \in V_{asyn}} 1/\omega_{i,\ell}}$;
11 $\forall j \in V_{\setminus i}$, $r_j \longleftarrow \frac{c_j}{\sum_{\ell \in V_{\setminus i}} c_\ell}$;
12 $\forall j \in V_{\setminus i}$, $p_j \longleftarrow \frac{p_{i,j} \cdot r_j}{\sum_{\ell \in V_{\setminus i}} p_{i,\ell} \cdot r_\ell}$;
13 **if** $\neg$ *flag_asyn_steal_request* && $Y \geqslant X$ **then**
14 $s \leftarrow$ a node in V_{asyn} selected with prob. q_s ;
15 **Send** an asynchronous work request to s ;
16 flag_asyn_steal_request $\longleftarrow$ true ;
17 **if** $\neg$ *flag_syn_steal_request* **then**
18 $k \leftarrow$ a node in $V_{\setminus i}$ selected with prob. p_k;
19 **Send** a synchronous work request to k ;
20 $c_k \leftarrow c_k + 1$; flag_syn_steal_request $\longleftarrow$ true ;
21 **Handle_Timer()** ;

2.4.2 A Generic Link-heterogenous Work-Stealing Algorithm

Inspired by PWS and ACRS, we propose a new generic distributed algorithm called LWS [VD14] and depicted by the high level code of Algorithm 2. Our algorithm is based on the key observation that increasing work locality is mandatory to counteract link-heterogeneity in dynamic load-balancing. The more we structure the steal probabilities over the weighted graph G and encourage nodes to communicate over fast links; the more it is likely for the work to flow over a set of edges forming paths of minimum weights. Nevertheless, waiting for work over fast links might be less time efficient than directly acquiring work by stealing over slow links. This poses a dilemma which is difficult to face because workload is unpredictable and nodes can not assume in advance where some fresh work can be fetched deterministically. The key ingredients of LWS are: (i) to dynamically identify a set of preferred victims, (ii) to 'predict' when synchronous probabilistic steals would be sufficient to make work flow efficiently, and (iii) when further asynchronous steals would be necessary. This is achieved locally and adaptively at each distributed node as discussed in the following.

When becoming idle, a node first starts sending *synchronous* steals. A thief sends synchronous steals probabilistically as in PWS; but using a modified probability function to select victims (lines 17 to 20): Every node i stores the number of synchronous work requests c_j that have been issued towards node j (lines 2 and 20). The probability for node i to choose victim j is inversely proportional to the communication latency between i and j, and proportional to the local counter c_j. This probability is denoted by variable p_j (line 12) which is computed as a multiplicative aggregation of the two probability functions r_j (line 11) and

Procedure HandleTimer

```
1 if a reply msg from node ℓ is pending then
2     work ⟵ check for work by processing msg;
3     if work ≠ ∅ then
4         Unpack work and push into the local pool;
5     if ℓ = k then
6         if work ≠ ∅ then Y ⟵ 0 ; X ⟵ X + ω_{i,k} ;
7         else Y ⟵ Y + ω_{i,k} ; X ⟵ X/2 ;
8         flag_syn_steal_request ⟵ false ;
9     else flag_asyn_steal_request ⟵ false ;
```

$p_{i,j}$ (given by Eq. 1). Clearly, this strategy accentuates the locality between a thief i and its *previous* victims, and aims at isolating few very preferred victims from where it is likely to be very fast to check for work. The other victims, which are likely to be connected with slow links, are not completely discarded. They are in fact requested for work *asynchronously* to avoid loosing time waiting for steal replies. However, only a restricted set of victims is considered as follows.

Each node separates between preferred synchronous victims and asynchronous ones using (at runtime) a partitioning procedure (Procedure PARTITION_VICTIMS line 9). It is actually based on the k-means clustering algorithm with $k = 2$ (i.e., neighbors are clustered in two groups); and where the victims' counters c_j are used to define similarity between the two so-obtained means. More specifically, the group of victims having the lowest counters' values is identified as the set V_{asyn} (line 9) from which asynchronous probabilistic steals should be performed (line 14). Besides, in order to avoid an unnecessary work transfer, a thief starts an *asynchronous* steal first only after it makes a number of synchronous steal attempts towards its preferred victims. The starting signal is handled in procedure HANDLE_TIMER through control variables X and Y which play the role of *adaptive* timers. Our idea is to distributively *detect* the availability of work among nearby processors by *self-adjusting* a time window over which work is expected to flow synchronously. Inspired by the additive-increase/multiplicative-decrease feedback approach for congestion avoidance [CJ89], if a synchronous steal made by thief i to preferred victim k is successful then the waiting window X is increased proportional to network latency, that is by $\omega_{i,k}$. Otherwise, X is decreased by half and the 'elapsed time' Y is increased by $\omega_{i,k}$. Only after Y exceeds X that an asynchronous steal is sent to a victim s selected from V_{asyn} with probability q_s (line 14), inversely proportional to the communication costs.

To be complete, notice that we also introduce a parameter T to define the number of synchronous steal attempts that have to be performed before control variables are updated (line 8). We also omitted specifying in Algorithm 2 that nodes are concurrently checking for incoming messages, as well as the technical details regarding distributed termination detection which is handled using a tree-overlay as previously described in this document.

2.4.3 Selected Experimental Results

METHODOLOGY. In order to gain insights into the impact of link heterogeneity, we adopt an emulation-based experimental methodology which is to contrast to real or simulation-based experiments. On the one hand, real experiments involve running a real-application on a real experimental-platform, which is generally believed to provide high realism. However, experimenters face many difficulties to validate their algorithms, e.g., platform dependency, result reproducibility, etc. In the context of studying link-heterogeneity, this is even more challenging since it is difficult to set up the network in a particular and already established platform. On the other hand, simulation facilitates the study of complex configurations at

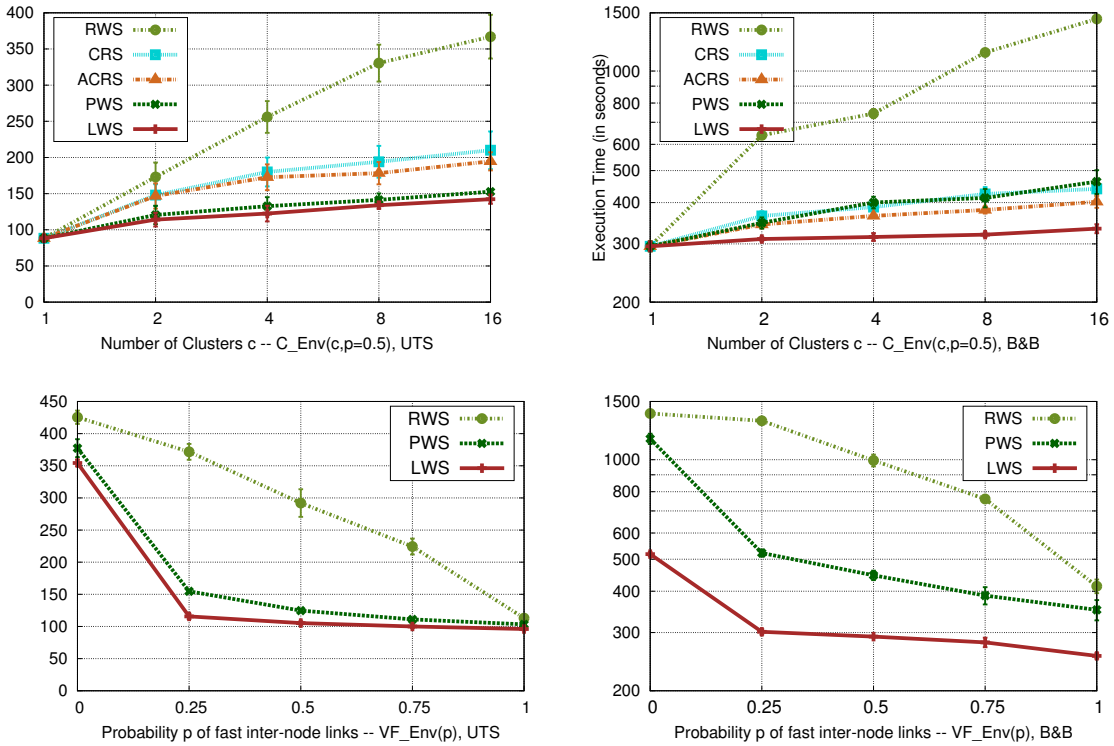


Figure 6: The obtained execution time of different competing algorithms and in different scenarios. **First column**: UTS. **Second column**: B&B (Ta_{21}). **Top**: C_Env$(c, p = 0.5)$ and **Bottom** VF_Env(p).

the prices of a relative loss in realism. By combining the advantages of these two experimental methodologies, *emulation* appears to be an appropriate solution for experimenting complex distributed configurations with real applications. We hence use Distem [Sar+13] to emulate a broad range of complex link-heterogeneous network configurations. Generally speaking, Distem is a distributed systems emulator for realistic environments appearing in cloud, peer-to-peer, high performance computing or grid systems. It uses virtualization to transform a homogeneous real-cluster into an experimental platform where nodes have different power and/or are linked together through a complex network topology; thus making it an ideal tool for our study.

We would like to emphasis that real-CPU compute nodes of a real distributed test-bed are used in our experiments; only network link latencies are artificially configured through Distem. In fact, Distem is used on top of one cluster of the Grid'5000 experimental grid [Gri]. The cluster in use has 92 nodes, each one equipped with 2 CPU of 2.5 Ghz Intel Xeon processor with 4 cores per cpu and a network card infiniband-20G. Besides the proposed LWS algorithm, the other competing protocols are: RWS [BL99], PWS [QW10b], CRS [VKB01] and ACRS [Van+04]. Every PU in our experiments is deployed with Distem on a dedicated physical compute core initialized and configured with the corresponding communication latencies. The latencies are managed internally by Distem without any additional operations at the application level. We use the Flowshop instances as target problems in our experiments, as well as the UTS benchmark as described previously in Section 2.2.3. As very brief overview of our experimental findings are described in the following.

RESULTS ON GRID-LIKE AND CLUSTERED ENVIRONMENTS (C_ENV). We here assume that nodes are grouped into some clusters where the network latency inside and outside the clusters defines a two-level communication hierarchy. In the first level, the latency between

nodes from the same cluster is set to a fixed value of 0.2 ms. In the second level, clusters are assumed to be fully connected with WAN links which are further split into two sub-groups, fast WAN links with latency in the range $R^{grid}_{fast} = \{30, 40, 50\}$ (ms); and slow WAN links with latency in the range $R^{grid}_{slow} = \{100, 150, 200\}$ (ms). Inter-cluster pairwise node latency is then picked uniformly at random from set R^{grid}_{fast} with probability p and from set R^{grid}_{slow} with probability $1 - p$. In Fig. 6 (Top), we summarize the average execution time obtained for $p = 1/2$ where we also manage to vary the number of clusters c in the range $\{1, 2, 4, 8, 16\}$ with equal number of nodes in each cluster. We can clearly see that LWS is able to obtain the best performance independently of the configuration and application benchmark. We also can see that RWS is the worst performing protocol which actually shows that link-heterogeneity can decrease performance dramatically if not handled accurately when balancing the load of such highly irregular benchmark applications. Besides, the rather simple PWS protocol is shown to perform consistently good with respect to the more sophisticated and platform-dependent CRS and ACRS algorithms. As it will be shown in the next paragraphs, PWS is however not a good option when handling a more stringe communication environment.

RESULTS ON 'VIRTUALIZED' FLAT ENVIRONMENTS (VF_ENV). Motivated by cloud, Internet, and peer-to-peer computing systems, we consider a network setting where no fixed hierarchy is set *a priori*. Instead, compute nodes are fully connected and the communication latency between each pair of nodes are randomly drawn from one of the two ranges $R^{flat}_{fast} = \{1, 3, 5, 10\}$ (ms) and $R^{flat}_{slow} = \{50, 100, 150, 200\}$ (ms) following a Bernoulli-distribution with parameter $p \in \{0, 0.25, 0.5, 0.75, 1\}$. Typically, the group of fast links is with respect to different clusters of different cities of the same country, or different workstations in distant countries of the same continent. The group of slow links represents typically inter-continent communication links. Notice that in such an environment, CRS and ACRS can not apply. In Fig. 6 (Bottom), we can see that LWS performs significantly better than PWS and RWS; achieving up to an acceleration factor of two. Actually, we see a significant impact of link latency when parameter p is in the range $\{0, 0.25\}$. This corresponds to roughly less than 25% of links being fast. Below that percentage, LWS suffers a deterioration in execution time for UTS while being relatively robust for B&B. Above that percentage, both LWS and PWS stabilize quickly with LWS being better. This indicates that LWS is able to schedule work steals efficiently by exploiting maximally the few fast links available in the network. Actually, by a more in-depth analysis, we are able to show that LWS is distributively constructing a kind of probabilistic network spanner [PS89] connecting nodes. Such a graph spanner is used to make work flow probabilistically in an efficient manner; which form both a theoretical and applied perspective suggests that defining and constructing probabilistic graph spanners is a good option for the purpose of dynamic load balancing. Interestingly, we found that that the communication structure implied implicitly by LWS between the different nodes has the very specific properties of being sparse and contains very few nodes with high in-degree modeling how much often a node shall steal from another one; thus improving work locality and optimizing the global cost of synchronous work transfers.

2.5 CONCLUSIONS AND PERSPECTIVES

SUMMARY. In this chapter, we presented our main contributions in designing parallel B&B while addressing two major challenges: dealing with the irregularity of B&B tree search, and dealing with the heterogeneity of the compute platform. From an optimization perspective, and although the designed algorithms are experimented using the permutational flowshop problem, there are thought to be as generic as possible in order to gain insights into the design of irregular tree-search algorithms in general. From a distributed and high performance perspective, the designed protocols are based on relatively advanced algorith-

mic paradigms, and technological and software considerations. As a by-product, novel dynamic load balancing distributed protocols based on hybrid and adaptive work stealing are derived and proved to be extremely efficient. For instance, our multi-core multi-CPU multi-GPU B&B approach can be considered as a state-of-the-art both in terms of parallel efficiency and distributed scalability.

The experimentation methodology that we adopted all along our research is also an important facet of our contribution. Since we are both concerned with optimization and high performance, a fair and reproducible experimental validation process was mandatory. On one side, we demonstrated that our approaches were competitive when compared to different well-established techniques for load-balancing that do not depend on the B&B algorithm that we are considering. This is for instance the motivation behind using the Unbalanced Tree Search benchmark. On the other side, we considered a broad range of of experimental distributed scenarios enabled either (i) through the use of a real test-bed with different configurations and different scales or (ii) the use of advanced emulated environments leading to a high degree of flexibility. This allowed us for instance to study the impact of network link heterogeneity while proposing a new generic protocol which is able to improve the performance of some state-of-the-art algorithms designed for specific compute environments. We in fact advocate for such a methodology, not only for the sake of reproducibility and fairness, but more importantly because it allows one to gain a more fundamental understanding of the main underlying challenges, both at the algorithmic and technological levels, and to push towards the design of innovative approaches and ideas to tackle them.

HIGH PERFORMANCE ... TO BE CONTINUED. From a high performance point of view, and despite the fact that the designed protocols are proved to be extremely efficient, the perspectives of the work described in this chapter are numerous. Firstly, different future developments can be considered with respect to the target compute environments. There is in fact a global race from both academia and industry in order to strengthen and to generalize the use, the development and the manufacturing of advanced computing products, services and technologies. For instance, virtualized environments and high performance cloud computing oriented (pay-as-you-go) facilities were shown to present a relevant and plausible alternative to classical and standard compute platforms such as in-door clusters and supercomputers. In such a setting, both the underlying networking and compute resources can be heterogenous and they might even be dynamic and/or not available to the final users. It is also with no-surprise that other hardwares and software components with new compute characteristics and abilities will continue to appear and to be integrated in existing compute platforms, which then implies to continue investigating how the existing techniques and algorithms can be adapted accordingly. This also implies that the optimization community should stay updated with respect to the rapid advances made by the distributed and high performance computing community, and should work actively in transferring and cross-fertilizing their respective expertise.

PARALLEL TREE SEARCH. There are also a number of other interesting perspectives of our work with respect to the design of novel parallel and distributed optimization algorithms in general. In fact, B&B is one particular tree-search algorithm using some specific policy to decompose the search space and to traverse it. Many other search algorithms are based on this decomposition principle and can be characterized by the dynamic and irregular nature of their workload, e.g., [FMM94; HW13; GK99]. We are also interested in other decomposition-based algorithms where parallelism and high performance computing are expected to play a crucial role as in parallel B&B. For instance, one contribution [DP15] (not described in this document) deals with the benchmarking of the so-called Simultaneous Optimistic Optimization (SOO) algorithm [Mun14], coming from the machine learning field, and tightly related to the DiRect (Dividing Rectangle) algorithm [JPS93]. In few words, the SOO algorithm is a global continuous optimizer which has theoretically provable perfor-

mance for functions being locally 'smooth' near their global optima but where the actual smoothness is not known. As B&B, SOO can be viewed as a divide-and-conquer tree search algorithm where tree nodes are dynamically mapped to cells representing decreasing continuous subdomains. Cells are selected and expanded dynamically in an iterative manner based on multi-armed bandit theory to (lower) bound the quality of already expanded cells. Deriving parallel SOO algorithms is one very challenging and promising research question, not only because the unpredictable workload of its computing intensive workflow, but also because parallelism opens an entire set of novel possibilities for designing new distributed and machine learning based *cooperative* policies for tree node selection.

CONNECTING EXACT AND HEURISTIC SEARCH. Finally, let us note that B&B as an *exact* optimization algorithm is different in nature from the other *heuristic* search algorithms that we shall consider in the rest of this document. Besides hybridization and other commonly known techniques used to speed up the search, there are a number of interesting and innovative ideas that one can consider to bridge the gap between exact and heuristic search algorithms in general. In fact, although several research works in connecting exact and heuristic search algorithms can be found, most efforts address them independently, and search strategies that are appropriate for one class of approaches are not directly applicable when the other class is considered. Very often, the argument for switching from exact to heuristics is only based on the instance size or computational complexity results. This is true to some extent; however, this division discourages cross-fertilization of algorithmic ideas developed for both methods. In deed, the idea would be rather to adopt a unified approach which is a requirement to efficiently and effectively tackle the optimization problems encountered in today's complex application domains. For that purpose, one interesting research path that we would like to investigate consists in (i) viewing/modeling an exact or heuristic procedure as exploring a possibly large neighborhood structure, e.g., partial trees in the case of B&B, (ii) characterizing what makes the exploration of such a neighborhood effective or not by designing high level features typically inspired by the so-called fitness landscape analysis, and (iii) use advanced machine learning techniques to both identify/predict the features that can play a role on the performance of different search strategies, and to adapt/configure the search accordingly before hand or at runtime. The establishment of such a methodology would represent a substantial advance in unifying exact and heuristic search algorithms, which we are actively working on in collaboration with our colleagues from the University of Coimbra, Portugal, with whom a bilateral project in this line was recently submitted.

3 DISTRIBUTED AND ADAPTIVE HEURISTIC OPTIMIZATION

In this chapter, we describe our main contributions on the design and analysis of high-level autonomous heuristic search algorithms. The main focus is on the adaptive operator selection problem tackled from a distributed perspective and using machine learning inspired techniques. The following aspects will be addressed:

- In the introductory section, we state the general motivations of our work and recall some background about online algorithm selection, and related paradigms such as offline tuning and hyperheuristics. In particular, some related work, focusing on existing adaptive operator selection techniques and the distributed island model, are discussed at the aim of illustrating the different challenges one has to consider.

- In Section 3.2, a distributed adaptive algorithm selection framework using both the island model and the Master/Worker paradigm is described. A brief analysis of its behavior and performance from a very abstract level is highlighted, i.e., unlike the previous chapter, we focus on studying the optimization and search ability of the proposed algorithms which is to be considered as a first step towards a practical deployment.

- In Section 3.3, a new model, based on Fitness Cloud, and allowing a more fundamental analysis of the effectiveness of the considered adaptive algorithm selection techniques is described. Some examples, to be considered as representative adversary optimization scenarios, are described and analyzed briefly in order to better illustrate what makes an adaptive selection technique relevant or not, especially when compared to a static offline strategy.
  This piece of research was mostly conducted within the PhD Thesis of Christopher Jankee under the supervision of my colleagues from the Univ. Littoral Côte d'Opale, Sébastien Verel and Cyril Fonlupt, .

- In Section 3.4, we complement the contributions presented in the previous section by discussing very briefly other tightly related research issues that we had the opportunity to study. More precisely, we highlight our work on the design of a high-level neighborhood tree search hyperheuristic, and our recent investigation on the development of a landscape-aware methodology for offline algorithm configuration.

RELATED PUBLICATIONS, PROJECTS AND COLLABORATORS.

COLLABORATORS. H. Aguirre, H. Derbel, C. Fonlupt, C. Jankee (PhD Supervision), A. Liefooghe, K. Tanaka, S. Verel

PUBLICATIONS. [Jan17; Lie+17b; Jan+16; Jan+15; Jan+17b; Jan+17a; DV11; DD12; Yah+15]

PROJECTS. S3-BBO Ayame/Inria associate team (2015-2017), JSPS-MEXT bilateral, France / Japan, project (2013-2016). (See Appendix A.3)

3.1 INTRODUCTION AND BACKGROUND

3.1.1 General context, motivations and goals

General Context

BLACKBOX OPTIMIZATION AND METAHEURISTICS. In this chapter, we consider the solving of blackbox optimization problems using general purpose search heuristics. Blackbox optimization refers to the situation where no problem-specific properties nor hypothesis can be known beforehand; and thus nothing but the fitness/objective values associated to a given (candidate) solution can be used by the optimization process. Solving a blackbox optimization problems consists in exploring a number of solutions by only evaluating their fitness values. Metaheuristics are general purpose heuristics that are designed to operate independently of a given particular problem, which makes them well suited to a blackbox optimization scenario. Among the large panorama of existing metaheuristics, we can distinguish those that handle a single solution at once in each iteration, such as, iterated local search (ILS) [LMS10], variable neighborhood search (VNS) [MH97], simulated annealing (SA) [KGV83], tabu search (TS) [Glo86], etc; and those that are based on a population (or a set) of solutions that are evolved at each iteration, such as genetic algorithms (GA) [Hol75; Gol89] and genetic programming (GP) [Koz90], and other (bio-inspired) and evolutionary algorithms (EA) such as ant-colony optimization (ACO) [Dor92], particle swarm optimization (PSO) [KE95] and model based heuristics such as Evolution Strategy [Rec73] and estimation of distribution algorithms (EDA) [LL01], to cite a few. Reviewing in details the design principles of such metaheuristics is out-of the scope of this document, and the reader is referred to existing surveys and books on the subject [GK06; Dre+06; Tal09; Jon06].

METAHEURISTICS WORKING PRINCIPLE AND LIMITATIONS. A metaheuristic can be viewed as an iterative search process, where in each iteration a number of solutions are newly generated using some *operators* that does not depend in general on the optimization problem it-self; but only, on the numerical representation of solutions. For example, a mutation/perturbation operator could generate a new solution by changing the value of some variables in a given solution to some other values, randomly or possibly following some probability distribution. In a permutation-based solution, a neighborhood operator can switch or shift some values in the permutation. In a binary solution, a crossover operator can be used to generate new solutions from some existing solutions by properly mixing their variable values, etc. There exist several kinds of operators that can be embedded within a metaheuristic in order to iteratively generate and explore new solutions. The fitness values of the solutions explored so-far, in the previous iterations, is then the only information that is used to guide the search. This is actually where different parameters and basic algorithmic components can eventually be embedded, e.g., different types of operators for solutions' generation, different types of acceptance criteria to define the search trajectory, different types of search memory, exploration rate, explicit diversity rate in a population or different mating selection and replacement policies, etc. Given the large spectrum of available metaheuristics and the corresponding components and parameters, it becomes more and more difficult, even for expert and well-trained algorithm designers, to identify the best possible choice when tackling a given instance of the same problem or when considering different problems coming from different application domains. We argue that this is actually one of the main limitations towards a more generalized use of metaheuristics and the related search paradigms.

General Motivations

ALGORITHM CONFIGURATION. In this chapter, we are concerned with the accurate configuration of general purpose search heuristics, with the main objective of gaining in generality and designing more powerful automated and autonomous search algorithms [HMS12]. It

is in fact well known that the performance of a heuristic search process, and any algorithm in general, heavily depends on the accurate choice of its algorithmic components and their respective parameters. Besides, from a theoretical point of view, the No Free Lunch theorem [WM97] demonstrates that no algorithm outperforms another one in average on the whole set of combinatorial optimization problems; hence, leading to the difficult challenge of designing an effective (autonomous) selection mechanism that is able to decide which algorithm to choose, or at least which parameters' values to set for a particular algorithm, and with respect to a given problem or instance. Notice that parameter setting can also be viewed as an algorithm selection problem when the number of parameters or components of the algorithm is a finite set. In this context, Rice [Ric76] was the first to propose an algorithm selection methodology based on feature extraction and on a per-instance selection principle. Such a methodology was extended in other studies [Smi09; ME14; Muñ+15]. For example, the work presented in [ME14] supports that by using the problem features, the performance of an algorithm can be predicted using some machine learning techniques and without effectively running the algorithm. With the increasing number of available algorithms, and the number of the possible components, the general problem of algorithm selection and configuration has become more and more popular, and is regularly tackled using different methodologies and techniques [Kot14; HMS12]. Instead of developing a new optimization algorithm, the design of a highly competitive algorithm turns out to the identification of the most suitable one among a portfolio of exiting algorithms or alternatively the most suitable combination/setting of basic components/parameters. This is in our opinion one of the major challenges that the optimization community has to face, and which constitutes the first motivation of our work.

DISTRIBUTED OPTIMIZATION. On the other hand, the ever-increasing demand in computing performance as well as the advent of new compute facilities and the establishment of robust and large scale massively parallel platforms, open tremendous research opportunities for pushing forward the development and uptake of metaheuristics and evolutionary optimization algorithms. In this context, a number of parallel and distributed optimization models and algorithms can be additionally considered. Consequently, further (cooperative) search strategies can be designed and additional algorithmic components and parameters can hence be integrated to optimally fit a distributed/parallel compute environment. This means that further design choices are available, more parameters are to be set, and thus, the process of deciding what is an optimal fit becomes even more complex and more challenging. This constitutes our second main motivation.

For completeness, let us recall that a relatively large number of studies exist on the design of parallel and distributed metaheuristics and optimization algorithms in general, e.g., [Sud15; Alb05; Tal09]. Although reviewing all the literature is out-of-the-scope of this document, let us comment that two main classes of approaches are usually distinguished: (i) those exposing low level parallelism with the main goal of providing a substantial parallel speed-up when solving an optimization problem without impacting the search process, typically when the evaluation function is costly, and (ii) those exposing high-level parallelism typically referring to the situation where multiple, possibly different, search processes are executed in cooperative and parallel manner. To some extent, our work can also be positioned in the second class of approaches with one main distinguishable feature. Our focus is on the design of algorithm selection mechanisms that are used to guide the distributed search by deciding adaptively on the accurate actions to be executed distributively. Moreover, our approach is of fundamental interest, in the sense, that we do not target a specific problem nor we consider specific compute environments. We rather attempt to gain a better understanding of the benefits of adaptive and autonomous search algorithms, by focusing on the design and analysis of novel distributed strategies. In particular, and in order to focus more deeply on the pure algorithmic and optimization challenges of setting a distributed approach to adaptive algorithm selection, we leave behind the scene the characteristics of

the intended compute platforms and the corresponding parallel and distributed implementation issues. We instead consider an abstract model where it is simply assumed that some computing nodes are available and can communicate using some abstract communication medium, typically by message passing.

General Goals

In the following paragraphs, we first recall a very general, and widely adopted, taxonomy for algorithm configuration and highlight our interests in the so-called on-line setting. We then state more explicitly the general challenges behind adopting a distributed approach and the target optimization scenario, namely adaptive operator selection.

ONLINE AND OFFLINE SELECTION. Two tightly related methodologies are commonly adopted for algorithm selection and parameter setting [HMS12; Eib+07; LLM07; Kra10]. In the *offline* setting, also called *tuning*, the algorithm is first selected and finely tuned, and only then it is executed from scratch on the target and unseen problem instance. Some methods use performance prediction methods based on problem features such as in SATzilla [Xu+08], and some others are based on searching in the space of possible algorithms or components or parameters, such as racing techniques [Bir+02; Bir09]. In the *online* setting, also called *control*, algorithm configuration is handled simultaneously with the optimization process, e.g., [Di +15; Fia+10a; BP14]). Different classes of approaches can be identified. Those using some *deterministic* policy to typically pre-schedule before-hand how the parameters will evolve all along the search process. Those that are based on an *adaptive* policy, that defines the set of components to be used according to some information acquired during the search. Those called *self-adaptive* that typically attach the parameters to the solutions being optimized and make them evolve as being intrinsically part of the optimization process. In this context, we are interested in adaptive selection approaches that consists in the online selection, among a number of alternatives in a portfolio given beforehand, of an appropriate choice to consider next in the search according to the current state of the optimization process. In other words, this consists in getting a continuous feedback from the optimization algorithm being executed, and deciding accordingly on the next choice. Generally speaking, this can also be viewed as an optimization process which follows the multi-armed bandit [KJ87] framework where the arms are the available alternatives (i.e., algorithms/components in the portfolio). The adaptive selection is then performed as follows. A reward is computed according to the performance observed in previous iterations. Then, a reinforcement machine learning is applied in order to decide which alternative to consider in the current iteration, typically according to some exploration-exploitation rules. Following this framework, several existing machine learning techniques have been used and studied in the past, e.g., [GLS16; Di +15; Fia+10a; DaC+08]. In the reminder, we mainly target an *online* setting and we essentially adopt a multi-armed bandit oriented methodology, although some of our recent contributions consider the offline setting as it will be highlighted in the very end of this chapter.

DISTRIBUTED *vs.* SEQUENTIAL SELECTION. On the other hand, we argue that the design and analysis of online selection strategies require a special attention when considering a distributed or parallel compute environment. In fact, specific distributed coordination and/or cooperation mechanisms between the set of available resources are required in order to achieve optimal performances, which raises a number of specific issues and challenges. For instance, in contrast to a sequential compute setting, one can benefit from the set of performances, and subsequently a set of rewards, observed by several processes running concurrently and not only a single one, which is a fundamental difference with a sequential design. One can adopt either a homogenous policy in which all distributed processes execute the same algorithm at each iteration or instead a heterogeneous policy where the processes are instructed to execute different algorithms; thus, allowing to pre-schedule different exploration-exploitation trade-offs. Additionally, it is not clear what distributed architecture

should be adopted, e.g., fully decentralized framework, Master/Worker, hierarchical, which then can imply different selection policies. Addressing such issues is precisely the main objective of the work described in this chapter.

OPTIMIZATION SCENARIO AND BENCHMARKING. The previously discussed aspects cannot be considered independently of the problem being tackled. In fact, although algorithm selection techniques aim at gaining in generality, their performance can still be impacted by the optimization scenario (e.g., the number of available algorithms, the scale of the distributed system, the type of components to choose, etc) and the characteristics of the considered problem instance. Firstly, most of our work falls in the framework of adaptive operator selection [GLS16; GL12; Fia+10a; Fia+09; DaC+08], where a portfolio containing combinatorial or evolutionary *operators* is assumed. We target the relatively simple, yet challenging, optimization setting where a number of (basic) operators are available and can be used during the search process. A noticeable feature of our work is however the nature of the distributed setting we are considering. Secondly, we mostly focus on common and widely used *abstract* benchmark functions, and we leave behind the scene the possible application to other more sophisticated or application oriented problems. In particular, we propose a new set of abstract benchmark instances, to be considered as a separate contribution in the sense that they aim to enable a more fundamental understanding of the expected behavior of the considered selection strategies.

In the following, and before moving to the description of our contributions, we shall first provide an overview of related work, by only focusing on the principled studies and approaches that are very tightly and directly related to our work, as well as on some exiting optimization benchmark problems. Our goal is not to cover the huge body of literature on algorithm selection and related aspects such as portfolio design, autonomous search, etc; but simply to provide the reader with the necessary background to better appreciate and position the main contributions discussed afterward in this chapter.

3.1.2 Adaptive Operator Selection: A focused Literature Overview

REINFORCEMENT LEARNING INSPIRED TECHNIQUES. A number of reinforcement machine learning techniques have been proposed for the online (adaptive) selection of operators. Back to the early works of Grefenstette [Gre86], one standard technique consists in predicting the performance of a set of operators using a simple linear regression, which then allows to select the best operator according to the prediction given by the regression. Recent works embeds selection into a multi-armed bandit framework dealing more explicitly with the tradeoff between the exploitation of the best so far identified algorithm, and the exploration of the remaining potentially under-estimated algorithms. The portfolio of available alternatives is then viewed as a set of arms, each one corresponding to one alternative that can be used at any iteration of the search. The optimization process has then to decide on the arm to be activated at each iteration. Once an arm is executed, a reward relating to the observed performance is computed, which constitutes the feedback that the optimization process gets at each iteration. The main difficulty is that the probability distribution of the reward with respect to an arm cannot be known in advance, and one has to make a guess on the best arm using a restricted number of samples. This corresponds to the situation where different (stochastic) evolutionary operator are available, but have an unknown and variable behavior when executed at some stage of the search. The main challenge is then to be able to define an accurate reward function and to decide online on the best arm or best sequence of arms to execute, knowing than the decision of executing an arm has typically the effect of modifying the current state of the search, e.g., the current solution, and hence to (unknown) current probability distribution of arms' rewards.

In this context, a simple selection strategy is the so-called ϵ-greedy strategy which consists in selecting the algorithm with the best estimated performance at rate $(1-\epsilon)$, and a random one at rate ϵ. More advanced strategies are however available as sketched in the following. The Upper Confidence Bound (UCB) strategy [ACF02] is a state-of-the-art framework in reinforcement machine learning which consists in estimating the upper confidence bound of the expected reward of each arm by $\hat{\mu}_i + C \cdot e_i$, where $\hat{\mu}_i$ is the estimated (empirical) mean reward of arm i, e_i is the standard error of the prediction and C a parameter allowing to tune the exploitation/exploration trade-off. It then selects the algorithms with the higher bound (for maximization problem). In our setting, where the arms (i.e., the algorithm/operator to select) could be neither independent nor stationary, the estimation of the expected reward can be refined using a sliding window where only the W previous performance observations are considered [Fia+10a]. The Adaptive Pursuit (AP) strategy [Thi05] is another technique using an exponential recency weighted average to estimate the expected reward, and based on a parameter α to tune the adaptation rate of the estimation. This is used to define the probability of selecting every algorithm from the portfolio. At each iteration, these probability values are updated according to a learning rate β, which basically allows to progressively increase the selection probability for the best algorithm, and to decrease it for the other ones.

Due to their generality, the before-mentioned techniques are very often used in different optimization and application contexts and adapted accordingly, e.g., [Wu+16; BP14; Li+02; VHS13; VMS12]. However, one key aspect for a successful application is the estimation of the expected quality of an algorithm according to the rewards computed at the previous iterations. Some authors showed that the maximum reward over a sliding window can improve the performance compared to the mean on some combinatorial problems [Fia+10a; Can+13]. In genetic algorithms, the reward can be computed not only based on the quality but also on the diversity of the population [Mat+09]. In other studies [Fia+10b], a rank based approach is adopted in order to avoid performance normalization issues. The choice of an accurate reward is in fact crucially important since it guides the whole selection process.

HYPERHEURISTICS. Another principled and very tightly related methodology to algorithm selection is based on the so-called Hyperheuristic search paradigm. A hyperheuristic [Bur+10] can be considered as a high-level search algorithm operating in the heuristic space, and implying a (hyper-)search strategy that can autonomously combine different available low-level heuristics considered as building blocks. Hyperheuristics can be classified from different orthogonal perspectives [Bur+10; Bur+13], e.g., whether their design involves learning or not, whether their are online or offline, etc. In particular, unlike generative hyperheuristics where one aims at generating new and more complex heuristics, *Selective* hyperheuristics allows one to select and to combine different low-levels heuristics adaptively during the search. In [CKS01; ÖBK08] for instance, low-level heuristics (e.g., operators/neighborhoods) are chosen either greedily or probabilistically, based on scoring and ranking functions that take into account several criteria such as the time it takes to run a low-level heuristic, its observed performance, etc. Other strategies inspired by the way metaheuristics operate can also be found in the literature, see e.g., [CC08].

DISTRIBUTED (ISLAND-BASED) TECHNIQUES. In the distributed setting, relatively few studies can be found on the design of adaptive selection approaches compared to the sequential setting. Nonetheless, the subject is not new and has been addressed implicitly or explicitly in the past, e.g., [Can98; Can+12; Bia+09; RGK05; DB13; PDB12]. It is actually the case of the so-called *Island* model which was extensively studied in the past [Tom05]. The island model is considered as inherently distributed and can be viewed as a suitable model to tackle the algorithm selection problem from a fully decentralized perspective. In its very basic form, different entities called islands are organized according to some graph structure, where each island is a node in the graph and two neighboring islands, linked by an edge,

can directly cooperate together. From a distributed or parallel perspective, an island can for instance be associated to a distributed process and the topology defining the neighboring islands can be viewed as the communication pattern underlying the distributed communication protocol. There are several ways to design an optimization algorithm following the island model; depending on the size (granularity) of the population handled at each island, the algorithm executed locally at each island, and the cooperation strategy such as the migration policy which defines how solutions are exchanged between neighboring islands. In particular, in the so-called heterogeneous island model, every island can execute a different algorithm than the other islands in the system, which makes it very tightly related to the design of distributed algorithm selection strategies. There are basically two classes of parameters that can be controlled in a (distributed) island model: the parameters related to the migration policy, and the parameters that define the algorithm to execute at each node. For instance, in [Sor+15; Can+12; GL12; GL11], a heterogeneous island model where each island can apply its own evolutionary operator and where the migration policy is controlled online is considered. More specifically, a probability matrix is maintained online in order to control the migration rate between islands. According to the performance of each island in producing promising solutions when applying the corresponding operator, the rates are in fact updated using a reinforcement learning principle. In [Fer+14], an approach to control the migration policy is also proposed when the population is spatially structured following a 2d-grid by moving solution either randomly or according to solutions' similarity. Several other investigations can actually be found in order to demonstrate the benefits of controlling the algorithms being executed by each island, e.g., [GF11; TF13; Gar+14; TC02]. It is in general shown that using a collection of different parameters, algorithms, etc, distributed over the collection of islands, provides better performance compared to a uniform static setting.

In our work, we consider to explicitly leverage the existing (machine learning based) sequential techniques sketched previously, to the distributed setting. We first consider the island model as a suitable tool to design the target distributed adaptive algorithm selection strategies. We also consider a more simple model based on a Master/Worker architecture. The Master/Worker (M/W) architecture has been extensively considered in the past due to its simplicity and effectiveness. However, we are not aware of in-depth studies addressing the design principles underlying a M/W adaptive algorithm selection approach. Both for the island-based and the M/W approaches, we empirically analyze the behavior of different selection strategies by adopting a simulation based methodology. As mentioned before, taking into account the implementation and technical issues arising when effectively implementing a distributed framework in a real compute environment are left as future work. In this respect, our work is to be viewed both as an attempt to fill the gap between distributed algorithm selection and the well-established existing sequential approaches, as well as a first step towards their effective deployment.

3.1.3 Benchmarking Operator Selection

Generally speaking, the understanding of the performance and dynamics of an algorithm selection strategy is a difficult issue, which motivated the establishment of a number benchmark functions and libraries [Bis+16]. Existing work from the literature considered different case studies ranging from simple abstract problems, to more concrete complex problems, possibly coming from different domains and applications, e.g., SAT, constraints satisfaction, scheduling, planning, bin packing. In the context of the *operator* selection problem, we follow the existing literature, and we there-by focus on simple combinatorial benchmark functions and evolutionary algorithms, constituting a simplification of what could occur in practice when solving a complex optimization problem. This is in an attempt to gain insights into the fundamental behavior of the considered distributed and adaptive algorithm selection techniques.

Firstly, we focus on a basic optimization setting (e.g., [DaC+08; Fia+09]) where an algorithm from the portfolio consists in the application of one iteration of a standard $(1+\lambda)$ evolutionary algorithm. In other words, given a current solution, λ new solutions (or offsprings, neighbors) are generated using an operator selected before-hand from the portfolio, and only the best solution (having the best fitness value) among the $(1+\lambda)$ ones survives in the next iterations. This is to be considered as a basic building-block that can be used in more sophisticated optimization algorithms.

Secondly, we focus on different abstract and tunable benchmark problems. This is a very critical aspect since the performance a given approach can only be understood in light of the properties of the considered problem instances. Two classes of benchmarks can be considered. In the first class, standard (blackbox) optimization problems are used. This however can only highlight the search behavior according to the properties of those considered problems. In the second class [GLS16; Thi05; Fia+10a], the problem and the stochastic operators are abstracted and the performance of each available operator is then defined according to the current state of the search. This has the advantage of studying important features such as the number of operators, the frequency of change of the best operators, the quality difference between operators, etc. In the following paragraphs, we recall the benchmarks that we pick from the first class, namely, the so-called OneMax and NK-landscapes. The benchmarks used from the second class are based on the so-called Fitness Cloud Model and are described later in this chapter as a separate contribution.

ONEMAX. The OneMax problem, the "drosophila" of evolutionary computation, was extensively used within the framework of adaptive operator selection [Fia+09]. Besides, different analytic studies on the behavior of standard evolutionary operators were conducted using the OneMax problem [Chi+15; DD14; GL11]. It is thus well suited to understand the performance and the dynamics of the considered adaptive approaches. OneMax is a unimodal problem defined on binary strings of size N. The fitness is simply the number of 1s in the bit-string, i.e., given a bit-string $x \in \{0,1\}^N$, $f(x) = \sum_{i=1}^{N} x_i$. The optimum is obviously trivial and its fitness is N.

Following the literature, we consider a portfolio composed of standard c-bit-flip and k-bit mutation operators. The c-bit-flip operator flips each bit with probability c/N where c is a parameter. The k-bit operator flips exactly k different bits chosen at random. In order to give an intuitive idea about the behavior of such operators, we depict in Fig. 7 the expected improvement in the fitness value (y-axis) when applying one iteration of a $(1+\lambda)$-EA using different operators on a solution of a given fitness (x-axis) [GL11]. In the top left subfigure, we can see that 1-bit-flip allows to obtain a slightly better improvement over the 1-bit operator, which is due to the fact that the 1-bit-flip is able to mutate more than one bit with a non-zero probability. Despite that the difference is small, it could significantly impact performance when one has the possibility to choose among the two operators all along the search. In the top right subfigure, the difference in the fitness improvement obtained by 4 operators suggest that there is an optimal sequence in which the operators have to be selected and applied successively from the portfolio in order to reach a high quality solution in minimum time. In the bottom subfigures, we can also see the impact of parameter λ on the fitness improvements. For instance, this should give the reader a hint on how a selection strategy can perform when several $(1+\lambda)$-EAs are executed by a variable number of distributed processes.

NK-LANDSCAPES. The family of NK-landscapes constitutes a more sophisticated model of multimodal problems [Kau93] that are proved to be NP-complete. As for the OneMax, the search space is binary strings of size N: $\{0,1\}^N$. N refers to the problem size, and K to

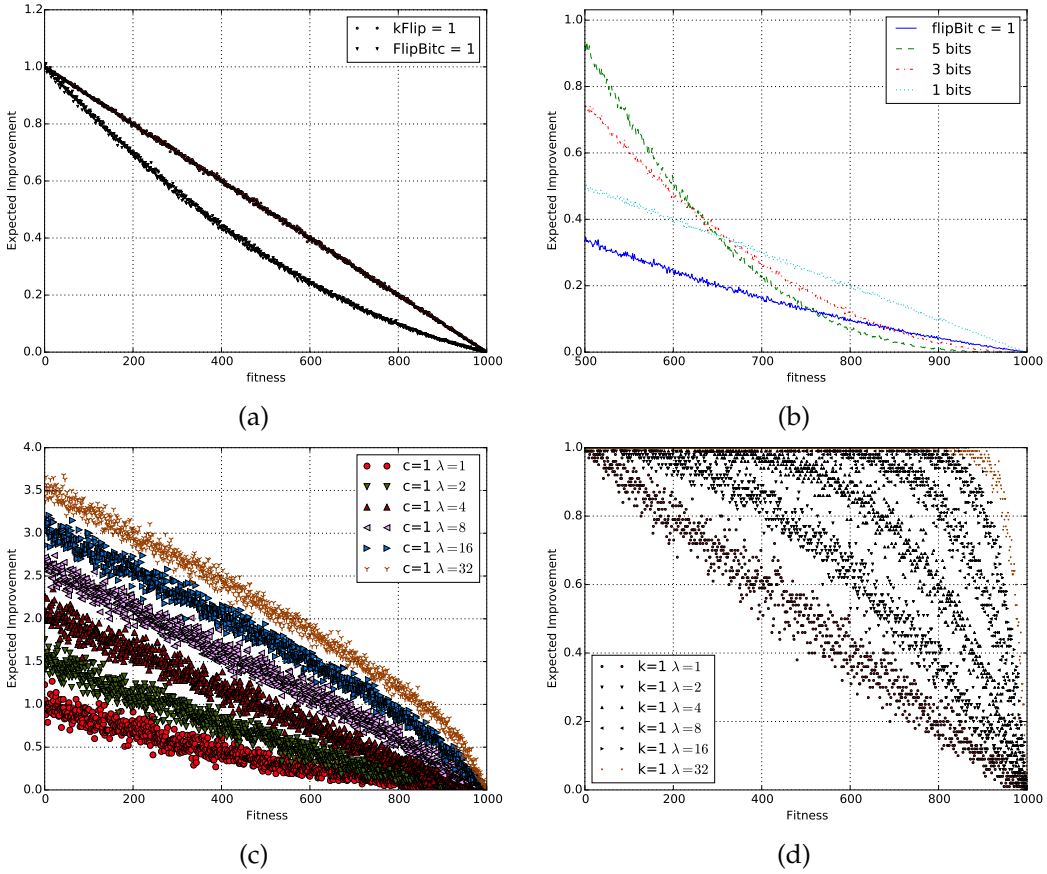


Figure 7: Expected improvement for the OneMax problem [GL11] for a bit-string of size $N = 1000$. (a) relative performance of the 1-bit-flip and the 1-bit operators with $\lambda = 1$. (b) relative performance of the 1-bit-flip, 1-bit, 3-bit and 5-bit operators with $\lambda = 1$. (c) relative performance of the 1-bit-flip operator using different values of λ. (d) relative performance of the 1-bit operator using different values of λ.

the number of bits that influence a particular position from the bit-string, i.e., the epistatic interactions. The objective function $f : \{0,1\}^N \to [0,1)$ to be maximized is defined as follows.

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots, x_{i_K})$$

where $f_i : \{0,1\}^{K+1} \to [0,1)$ defines the component function associated with each bit x_i. By increasing the number of epistatic interactions K from 0 to $(N-1)$, NK-landscapes can be gradually tuned from smooth to rugged and highly non-linear. The position of these interactions are set at random and the component values are uniformly distributed in the range $[0,1)$. Of course, the interaction between variable is not available in a blackbox optimization scenario. Our interest in the NK-landscapes stems from the fact that different bit-flip mutation rates are believed to provide different performances. To illustrate this claim, we show in Fig. 8, the empirical probability to improve a solution with a given fitness and using the c-bit-flip for different values of parameter c. We can clearly see that the operator which is likely to provide an improvement depends strongly of the attained fitness level, which makes NK-landscapes both challenging to solve by a blackbox optimizer and interesting to study and analyze.

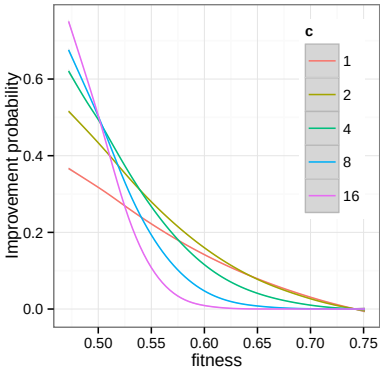


Figure 8: Empirical improvement probabilities *vs.* fitness level for a NK-landscape with $K = 4$ and different c-bit-flip operators $c \in \{1, 2, 4, 8, 16\}$.

Algorithm 3: Example of a distributed oracle strategy

1 Generate a random solution on each node N_i with $i \in \{1, \cdots, 4\}$;
2 Run M_3 in parallel on each N_i with $i \in \{1, \cdots, 4\}$;
3 Exchange best found solution;
4 Run M_1 in parallel on each N_i with $i \in \{1, 2\}$;
5 Run M_2 in parallel on each N_i with $i \in \{3, 4\}$;
6 Exchange best found solution;
7 Run M_4 in parallel on each N_i with $i \in \{1, \cdots, 4\}$;
8 Return best found solution;

3.2 CONTRIBUTION #1: DISTRIBUTED ADAPTIVE OPERATOR SELECTION

The first contribution described in this chapter is on the design and analysis of a distributed framework for adaptive algorithm selection. Let us assume we are given a set of computational resources that can for instance be distributed over a network and exchanging messages, or some parallel processors having some shared memory to communicate. As an illustration, let us consider the following simple artificial example. Assume we have 4 distributed nodes $\{N_1, \ldots, N_4\}$ organized as depicted in Fig 9 and a portfolio of 4 metaheuristics $\{M_1, \ldots, M_4\}$. Our goal is to design a distributed strategy that decides how to distribute the execution of these metaheuristics over the available nodes and what kind of information should be exchanged by computational nodes to obtain the best possible solution in least time. To make it simple, assume that the best (oracle) strategy is actually given by the template of Algorithm 3 and illustrated in Fig. 9, i.e., any other distributed strategy cannot outperform Algorithm 3. The question we are trying to answer is then: How can we design a distributed strategy which is competitive compared to the (unknown) distributed oracle? Finding such a strategy is obviously a difficult task. In fact, finding the best mapping of our metaheuristics into the distributed environment is in itself an optimization problem which could be even harder than the initial optimization problem we are trying to solve.

In this context, we propose a distributed adaptive metaheuristic selection (DAMS) framework [DV11; Jan17] allowing us to leverage previous sequential techniques. In the remainder, we assume that the distributed resources can communicate together following an island model, while abstracting away the physical communication layer and the underlying technical implementation issues. We first discuss the DAMS framework and a simple adaptive selection strategy called SBM. Alternative independent and collective selection strategies

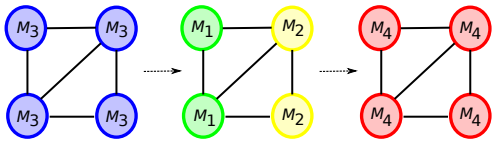


Figure 9: Illustration of the distributed oracle strategy of algorithm 3.

based on classical multi-arms bandits are then discussed, and a simple Master/Worker alternative model is also described [Jan+17b; Jan+17a; Jan17].

3.2.1 DAMS and Select-Best-and-Mutate strategy

The DAMS framework, initially presented in [DV11] and then extended in [Jan+17b; Jan+17a; Jan+15; Jan17], is basically a heterogeneous island-based framework that allows computational nodes to coordinate their actions distributively in an online fashion. In Algorithm 4, we sketch the high-level code to be executed in parallel by every distributed node. At each iteration, or distributed round, a different metaheuristic (or algorithm or operator) from a portfolio can be applied to the subpopulation maintained locally at every node, and the metaheuristic could be different from one node to another. We distinguish three basic levels that can be controlled during one round of a DAMS algorithm: the distributed, the selection and the atomic levels. At the distributed level, information between neighboring nodes are exchanged, migration of solutions is achieved, and the reward with respect to the previously executed metaheuristics are eventually communicated. At the metaheuristic selection level, which is our main focus, one metaheuristic is selected from the portfolio according to the previously collected rewards. At the last level, the selected metaheuristic is applied and a corresponding (local) reward is computed.

A simple policy that we can adopt at the algorithm selection level is the so-called Select-Best-and-Mutate (SBM) strategy [DV11]. With probability $1 - p_{mut}$, SBM selects the metaheuristic having the best reward in the last round among all neighbors (including the current node), and with a mutation rate p_{mut}, SBM selects one random metaheuristic from the portfolio while excluding the best (lastly rewarded) one. In others words, SBM has an intensification component that selects the best rewarded metaheuristics among the ones selected at the previous round by the neighboring nodes, and a diversification component that allows to explore new randomly selected metaheuristics. In SBM, the reward is the maximum reward observed in the last round in the actual node and its neighbors. In other words, the maximum reward is estimated using solely the neighboring nodes and no long-term memory is used to store the rewards from the previous rounds.

As an illustration, we show in Fig. 10 the behavior of the SBM strategy using a fully connected island model. For simplicity only two operators (using the OneMax problem) are considered, namely 1-bit and 4-bits. In the right subfigure, we consider the expected improvement of the best solution obtained with SBM (circled gray points), compared to the one obtained when applying either the 1-bit operator (triangle red points) or the 4-bit operator (square green points) all over the nodes. We can see that the performance of the two operators (in terms of the expected fitness improvement, right subfigure) decreases when the global optimum is approached. More importantly, depending on the actual value of the current best in the system, one should alternatively select either the 1-bit or the 4-bit operator, and the SBM strategy seems to correctly handle this issue.

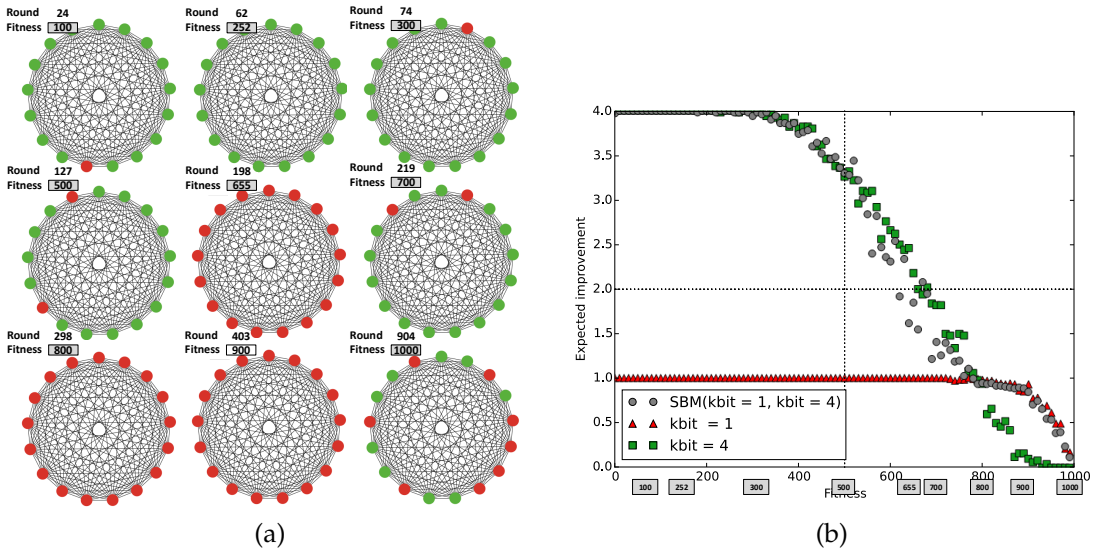
3.2.2 Independent and Collective Machine Learning based Strategies

The (maximum) reward considered at the selection level of SBM has a *collective* nature since every distributed node needs to know the local rewards of neighbors computed at the low

Algorithm 4: DAMS algorithm for each computation node

Inputs: A portfolio of metaheuristics $\mathcal{M}$

```
1  P ← INIT_POP() M ← INIT_META(M); r ← INIT_REWARD() ;
2  repeat
3      /* Distributed Level: migration and information sharing */
4      Send Msg(r, M, P) to each neighbor ;
5      P ← {} ; S ← {} ;
6      for each neighbor w do
7          Receive Msg(r', M', P') from w ;
8          P ← P ∪ {P'} ;
9          S ← S ∪ {(r', M')} ;
10     P ← UPDATE_POPULATION(P, P) ;
11     /* Metaheuristic Selection Level */
12     M ← SELECT_META(M, (r, M), S);
13     /* Atomic Low Level: apply metaheuristic and compute reward */
14     P_new ← APPLY(M, P) ;
15     r     ← REWARD(P, P_new) ;
16     P     ← P_new ;
17 until Stopping condition is satisfied;
```



(a) (b)

Figure 10: Illustration of a run of the DAMS-SBM strategy for OneMax (N = 1000) and a portfolio of 2 operators.

level after executing the previously selected metaheuristics. One could however remark that instead of selecting the best rewarded metaheuristic from neighboring nodes, each distributed node can *independently* select the best rewarded metaheuristic, eventually using solely a sliding window of size W, as usually done in sequential machine learning based strategies. Thus, by considering that the rewards are computed and maintained in an *independent* manner by every distributed node, several other techniques from the literature can actually be used in a rather straightforward manner. For example, a variation of SBM can be the well-known ϵ-greedy strategy in multi-armed bandits, which selects the arm (algorithm) with the highest estimated expectation with rate $1-\epsilon$, and uniformly at random an arm with rate ϵ. Accordingly, the DAMS framework can be enriched with other more

sophisticated (independent and collective) approaches based on Adaptive Pursuit (AP) and Upper Confidence Bound (UCB) algorithms, as briefly discussed in the following.

INDEPENDENT SELECTION STRATEGIES. Using AP, a metaheuristic i is selected by distributed node j, at each distributed round t, with probability $p_{i,j,t}$, and the probabilities are updated according to the reward computed locally at the low atomic level of node j. Technically speaking, AP is divided into three parts: the update of the reward estimation $\hat{q}_{i,t,j}$ of the metaheuristics, the update of the probabilities $p_{i,j,t}$, and the selection of the metaheuristic. The following equation defines how the reward of a metaheuristic i can be updated by node j, i.e., variable $r_{i,j,t}$ is the reward at round t of the metaheuristic i with respect to node j, and parameter $\alpha \in (0, 1]$ is the adaptation rate.

$$\hat{q}_{i,j,t+1} = \hat{q}_{i,j,t} + \alpha \,.\, (r_{i,j,t} - \hat{q}_{i,j,t})$$

The update of the probabilities $p_{i,j,t}$ is as follows (where i^*_t denotes the metaheuristic with the best $\hat{q}_{i,j,t}$):

$$p_{i,j,t+1} = \begin{cases} p_{i,j,t} + \beta \,.\, (p_{max} - p_{i,j,t}), \text{ if } i = i^*_t \\ p_{i,j,t} + \beta \,.\, (p_{min} - p_{i,j,t}), \text{ otherwise.} \end{cases}$$

Similarly, several Upper Confidence Bound (UCB) algorithms from the literature [Fia+10a] can also be applied. Let $n_{i,j,t}$ be the number of times the i^{th} metaheuristic is applied up to round t by a given distributed node j, and let $\hat{q}_{i,j,t}$ denotes the average empirical reward of metaheuristic i at that node. At each round t, a distributed node j selects the metaheuristic that maximizes the following quantity:

$$\hat{q}_{i,j,t} + C \,.\, \sqrt{\frac{2 \log(\sum_{\ell} n_{\ell,j,t})}{n_{i,j,t}}}$$

where parameter C enables to control the exploitation/exploration trade-off. Notice that the UCB strategy is an optimal strategy for stationary problems with independent arms which is actually not the case when dealing with our algorithm selection setting. In fact, the average empirical reward could be far from the current new reward, since the current solution is being optimized by the search process. To overcome this drawback, the average empirical reward can be computed over a sliding window by considering the last W rounds. We also consider a dynamic version of UCB introduced in [Fia+10a] and using the Page-Hinkley (PH) test to detect whether the empirical rewards collected for the best metaheuristic have changed significantly (which comes at the price of using two additional parameters, a restart threshold γ and a robustness threshold δ).

COLLECTIVE SELECTION STRATEGIES It should now be clear how to design other *collective* DAMS variants by simply injecting the rewards from neighboring nodes in the previous equations. For example, in collective AP, the rewards collected by a node from its neighbors during the previous distributed round can be iteratively used to update the reward estimation at that node. In the collective versions of UCB strategies, the empirical average can also be updated using the rewards from neighboring nodes and the numbers of times $n_{i,j,t}$ that each metaheuristic i is applied by a node j can also updated accordingly. This kind of information can be exchanged locally between node at the distributed level and used subsequently at the selection level of DAMS. As it will be argued later, collective strategies should in general be preferred over independent strategies.

3.2.3 A simple Master/Worker Architecture

The DAMS framework was initially thought to operate in an island like distributed model, which makes it general enough to leverage several existing techniques. However, its analysis and deployment could be challenging in practice due to the interdependency of the

Algorithm 5: Adaptive M/W algorithm for the master node

```
(θ^1, θ^2, ..., θ^n) ←Selection_Strategy_Initialization();
x* ← Solution_Initialization() ; f* ← f(x*);
repeat
    for each worker i do
        Send Msg(θ^i, x*, f*) to worker i;
    Wait until all messages are received from all workers;
    for each worker i do
        (r^i, x^i, f^i) ← Receive Msg() from worker i;
    x* ← x^i; f* ← f^i s.t. f^i = max{f*, f^1, f^2, ..., f^n};
    (R_1, R_2, ..., R_k) ← Reward_Aggregation((θ^1, r^1), ..., (θ^n, r^n));
    (θ^1, θ^2, ..., θ^n) ← Selection_Strategy(R_1, R_2, ..., R_k);
until stopping criterion is satisfied;
```

local decisions that are made locally by every distributed node and the complex behavior that can emerge form such a system. We there-by consider an alternative simple architecture [Jan+17a; Jan+17b] based on the Master/Worker model.

OVERALL DESIGN PRINCIPLES. The proposed architecture is sketched in Algorithm 5 depicting the high-level code executed by the master. One can remark that the three levels (distributed, selection, and atomic) designed for DAMS are re-visited to fit into the M/W model. More specifically, the M/W framework operates in different parallel rounds. At each round, the master sends an initial solution and algorithm (θ^i) to be executed by each worker i. The role of each worker is to compute a new candidate solution to be send back to the master. In addition, every worker computes a local reward in order to render the quality of its assigned algorithm θ^i. Different kinds of local rewards can be considered at this stage. In our work, we focus on elitist evolutionary algorithms to be selected and applied by the workers. Hence, the best solution is always sent to the workers and the local reward computed by a single worker is the positive improvement observed when applying the selected algorithm. The master then waits for all local solutions computed in parallel by the workers, and updates the best solution $x^\star$ to be considered in the next round, and so on. More importantly, the local rewards collected by the master are used in order to select a new set of algorithms to be assigned to the workers in the subsequent rounds, which actually constitutes the adaptive and core part of our proposal. Two tightly coupled issues are to be handled by the master in order to set up an effective adaptive mechanism: (i) how to aggregate the local rewards sent by the workers and (ii) how to select the new set of operators accordingly. This is sketched in the next paragraphs.

LOCAL REWARD AGGREGATION. All adaptive operator selection strategies such as ϵ-greedy, Adaptive Pursuit, Upper Confidence Bound, etc (see Section 3.1.2), need to get one single reward value as a feedback. In our distributed setting, a set of local rewards are computed by the distributed nodes (the workers). Unlike sequential algorithms, the set of local rewards observed in parallel cannot be viewed simply as a sequence of independent rewards that would be given iteratively to a sequential strategy. This holds for the DAMS framework discussed previously and is now made more explicit since only the master is allowed to execute the selection step. Hence, one specific design component of our M/W framework is the way to aggregate the local reward values into one global reward value. We distinguish two main aggregation strategies: (i) the mean or the (ii) maximum of the local rewards, possibly computed over a sliding window of size W. Despite their simplicity, the two previous local reward aggregation strategies are fundamentally different. For instance, assuming that the fitness improvement after applying a stochastic operator is given by a

probability distribution, the mean of the reward values computed by the workers allows to estimate the expectation of this distribution with a high accuracy, whereas the maximum gives information on its extremes.

HOMOGENEOUS *VS.* HETEROGENEOUS ADAPTIVE SELECTION. As mentioned previously, the master needs to select one operator for each worker. We consider both (i) a *Homogeneous* (Ho) adaptive strategy, in which the same operator is selected by the master and assigned to all workers, and (ii) a *Heterogeneous* (He) adaptive strategy, in which the master selects, possibly different operators to be assigned to the workers. The rationale behind a homogeneous strategy is that in each round there exists one relevant operator providing an optimal performance, and hence should be executed simultaneously in parallel by all workers. This is a rather exploitation-guided strategy which aims at avoiding to loose function evaluations, and to post-pone the exploration component to act in-between two consecutive rounds. In contrast, the rationale behind a heterogeneous strategy is that a set containing a mixture of different operators is expected to perform better than a set containing the same operator, in the sense that: (i) the probability of obtaining a better solution when executing different operators in each round is larger, and/or (ii) a relatively small number of evaluations spent exploring non-necessarily optimal operator(s) at each round allows to better predict the best operator(s) to select next.

In the homogeneous setting, we consider three standard machine learning based selection strategies, namely, ϵ-greedy, AP, and UCB. The same operator computed by any of these strategies is assigned by the master to the workers. Notice that the difference with a sequential selection is the way the reward is computed by the workers and maintained by the master, which is crucially important for those methods to operate accurately. In the heterogeneous setting, we consider to execute either the ϵ-greedy strategy or the AP strategy iteratively for each worker.

ADAPTIVE BATCH SCHEDULING. Finally, we consider the situation where the communication cost is non-negligible compared to the time a distributed node takes to execute the optimization step. For this purpose, we propose to endow our M/W framework with an adaptive batch scheduling mechanism allowing the master to select and send a whole bench of algorithms to be executed iteratively in a sequence by every worker [Jan+17a] (instead of executing just a single one). This is achieved by simply extending the previous homogenous and heterogeneous strategies to handle a variable number of selected algorithms for every worker. Notice that addressing batch scheduling using machine learning techniques is a challenging issue [DKB14]. In this context, our work is to be viewed as a first step allowing to better understand the different trade-offs that can be attained in terms of communication cost and solution quality when applying a distributed adaptive selection strategy.

3.2.4 Selected Experimental Results

The selection strategies designed in our work were systematically analyzed under different configurations in order to gain a better understanding of their behavior. In the following, we only highlight some illustrative results using the DAMS island based framework.

DAMS SETUP. We use an elitist migration policy. Each node (island) sends its current solution to its neighboring nodes. The best solution from the set containing the received solutions and the current solution of the node replaces the current solution. The DAMS algorithm stops when the global maximum is found by one node, or when the number of distributed rounds exceeds a fixed limit (i.e., a synchronous distributed model is assumed). The performance of algorithms is measured either by considering the number of rounds to reach the global maximum (for OneMax), or using the so-called expected running time (ERT) (for NK landscapes). The ERT is the expected running time to reach a fixed fitness with a simulated restart. It is given by: $E_s[T] + (1 - \hat{p}_s)/\hat{p}_s.T_{limit}$ where $\hat{p}_s$ the estimated success

Table 1: One-Max problem with $N = 1000$. Rank of each strategy. For each topology and graph size, a cell depicts the number of other selection strategies which statistically outperforms (according to the Wilcoxon test at confidence level $p = 0.05$) the strategy considered in the column. The o value is the best one: no other strategy significantly outperforms the considered one.

Topo.	Size	cst	rand	SBMi	SBMc	APi	APc	UCB					
								UCBi	UCBc	HPi	HPc	Wi	Wc
circle	4	8	4	1	0	7	7	10	11	2	3	3	3
circle	16	4	6	3	0	4	0	10	11	1	6	6	6
circle	32	4	6	3	1	4	0	10	11	2	6	6	9
circle	64	4	6	3	2	4	0	10	11	1	6	6	9
grid	4	8	4	1	0	4	9	10	11	2	4	3	3
grid	16	4	5	2	0	4	0	10	11	1	4	6	4
grid	32	4	5	3	1	4	0	10	11	1	4	4	6
grid	64	4	6	3	1	4	0	10	11	1	6	6	9
rnd.	4	7	3	0	0	5	7	10	11	0	3	3	3
rnd.	16	4	4	1	0	4	3	10	11	1	4	4	5
rnd.	32	4	4	3	1	4	0	10	11	2	4	4	9
rnd.	64	4	4	3	1	4	0	10	11	1	4	4	9
compl.	4	7	3	1	0	7	7	10	10	2	3	3	3
compl.	16	6	3	1	0	5	6	11	10	1	4	3	9
compl.	32	3	3	2	0	3	8	11	10	1	3	3	9
compl.	64	3	3	2	0	3	3	11	10	1	3	7	9
Average		4.875	4.312	2	0.4375	4.375	3.125	10.187	10.75	1.25	4.187	4.437	6.562

rate, and $E_s[T]$ is the average number of rounds when the fitness level is reached. Four island topologies are considered in the following: the complete topology where each node is connected to all others nodes, a random topology where there is an edge between two nodes with probability parameter ($p = 0.1$), the grid topology which is a two-dimensional regular square where each node is connected to the four nearest neighbors, and the circle topology where every node is connected to two others nodes to form a circle. The size of the networks is set to $n \in \{4, 16, 32, 64\}$. In order to have the same number of fitness evaluations in one round whatever the network size n, the λ parameter used in the $(1+\lambda)$-EAs is set to $64/n$. The independent strategies (i.e., no reward sharing) are highlighted with letter i and collective ones with letter c. When a window is used to compute the rewards (i.e., for UCB), letter W is used. Moreover, some baseline strategies are used: the random strategy (rnd.) where each node selects independently at random one metaheuristic to execute at each round, the constant strategy (cst.) where each node selects independently at random one metaheuristics to execute during the whole algorithm, and the uniform constant strategy (unif.) where all nodes execute the same metaheuristic during the whole algorithm (only the results obtained with the best performing metaheuristic are presented).

ONEMAX. In Table 1, we show the relative behavior of different strategies experimented with a portfolio composed by four $(1+\lambda)$-EA based on 1-bit-flip, 1-bit, 3-bit and 5-bit operators[1]. First, the performance of the different strategies are consistent with the considered network configurations in the sense that they can overall be ranked similarly independently of the topology type or graph size. More importantly, we remark that the impact of exchanging reward information (collective strategies) between nodes has a strong impact on performance. Interestingly, the impact is positive in the case of SBM and AP, whereas it is not when considering UCB. In fact, SBMc appears to overall outperform all the other strategies and APc appears to performing best when both considering the circle, grid and random topologies with large number of nodes. In contrast, the performance of the four implemented versions of UCB is deteriorating systematically as the information from neighbors is incorporated. This also suggests that the UCB strategy has to be carefully rethought in order to infer accurate exploration-exploitation tradeoffs in the distributed setting. We shall see in the rest of this chapter that other variants using UCB and a simple Master/-Worker model are actually very efficient.

1 This is a standard experimental setting from the literature.

Table 2: NK-landscapes with N = 1000 and K = 1, 4, 8. Rank of the different strategies according to the topology and the number of nodes.

Topo.	Size	unif	cst	rand.	SBMi	SBMc	APi	APc	UCB					
									UCBi	UCBc	HPi	HPc	Wi	Wc
		K = 1												
compl.	16	0	9	6	3	7	12	2	1	10	4	5	8	11
compl.	64	0	10	7	6	1	4	9	12	3	2	5	11	8
circle	16	0	2	11	3	5	1	10	8	7	6	4	12	9
circle	64	0	7	8	2	1	6	4	12	9	3	10	11	5
average		0	7	8	3.5	3.5	5.75	6.25	8.25	7.25	3.75	6	10.5	8.5
		K = 4												
compl.	16	0	6	12	9	1	11	3	2	4	8	10	5	7
compl.	64	0	6	3	8	5	11	12	1	4	10	2	7	9
circle	16	1	11	12	8	4	5	6	3	7	0	2	10	9
circle	64	0	10	9	11	6	7	12	5	4	3	2	1	8
average		0.25	8.25	9	9	4	8.5	8.25	2.75	4.75	5.25	4	5.75	8.25
		K = 8												
compl.	16	1	3	9	0	11	7	6	2	10	4	8	5	12
compl.	64	0	12	4	10	3	6	9	11	2	5	8	1	7
circle	16	7	0	4	5	6	12	3	9	10	2	1	8	11
circle	64	0	2	12	3	11	8	9	5	10	7	1	4	6
average		2	4.25	7.25	4.5	7.75	8.25	6.75	6.75	8	4.5	4.5	4.5	9

NK-LANDSCAPES. In Table 2, we show the relative behavior in terms of ERTs of different strategies experimented with a portfolio composed by five $(1+\lambda)$-EA based on the c-bit-flip operator with $c \in \{1, 2, 4, 8, 16\}$. Perhaps the most interesting observation for NK-landscapes is that the uniform-static strategy is the best performing and none of the considered DAMS variants is able to outperform it. This might be surprising at first sight, but not if we account for the time required to learn the metaheuristic to apply. In fact, the fitness level for NK-landscapes can be shown to increase very abruptly in the early stages of the search (see Fig. 8). Hence, the different fitness windows where one has to choose the best operator are very tight which is to contrast with the time it may need for a strategy to detect which operator is actually the best to apply. Actually, the general lessons that we can learn from our experiments with the NK-landscapes can be formulated as following. First, in a blackbox scenario, the time during which a metaheuristic is the best one could depend strongly on the problem fitness landscape. Hence, learning this landscape at runtime is for sure a plausible alternative. Second, further studies are needed to characterize the cost of the learning stage in an adaptive selection strategy in function of the considered landscape, and to design novel alternative strategies that would be able to minimize the learning cost.

3.3 CONTRIBUTION #2: BENCHMARKING FOR ADAPTIVE OPERATOR SELECTION

With no surprise, an algorithm selection strategy can have different behaviors depending on the benchmark problem at hand. For instance, for the NK-landscapes, the performance of the adaptive strategies was mitigated compared to static ones. This raises the difficult challenge of eliciting what makes an online selection strategy effective and when it should be preferred to an offline and finely tuned static one. We argue that two issues have to be considered in a combined manner in order to tackle this challenge: (i) the exploitation/exploration trade-off exposed by a given strategy which is often tightly related to the way the rewards are computed and updated, and (ii) the features relating the optimization problem to the considered algorithms in the portfolio. These two aspects were previously considered in the literature. For instance, in [BP14], continuous benchmarks from the COCO platform are experimented using a portfolio of variants of the well-established differential evolution operator. Other studies considered to directly define the rewards associated with the algorithms included in the portfolio using particular stochastic distributions [Thi07; Fia+10a]. In [Thi07], the set of possible rewards is defined by different uniform random distributions

Algorithm 6: A single-solution single-operator basic metaheuristic (e.g., a $(1+\lambda)-$ EA).

```
1 x_0 ← initial solution();
2 repeat
3 |   for i ∈ {1,...,λ_t} do
4 |   |   y_i ← operator(x_t);
5 |   x_{t+1} ← selection(x_t, y_1, ..., y_λ);
6 until stopping criterion is satisfied;
```

that are reassigned randomly to the portfolio at different time intervals. Seemingly the same idea is used by a panel of studies by Fialho and others [DaC+08]. For instance, in [Fia+10a], the so-called 'Two-Values benchmarks' is used where two possible reward values and a probability of wining the highest is defined depending on pre-computed time intervals. Recently, a benchmark is proposed in [GLS16] where the rewards depend on the number of times that an operator is applied during a time window.

Several properties should ideally be fulfilled by a relevant benchmark for the algorithm selection problem. First, one has to take into account the stochasticity of most heuristic algorithms. Hence, the reward of each algorithm in the portfolio should typically be defined by choosing a relevant probability distribution. Moreover, to accurately appreciate the *relative* quality of a target adaptive algorithm selection strategy, the so-called 'oracle', that is the optimal selection strategy, should be known. At last, since the performance of an algorithm in the portfolio could evolve during the optimization process, the reward distribution has to be tightly coupled with the state of the search. We argue that despite their skillful design, the existing benchmarks are not sufficient by their own to allow for a global fundamental understanding of the design of adaptive methods and the setting of relevant theory for them.

In the following, we discuss our contribution [Jan+16; Jan17] to the design of a general purpose benchmarking methodology. Inspired by previous work form fitness landscape analysis [VCC03], we propose a high-level approach to generate abstract 'benchmark' scenarios based on the Fitness Cloud (FC) model. The proposed approach is to be viewed in a complementary manner to existing benchmarks. In the FC model, the state of the search is naturally defined by the fitness of the current solution, and the performance of a given metaheuristic is function of this fitness value. The reward distribution is hence *not* controlled explicitly; but instead, implied by the considered adaptive mechanisms.

3.3.1 The Fitness Cloud Model

Although the proposed approach is independent of a particular metaheuristic, let us consider for the sake of clarity the template of Algorithm 6 rendering the design of a basic single-solution single-operator metaheuristic. The considered iterative algorithm has two parts. First, a stochastic local operator is applied to the current solution x_t to produce a set of λ_t candidate solutions y_i. For instance, such an operator could be the random bit-flip mutation when the search space is the set of binary strings. Second, a new current solution x_{t+1} is selected. This is typically performed according to the fitness values of the newly generated solutions y_i, and the current solution x_t. Notice that despite its simplicity such a template encompasses a wide range of algorithms.

FITNESS CLOUD. The Fitness Cloud Model (FCM) informs about the fitness value of solutions after one iteration according to the fitness of the current solution. To make it simple, the FC model supposes that the state of the search is only given by the fitness $f_t = f(x_t)$ of the current solution x_t. Assuming that the selection of the next solution only takes into

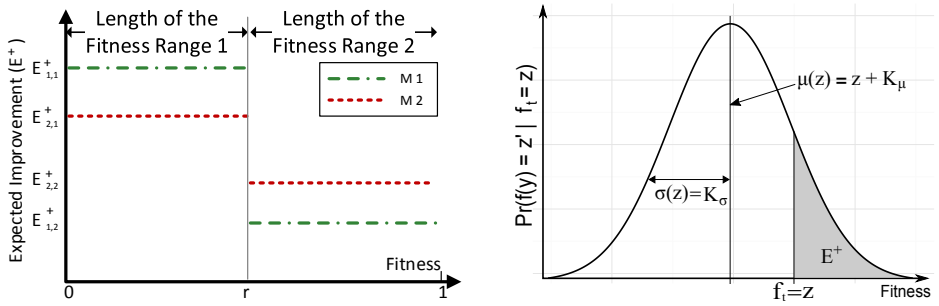


Figure 11: Fitness cloud model: scenario with two metaheuristics and two fitness ranges.

account the fitness values (which is a common practice for a wide range of blackbox optimization algorithms), no particular model is required for this. However, a specific model is needed for the fitness values of the newly generated solutions in order to capture the stochastic behavior of most evolutionary operators. The basic idea behind the FC model is to assume that the fitness distribution after applying a stochastic operator, say op, is given by a conditional probability distribution that depends solely on the current fitness: $\Pr_{op}(f(y) = z' \mid f_t = z)$. Of course, such a distribution is not known in practice. Nonetheless, we consider to define it explicitly, which allows us to explicitly relate the behavior of an operator to the current state of the search, i.e., the fitness level attained currently. Consequently, by simply defining different such probability distributions emulating the behavior of different (virtual) operators, we end up with an abstract and high-level 'benchmark' that can be emulated and experimented with respect to specific adaptive selection strategies. Notice however that this is very different from standard benchmark design methodologies which operate at the solution space. Roughly speaking, the word 'benchmark' is to be understood with respect to the online algorithm selection problem which consists in computing the best/optimal scheduling of operators.

The first step when using the Fitness Cloud Model (FCM) as a benchmark for algorithm selection, is to specify the conditional probability distribution. Different choices can be made such as discrete distributions (binomial, Poisson, etc.), or continuous distributions (normal, Weibull, etc.). Given its properties of convergence, we choose to use a normal distribution: $\Pr(f(y) = z' \mid f_t = z) \sim \mathcal{N}(\mu(z), \sigma^2(z))$ where $\mu(z)$ and $\sigma^2(z)$ are respectively the mean and the variance of the normal distribution which can depend on the current fitness f_t. As a consequence, the evolution of the fitness during one iteration follows a conditional probability distribution which embeds the previous distribution and which depends on how the selection of the next solution is carried out by the optimization process. One important feature of the probability distribution is the *expected improvement* after one iteration, denoted by $E^+(z)$, which is the expected progress of the fitness given the current fitness value is z:

$$E^+(z) = \int_z^{\infty} \Pr(f_{t+1} = z' \mid f_t = z)\ z'\ dz'$$

The previous considerations are enough general to allow us to define in a high-level manner the behavior of applying one algorithm from a portfolio as a function of the fitness level attained globally by the optimization process. By only fixing the parameters of the conditional probability (or the expected improvement), we end up with a high-level portfolio in which the effect of applying one metaheuristic given one current initial solution with a given fitness level is explicitly fixed. Of course, those parameters are not made available for the algorithm selection strategy it-self.

A CONCRETE EXAMPLE. A simple (benchmark) optimization scenario that we can emulate using the FC model is illustrated in Fig. 11. Two elitist (virtual) metaheuristics are considered in the portfolio and the possible fitness values are normalized in the range $[0, 1]$. The

whole range $[0,1]$ is then divided into two fitness ranges: the first one from fitness 0 to $r \leqslant 1$, and the second range from r to 1. The relative behavior of each algorithm in the portfolio is modeled accordingly in each fitness range. At each fitness range, we shall fix the mean and variance of the conditional normal distribution as follows: $\mu_i(z) = z + K_{\mu_i}$ and $\sigma_i^2(z) = K_{\sigma_i}$ where for each metaheuristic M_i, $i \in \{1,2\}$, parameters K_{μ_i} and K_{σ_i} are different constant numbers at each fitness range. Therefore, we end up with 9 parameters to be fixed in this scenario: r, and the 8 parameters for the normal distributions for each metaheuristic and at each fitness range. However, we can show that the expected running time to reach the optimal value 1 depends solely on the expected improvement of each metaheuristic (when using an elitist strategy). Hence, only 5 parameters are free as illustrated by Fig. 11; where $E^+_{i,j}$ denotes the expected fitness improvement of metaheuristic M_i, $i \in \{1,2\}$, for the fitness range $j \in \{1,2\}$. Notice that in our example, we assume that the best metaheuristic for the first fitness range is M_1, whereas it switches to M_2 in the second fitness range, i.e., $E^+_{2,1} < E^+_{1,1}$, and $E^+_{1,2} < E^+_{2,2}$, which makes it straightforward to figure out the 'oracle' strategy when evaluating the performance of given selection strategy. Finally, like in many optimization problems, we assume that the expected improvement decreases when the fitness range increases: $E^+_{1,2} < E^+_{1,1}$, and $E^+_{2,2} < E^+_{2,1}$.

Besides this example, we can generate several other types of scenarios [Jan17]; by considering different number of algorithms with variable relative fitness cloud distributions, different fitness ranges modeling different stages of the optimization, different modeling of how the fitness cloud distribution with respect to one algorithm evolves as a function of the fitness range (e.g., linearly), etc. The major difficulty is then to fix the parameters of the model in order to end up with an appropriate and relevant benchmark allowing to study a particular target issue, e.g., the ability of a selection strategy to detect the best algorithm, the ability to discard non-useful ones, the time to learn the best algorithms, the ability to outperform a static strategy, etc.

3.3.2 Selected Experimental Results

RESULTS IN THE SEQUENTIAL SETTING. From the illustrative scenario described in the previous section, we are able to study 3 concrete cases exposing different challenges. In Fig. 12, we summarize these 3 experimental cases while providing the parameters used in the FC model. First, notice that we choose K_μ negative in order to emulate the behavior of a typical stochastic evolutionary operator that decreases (in average) the fitness of the current solution as it is the case very often in practice. In all the 3 cases, the EIs of M_1 and M_2 are the same in the first fitness range and differ by a factor of 2. In Case 1, the EI values are much more closer in the second range compared to Case 2, i.e., an oracle for Case 1 would act as in Case 2, but the performance of a static selection strategy that would always choose M_1 becomes much closer to the oracle than in Case 2. As for Case 3, the same factor of 2 is kept between the EIs in the second range; but the EIs has been reduced by a huge factor of 15, hence making the progress in the second range relatively much more difficult than in the first range compared to Case 2.

In Fig. 13, we show the performance obtained when using: (i) two static strategies (choosing one metaheuristic during the whole run), (ii) two adaptive ones (based on standard AP and UCB), (iii) a random (baseline selection) strategy, and (iv) the oracle strategy. The performance is the average number of evaluations to reach the target fitness value 1 and it is depicted as a function of the length r of the first fitness range considered in the scenario. Actually, the relative expected improvements and the r-value allows us to elicit different relative behaviors of the considered strategies. Notice in particular that the difference over the r-values of the performance between the oracle and the best static strategy (either with M_1 or M_2) represents the maximum performance gap between an optimal adaptive method and an optimal offline static strategy tuned for each value of r. This gives a hint on when an adaptive online strategy should be preferred over an offline one. The comparison of case

Values are given with a factor of 10^{-3}.

Cases	Meta.	Fitness range 1			Fitness range 2		
		$E^{+}_{i,1}$	K_{μ_i}	K_{σ_i}	$E^{+}_{i,2}$	K_{μ_i}	K_{σ_i}
Case 1	M_1	6	−1	16.27	1.8	−2	6.72
	M_2	3	−1	8.72	2	−2	7.24
Case 2	M_1	6	−1	16.27	1	−2	4.59
	M_2	3	−1	8.72	2	−2	7.25
Case 3	M_1	6	−1	16.27	0.2	−2	2.14
	M_2	3	−1	8.72	0.4	−2	2.84

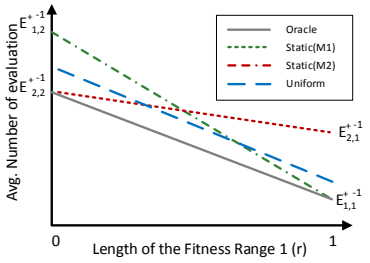


Figure 12: Parameters values of the FC model in 3 representative cases (left table) with the corresponding (tight) upper bounds on the expected running time (right subfigure)

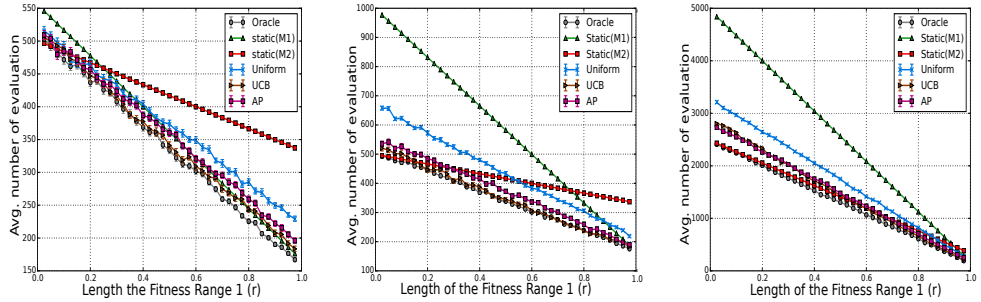


Figure 13: Comparison of selection methods with the best parameters settings for UCB, and for AP. From left to right: test cases 1, 2, and 3.

2 and 3 shows that when the average expected improvements at the second stage is much lower than in the first stage, an adaptive method becomes less efficient except when the length of the first stage is very large. Indeed, the time that can be gained in the first stage becomes negligible and the main difficulty then turns out to be the final convergence to the optimum value. When the scale of the average expected improvements between the two stages is moderate like in test case 2, an adaptive method like UCB strategy is very effective. However, when the performance difference between metaheuristics at one stage of the search becomes small, like in test case 1, the problem is equally difficult for all metaheuristics in the portfolio, and the adaptive selection becomes rather useless besides the fact that it becomes more difficult to detect the best performing metaheuristic at a given iteration.

RESULTS IN THE DISTRIBUTED SETTING. In the distributed setting, we use an even more simple scenario where only one fitness range and two operators with fixed parameters of the FC model are considered. However, the mean and standard deviations are chosen in a specific manner in order to better grasp the behavior of different distributed selection strategies with respect to the different aspects such as: the number of distributed nodes, the reward aggregation function, the impact of a heterogeneous selection strategy, and the communication-to-computation cost. In particular, one can see in Fig. 14 the relative performance of different Homogeneous and Heterogeneous selection strategies discussed previously and using the adaptive Master/Worker architecture. We are for instance able to show that using a mean reward aggregation function is clearly outperformed by a maximum reward function. The difference between a homogeneous and heterogeneous setting is also found to be mitigated and depends on the selection it-self. When comparing the best setting of given selection policy, the UCB strategy appears to be the best one, followed by ϵ-greedy strategy, which are both better than the AP strategy. These first results can be explained by the ability of the UCB machine-learning inspired strategy to efficiently learn the best operator to apply in a given distributed round when using the maximum reward. All adaptive strategies are also found to share a relatively good performance when the other

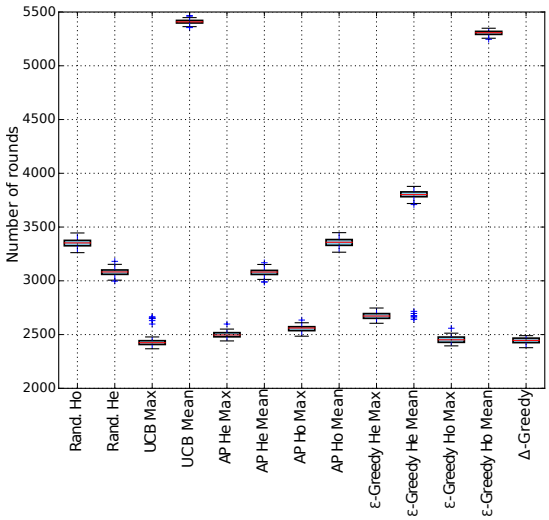


Figure 14: Number of distributed rounds to the optimal fitness using operator $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$, and operator $2 \sim \mathcal{N}(-10^{-3}, 5 \times 10^{-4})$ and $n = 256$ workers.

design components, that is the choice of the reward function, and the heterogeneity type, are well designed.

3.4 OTHER RELATED CONTRIBUTIONS

As briefly discussed in the introduction, in addition to our interest to the online adaptive and distributed algorithm selection problem, we were also concerned by other tightly related issues and methodologies dealing with the proper configuration and high-level design of general purpose search heuristics. This is discussed in the following by mainly highlighting the general context of our research and sketching some contributions on hyperheuristic design [DD12] (Section 3.4.1) and offline parameter configuration [Lie+17b] (Section 3.4.2). Both contributions are with respect to a sequential compute environment.

3.4.1 Hyperheuristics based on neighborhood tree search

As mentioned in the introduction of this chapter, online adaptive algorithm selection shares a number of issues with selective hyper-heuristics [Bur+10]. Selective hyper-heuristics deal with the accurate (adaptive) combination of different low level heuristics, especially when dealing with cross-domain optimization [Och12; Bur+11], i.e., different problems from different domains tackled by the same ('autonomous') optimization algorithm. In our research, we were also interested in such high-level search algorithms, specifically in the context of the design of advanced local search based algorithms combining multiple neighborhood operators [DD12; Yah+15]. In the following, we provide an overview of our contribution on the combination of different neighborhood structures within an effective and problem independent local search hyperheuristic [DD12].

MULTIPLE NEIGHBORHOOD LOCAL SEARCH. Local search heuristics [HS04] refer to algorithms where a solution is updated iteratively while performing little transformations on its vicinity. Those transformations are usually based on a *neighborhood* function (or structure) $\mathcal{N} : X \to 2^X$, which assigns a set of neighboring solutions $\mathcal{N}(x) \subset X$ to any solution $x \in X$. In its most basic variant, referred to as *hill-climbing*, the local search stops when the current solution is not outperformed by any neighbor, i.e., a *local optimum* is reached.

Given a number of neighborhood structures, their accurate combination within a local search based optimization process raises different design issues. Variable neighborhood search (VNS) and its several variants [MH97], is a popular metaheuristic in this context which is are based on the systemic change of the neighborhood structure within the search. Starting with a first neighborhood structure, a Variable Neighborhood Descent (VND) performs local search until no further improvements are possible. From this local optimum, the local search is continued with the next neighborhood. If an improving solution is found, then the local search continues with the first neighborhood, otherwise the next available neighborhood is explored, and so on until no further improvements can be obtained. It is well known that the performance of VND (and VNS in general) can highly depend on the order the neighborhoods are alternated. In standard variants, it is often admitted that ordering neighborhoods in an increasing cost/size is a reasonable strategy.

The issue of how to combine/exploit/search different neighborhoods is not new and one can find many different studies on the subject. To cite a few, in [PR08], a fast relaxation of neighborhoods is evaluated in order to select the most accurate solutions. In [HR06], a self-adaptive strategy is used to rank neighborhoods and to dynamically choose the best suited ordering. A number of specific multi-neighborhood combination functions can also be found. For instance, many studies consider to take the union of some basic neighborhoods. The so-called neighborhood composition and the token-ring search are also other well known neighborhood combination functions, see e.g., [LHG11; GS06; GRH08; LGH11; VMS12]. Hyper-heuristics can also be considered as a high-level approach operating in the neighborhood space and providing advanced hyper-strategies for neighborhood combination.

NEIGHBORHOOD TREE SEARCH. In an attempt to gain in generality when dealing with the combination of several available neighborhood structures, we rely on the simple observation that defining how neighborhoods are alternated is nothing else than defining a specific hyper-strategy to traverse a neighborhood tree, where the root of the tree represents the initial candidate solution and intermediate nodes represent solutions obtained by applying one of the possible neighborhoods. In other words, we view the trajectory of a multiple neighborhood search as a high-level neighborhood path, where path nodes are solutions and every hop in the path represents the exploration of one solution using one neighborhood among those available. Following this observation, we term *a neighborhood tree search (NTS)* a strategy which is able to traverse the neighborhood tree efficiently searching for promising paths. It should be clear that a systematic traversal (exploration of all neighborhood branches) is not efficient especially when the number of neighborhoods is high. This should actually remind us of several general purpose strategies from exact optimization, such as dynamic programming and branch-and-bound, as well as from machine learning, such as Monte-Carlo tree search.

In [DD12], we investigate the possibility of backtracking to previously visited solutions while branching and pruning tree nodes all along the search trajectory. We show that this idea with basic iterative improvement descents leads to efficient local search strategies both in terms of solution quality and computing cost. More specifically, we consider a simple randomized neighborhood selection strategy, where the choice of which neighborhood to select at runtime is made uniformly at random among those not yet explored. When effectively branching a neighborhood, we consider both deterministic and randomized adaptive strategies, basically relying on the neighborhood path traversed by the search in previous rounds. As for backtracking, we investigate strategies based on tournament selection inspired by standard evolutionary selection.

We study the properties of the proposed approach by considering simultaneously two different and well-studied problems: the Single Machine Total Weighted Tardiness Problem (SMTWTP) in the family of scheduling problems, and Location Routing Problem (LRP). Both problems are NP-Hard and allows us to study the effectiveness of our hyperheuristic

approach when tackling relatively complex problems coming from different optimization domains. Many previous studies have been successfully applied to solve the SMTWTP using hybrid variable neighborhood like searches. LRP is a more sophisticated problem which involves two simultaneous decisions: which depots to open (location problem) and what routes to plan (routing problem). Common to these two problems, a number of natural neighborhood structures can be considered making them two excellent case studies to analyze how the designed NTS variants perform (e.g., twelve neighborhoods are used for LRP). Through extensive experiments, we show that NTS is able to dynamically find its way along the neighborhood tree without any specific tuning and leads to substantial improvements in the solving of the two considered problems. More importantly, we were concerned by providing a comprehensive analysis of the complex behavior of an NTS based algorithm at the aim of gaining a better understanding of its critical design components. In particular, we show that NTS can lead to different (and incomparable) trade-offs in terms of solution quality and running time depending on the branching/pruning strategy used to explore/-traverse the neighborhood tree which makes it a promising approach offering novel and interesting (hyper-)search abilities.

3.4.2 Landscape-aware offline algorithm configuration

In addition to online adaptive algorithm selection approaches, we were recently concerned by *offline* algorithm configuration. Given a number of algorithm parameters, (offline) automatic algorithm configuration [Bir09] seeks a good configuration, that is a particular fixed choice of the parameter values that best suits the solving of some *a priori* unknown instances. The motivation of such a methodology is not only to get rid from the burden of a manual calibration or the bias of personal and ad-hoc configuration processes, but more importantly to set up a principled approach for algorithm design, allowing to systematically explore their strengths and weaknesses when tackling a whole family of problems. Several approaches have been proposed, ranging from experimental design [AL06], statistics [Bar06], heuristic search [Hut+09], and racing [Bir+02; Bir09; Lóp+16]. In the following, we provide an overview of our contribution on bridging automatic algorithm configuration with fitness landscape analysis [Ver16], at the aim of achieving more powerful general purpose blackbox optimization algorithms [Lie+17b].

AUTOMATIC ALGORITHM CONFIGURATION. Most of the existing automatic algorithm configuration methods can be viewed from a machine learning perspective as operating in a training phase followed by a test or a production phase. Based on some given instances forming the training set, the training phase is intended to learn a good configuration that would hopefully perform well when experimented later, on some new unseen instances coming from the production phase. Like any machine learning technique, the properties of the training set is a key issue in order to guarantee a high accuracy of the output configuration. The heterogeneity of training instances was for instance found to be a challenging issue when using iterated racing in the context of a tuning scenario implying SAT instances [Lóp+16]. We actually argue that a single output parameter configuration might not be suitable for the target algorithm to best suit a whole set of (heterogeneous) instances having different properties. Like a number of other existing studies [XHL10; HHL11; Kad+10], we hence advocate for the computation of a *set* of configurations, not a single one, that can then be mapped accurately with respect to the characteristics or features of an instance. For instance, in Hydra [XHL10], a portfolio builder is used together with an automatic configuration method in order to construct a portfolio of algorithm configurations. The portfolio builder typically uses problem features to discard or add new configurations found by automatic configuration, and the method was proved effective when experimented with SAT specific tools. However, it requires both a suitable portfolio builder and a domain-specific knowledge. In our work, we are concerned with blackbox optimization and hence propose to rely on the

so-called fitness landscape analysis to extract and to inject problem independent features into the tuning process.

FITNESS LANDSCAPE ANALYSIS. Generally speaking, fitness landscapes analysis [Ver16] provides a set of general-purpose tools and a principled approach to systematically investigate the characteristics of an optimization problem in an attempt to guide algorithm designers towards a more in-depth understanding of the search behavior, and thus towards more effective algorithms [Mer04; RE14]. A typical issue addressed in fitness landscape analysis consists in studying how the performance of a given algorithm configuration can be impacted in light of some features extracted with respect to the considered problem instances. In particular, different general-purpose features were studied for this aim [RE14], and such landscape features have proved their usefulness in characterizing instances [SL12]. The general idea developed in our work is that such features can actually serve to differentiate which parameter configuration can be more suitable for a particular problem instance, both during the training phase and during the production phase. In other words, since it might be useless to search for just one single parameter configuration for a heterogeneous instance set, an alternative solution would be to consider a whole set of configurations that are explicitly associated with some computable instance features.

INJECTING LANDSCAPE-EXTRACTED FEATURES INTO ALGORITHM CONFIGURATION. In our recent work [Lie+17b], we adopt a landscape-oriented methodology to strengthen the accuracy of automated algorithm configuration based on iterated racing. By partitioning the training set into different groups based on the value of landscape features, we conduct an independent training phase in parallel for each group, thus ending up with multiple algorithm configurations corresponding to the different groups. At the production phase, the appropriate configuration is selected based on the feature value of the considered instance. As a byproduct, we are able to derive a novel landscape-aware methodology that complement existing automatic algorithm configuration techniques. By fairly taking into account the extra computational cost induced by our methodology, we investigate the gain of deciding which parameter configuration to choose for an unseen production instance based on general-purpose low-cost computable features. Our empirical findings, using a set of NK-landscape heterogeneous instances with a variable degree of ruggedness and neutrality, reveal that landscape-aware iterated racing is able to find better configurations when experimented with a conventional memetic algorithm with tunable population size, variation operators, crossover and mutation rates.Our work can actually be viewed as a first step towards the establishment of more powerful and finely tuned landscape aware optimization algorithms. It can also be viewed to some extent as providing a very simple high-level (blackbox) portfolio builder that, it is our hope, would serve as a basic template to design a more advanced and principled approach targeting the solving of complex, blackbox and cross-domain optimization problems.

3.5 CONCLUSION AND PERSPECTIVES

SUMMARY. In this chapter, we described our work on the design of high-level optimization techniques for the proper choice of algorithm components and parameters, with a particular focus on adaptive online operator selection techniques. Two main challenges are specifically addressed: (i) how to design distributed strategies allowing distributed nodes to adaptively coordinate their actions at the aim of cooperatively identifying the best performing algorithm at different steps of the optimization process, and (ii) what makes an adaptive selection strategy effective and when it should be preferred to a static *a priori* fixed choice. For that purpose, we rely both on an island model and a Master/Worker architecture, to leverage existing machine learning techniques designed to operate in an inherently sequential

setting. In particular, we show that reward aggregation and heterogeneity can play a crucially important role. We also propose an abstract and high-level methodology based on the fitness cloud model allowing us to emulate an adversary benchmark optimization setting for the operator selection problem. By explicitly modeling the behavior of an algorithm as a function of the fitness level reached at some steps of the optimization, we are able to show that state-of-the-art techniques can only succeed when the learning-to-optimization cost/-time is kept reasonable which might actually be a function of the underlying landscape(s). On the other hand, and besides adaptive algorithm selection, we were also concerned by other high-level autonomous and automated search paradigms, i.e., hyperheuristics and offline parameter tuning. Several perspectives of our work can be identified. Some of them are discussed in the following.

PARALLEL AND DISTRIBUTED DEPLOYMENT. The adaptive distributed algorithms proposed in our work were validated using a simulation oriented methodology, since our primary goal was to first gain a more fundamental understanding of the main design challenges and the way to effectively address them. In this respect, a clear perspective of our work is to consider concrete implementations of our protocols in a real distributed and parallel compute environment and considering more sophisticated application-oriented problems. We argue that the main challenge will be to handle the practical communication-to-computation cost, which we addressed in a very abstract manner, e.g., number of messages, batch scheduling. Two extreme scenarios need further investigations. In a very fine-grained compute setting where the cost of function evaluation is extremely low, different extensions of our distributed algorithm selection framework can be considered in order to better fit the concrete compute environment. For instance, a hierarchical distributed design based on the hybridization of the Master/Worker architecture and the Island model is an interesting option which would: (i) eventually fit a complex hardware setting where shared memory and/or distributed devices are available, and (ii) provide new opportunities in designing more powerful adaptation mechanisms using advanced reward aggregation functions and cooperative selection mechanisms. The other extreme scenario is when the cost of function evaluation is huge, typically corresponding to the so-called expensive optimization setting requiring to use specific sampling techniques and meta-models, e.g. [JSW98; Vu+17]. Besides the fact that parallelism is a must since one can for instance take much benefit from the fact that more function evaluations can be made in parallel; several other interesting research questions can be raised. For instance, an interesting issue is whether different exploration/exploitation trade-offs ruling the design of existing adaptive techniques can be designed in order to satisfy the harsh conditions and/or restrictions on the amount of available (distributed) computing time.

OTHER DISTRIBUTED TECHNIQUES. We notice that our work on hyperheuristics and offline landscape aware tuning was described in a sequential setting. Nonetheless, parallelism and distributed algorithms are not far away when having a more close and global look at our work. Firstly, the proposed neighborhood tree search hyperheuristic can benefit from a high-level distributed design. Similar to the parallel Branch-and-Bound discussed in the previous chapter, high-level parallelizations would be an interesting research path. From a solution quality perspective, this means that cooperative and distributed branching and pruning operations at the neighborhood space level can be designed, which would eventually lead to a more effective search. For instance, by smartly coupling the solution space and the neighborhood space, the computing flow can be redesigned in order to make different parallel processes explore concurrently and cooperatively diversified and promising subtrees in the neighborhood tree. From a running time perspective, parallel tree exploration strategies would definitively lead to a high-level parallel search with a substantial speed-up. Secondly, our offline landscape aware methodology is actually inherently parallel, since the training instances are partitioned into independent clusters. This means that the training time can

be straightforwardly decreased proportionally to the number of clusters. Knowing that the training phase can be rather compute intensive, such a straightforward implication is in our opinion a nice property. Moreover, this simple observation leads to several other advanced research issues. In fact, the configurations found for a subset of instances can serve to finely tune an algorithm on an other subset of instances. Distributed tuning processes can hence benefit from information exchange and online cooperation to share (resp. discard) promising (resp. poor) subsets of parameters, which would lead to novel high-level and distributed automatic configuration algorithms. Interestingly, the assumption that instances with 'similar' landscape features can be tackled using 'similar' algorithms, configurations, components, etc, needs to be addressed in a more comprehensive manner especially in a blackbox optimization scenario. For instance, deriving a similarity or distance measure that can relate the instance space to the parameter space is a hot issue. Thus, there is much research to conduct in this direction independently of the target algorithmic technique (e.g., online or offline) and the nature of the compute environment (e.g., distributed or not).

ONLINE AND OFFLINE CROSS-DOMAIN OPTIMIZATION. The work described in this chapter can be viewed from a more global research perspective as an attempt to increase the generality of heuristic search algorithms given the wide range of problem domains that one can encounter. A global perspective of our work is to generalize the use of online and offline control and tuning techniques and to improve their *cross-domain* ability. For that purpose, we have identified three major research approaches. Firstly, online and offline techniques can be coupled to work hand in hand in order to better address the heterogeneity of problem domains and instances. For example, offline parameter tuning can be viewed as an optimization/decision problem under uncertainty, where the uncertainty comes from the algorithm it-self and from the problem domain/instance as well. In this respect, machine learning based (adaptive) approaches, e.g., multi-armed bandits, domain adaptation, transfer learning, can be particularly helpful in order to improve the robustness of existing automatic configuration tools. Secondly, injecting blackbox landscape features into the optimization process is a promising research direction that needs more in-depth investigations. For example, what (cross-domain) blackbox features are the most effective in informing about the difficulty of a problem instance, or the efficiency of some available algorithms, is a difficult question. How to extract those features, a priori or even a posterior, in an online or in an offline fashion, and how to guide the high-level search accordingly is another challenging perspective. Finally, parallelism and distributed computation are of course a key aspect. The design of distributed and autonomous solvers that would be able to learn while optimizing, or inversely to optimize while learning, is a promising research path, since such an approach would naturally take into account the cost of additional learning or optimization efforts and simultaneously offer novel algorithmic options such as the possibility of exchanging useful information informing about the degree of similarity or difficulty of different problems.

4 MULTI-OBJECTIVE OPTIMIZATION AND DECOMPOSITION

In this chapter, we describe our main contributions in multi-objective optimization with a particular focus on decomposition based approaches. The main challenge is on the design and analysis of novel algorithmic components at the aim of studying and improving the search ability of decomposition-based algorithms, and taking full benefits from their high-level enabled parallelism. The following aspects will be addressed:

- In the introductory section, we recall some background on multi-objective evolutionary optimization. We also provide an overview of the state-of-the-art decomposition-based MOEA/D algorithm, while stating our general scientific interest in such a framework.
- In Section 4.2, we discuss our contributions on two core issues in decomposition, namely (i) the specification and understanding of the scalarizing function used to transform the original multi-objective problem into several single-objective problems, and (ii) the mating selection and replacement mechanisms used to evolve the population. These aspects are studied in a sequential compute environment.
- In Section 4.3, we discuss our contributions on designing parallel decomposition algorithms. First, we deal with the adaptive and distributed setting of the weight vectors used in decomposition. We show that using very local distributed rules to define the search directions adaptively, can lead to high quality solutions and high parallel efficiency. Second, we provide a fine-grained message passing variant of MOEA/D which is to our best knowledge the first to achieve scalability while providing non-trivial approximation quality / acceleration trade-offs.
- In Section 4.4, we discuss very briefly some issues related to the design of effective search operators. First, we highlight our work on connecting local search with decomposition for discrete problems, and we show how decomposition can lead to novel evolutionary components that are simple enough to enable parallelism. Second, we highlight our work on incorporating machine learning inspired techniques to design multiobjective evolutionary operators. In particular, we describe a new alternative to leverage the well-established single-objective CMA-ES (Covariance Matrix Adaption Evolution Strategy) optimizer for multi-objective problems.

COLLABORATORS, PROJECTS AND RELATED PUBLICATIONS

COLLABORATORS. H. Aguirre, D. Brockhoff, M. Basseur, O. Cuate, M. Drodzik, A. Goëffon, A. Liefooghe, G. Marquet, S. Martinez-Zapotecas, H. Monzon, J. J. Palacios Alonso, M. Sagawa, O. Schuetze, J. Shi, K. Tanaka, E-G. Talbi, S. Verel, Q. Zhang

PUBLICATIONS. [Der+14b; Lie+17a; Mon+17; Shi+17b; Sag+17; Cua+17; Der+16; Der+16; Sag+16; Bas+16; Mar+15a; Mar+15b; Dro+14; TBD13; LD16; AD15; Mar+14; Der+14a; Der+15; DBL13]

PROJECTS. ANR PRCI France/Hong-Kong BigMO project (Coordinator; HK coordinator: Q. Zhang) (2017-2021), PHC Procore France/Hong-Kong (Coordinator) (2016-2017), International associated Lab France/Japan LIA-MODO (Co-founder) (2017-), JSPS-MEXT France/Japan project (2013-2016), S3-BBO Ayame/Inria associate team (2015-), ECOS Nord France/Mexico Project (2016-2020). (See Appendix A.3)

4.1 INTRODUCTION AND BACKGROUND

4.1.1 General Context and Definitions

MULTI-OBJECTIVE OPTIMIZATION AND DECISION MAKING. Multi-objective optimization appears in several application fields, such as ambient intelligence, cloud computing, logistics, smart-cities, etc, where the underlying problems can be modeled as optimization problems with multiple criteria to be optimized simultaneously. For example, on may wish to minimize the cost of deploying sensors in a city while maintaining a good quality of service to users and minimal disruption during the deployment phase. In practice, these objective functions are many often conflicting, and it is unlikely to find one single solution which is able to optimize all target objectives simultaneously at the same time. Instead, there exist a whole *set* of solutions providing different trade-offs between the considered objectives.

From a decision making perspective, the main issue when considering a multi-objective optimization problem is to choose one appropriate solution that best meets the expectations, constraints or preferences of the decision maker. Actually, solving a multi-objective optimization problem is tightly related to the decision making process, and three main classes of approaches are to be differentiated. In the first class, called *a-priori*, the decision maker provides some knowledge about her preference before even the optimization process is carried on. The main goal is to help the optimization process computing the desired solution. In practice, this consists many often in transforming the original multi-objective problem into a single objective problem where traditional techniques form single-objective optimization can be used. However, the decision maker might not have any specific knowledge about the problem at hand, and providing an accurate modeling of her preferences might be a difficult issue. In the second class, called *a-posteriori*, the goal is to find the whole set of solution offering the best attainable objective trade-offs or a good representative (approximated) subset. This allows the decision maker to eventually acquire a full knowledge about this set and to extract one solution that best suits her expectations. Although there is no need to model the decision making preferences, computing a whole set of possible solutions is a challenging task and the analysis of these solutions by the decision makers might also be an issue, especially when a large number of objectives are to be considered. In the third class, called *interactive*, the decision maker needs to continuously interact with the optimization process in an online iterative fashion. In particular, the decision maker can progressively refine her preferences according to the information and knowledge gained while the optimization process is running.

It is worth-noticing that each of these different classes present both some limitations and some benefits, depending on the problem to tackle and the decision making setting. In our work, we are mainly concerned with a posteriori approaches, and more particularly on providing the decision maker with an approximated solution set, i.e., the decision making phase it-self is not addressed but only the optimization phase.

BASIC DEFINITIONS. In the rest of this chapter, a *multi-objective optimization problem* is defined by an objective vector function $f = (f_1, f_2, \ldots, f_M)$ with $M \geqslant 2$, and a set $\mathcal{X}$ of feasible solutions in the *decision space*. In the combinatorial case, $\mathcal{X}$ is a discrete set. Actually, our work is mostly concerned with the combinatorial case, and hence this is assumed in the rest of this chapter unless stated explicitly. Let $\mathcal{Z} = f(\mathcal{X}) \subseteq \mathbb{R}^M$ be the set of feasible outcome vectors in the so-called *objective space*. To each solution $x \in \mathcal{X}$ is then assigned exactly one objective vector $z \in \mathcal{Z}$, on the basis of the vector function $f : \mathcal{X} \to \mathcal{Z}$ with $z = f(x)$. In a maximization context, an objective vector $z \in \mathcal{Z}$ is dominated by an objective vector $z' \in \mathcal{Z}$, denoted by $z \prec z'$, iff $\forall m \in \{1, 2, \ldots, M\}$, $z_m \leqslant z'_m$ and $\exists m \in \{1, 2, \ldots, M\}$ such that $z_m < z'_m$. By extension, a solution $x \in \mathcal{X}$ is dominated by a solution $x' \in \mathcal{X}$, denoted by $x \prec x'$, iff $f(x) \prec f(x')$. This is illustrated in Fig. 15. A solution $x^\star \in \mathcal{X}$ is said to be *Pareto optimal* (or *efficient*, *non-dominated*), if there does not exist any other solution $x \in \mathcal{X}$ such that $x^\star \prec x$.

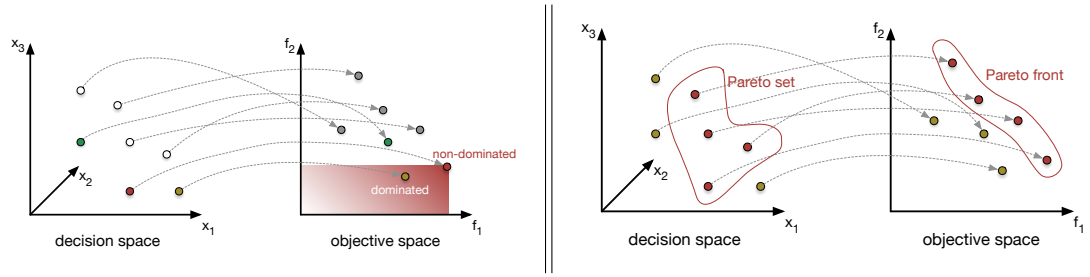


Figure 15: Illustration of the Pareto dominance relation, Pareto Set, and Pareto Front.

The set of all Pareto optimal solutions is called the *Pareto set* (or the *efficient set*). Its mapping in the objective space is called the *Pareto front*. One of the most challenging task in multi-objective optimization is to identify a minimal complete Pareto set, i.e., one Pareto optimal solution for each point from the Pareto front.

Generating a complete Pareto set is often infeasible for two main reasons [Ehr05]: (i) the number of Pareto optimal solutions is typically exponential in the size of the problem instance, and (ii) deciding if a feasible solution belongs to the Pareto set may be NP-complete. Therefore, the overall goal is often to identify a 'good-quality' *Pareto set approximation*. To this end, heuristics in general, and evolutionary algorithms in particular, have received a growing interest since the late eighties [Deb01; CLV07], which constitutes the main focus of our work. We also notice that two classes of problems can be distinguished according to the number of objectives. Multi-objective optimization problems often refer to a setting where 2 or 3 objectives are considered, whereas problems with at least 4 are termed as *many*-objective [ITN08]. This is mainly motivated by the specific challenges and goals corresponding to these two situations. In fact, the probability that a solution is Pareto optimal usually increases with the number of objectives and hence the size of the Pareto set increases. Consequently, the properties that should be fulfilled by an approximation set, as well as the design and relative performance of an optimization algorithm, can be seemingly different according to which class of problems is considered.

Before providing an overview on the existing evolutionary multi-objective optimization algorithms, we first discuss standard techniques used to evaluate the quality of an approximation set.

QUALITY INDICATORS. In contrast to a single-objective scenario, where the objective value of a solution is a single numerical value which is sufficient to assess its relative quality, in multi-objective optimization it is not clear how to appreciate the quality of a whole set of solutions. This is a critically important issue which is still being addressed in several research work, not only to compare different sets or algorithm outputs, but also to improve the dynamics of a multi-objective optimization process. Generally speaking two main properties are considered: (i) the *diversity* of solutions that is their ability to represent a wide and representative range of the Pareto front/set, and (ii) *convergence* which provides an idea on the how far the solutions are from the Pareto front. Of course, when considering a specific algorithm, this is to be related to the available computing effort. Among the different approaches allowing to render the quality of an approximation set, quality indicators are a standard and common tool. Such indicators assign, to any solution set, a real-value reflecting a given aspect of approximation quality. A large spectrum of quality indicators have been designed such as the inverted generational distance (IGD), The epsilon indicator (EPS), the R-metrics family (R), the hypervolume (HV) [KTZ06; ZKT08; Zit+03]. Notice that one single indicator might not be sufficient to provide a precise interpretation of the quality of a set, and it also depends on many factors such as the shape of the Pareto front, the distribution of non-dominated vectors in the objective space, or some user-defined parameters.

Consequently, there is no perfect agreement between these different indicators, which we actually studied more thoroughly in [LD16], but we do not present in this document for clarity. In the rest of this document, we shall use two popular indicators when assessing the performance of the different proposed algorithms, namely the epsilon indicator and the hypervolume. Roughly speaking, the epsilon indicator family gives the minimum factor by which the approximation set has to be translated in the objective space in order to (weakly) dominate the reference set, that is the exact or the best known solution set. The hypervolume gives the multidimensional volume of the portion of the objective space that is weakly dominated by an approximation set. The reader is referred to [KTZ06; ZKT08; Zit+03] for more formal definitions and for further considerations on the performance assessment of multi-objective optimization algorithms.

4.1.2 Evolutionary Multi-Objective Algorithms

As mentioned previously, the Pareto set/front is generally impossible to enumerate in an exact manner and/or in a reasonable amount of time, though some exact techniques exist in the literature, e.g., [Vis+98; SSP06]. In our work, we focus on the design and analysis of evolutionary heuristic search algorithms [Deb01; CLV07]. Evolutionary Multiobjective Optimization (EMO) approaches have been in fact proved to be extremely effective due the high quality of solutions they are able to compute, and also due to their adaptability to a wide spectrum of applications. These approaches have solid foundations and come with different classes containing multiple algorithms that are being continuously developed and enhanced. Generally speaking, most multi-objective evolutionary algorithms follow a standard algorithmic scheme where a *population* of solutions is evolved in an iterative manner. An *external* archive, of bounded or unbounded size, gathering all non-dominated solutions found so-far during the search, might be additionally maintained. The output is then either the population it-self, the archive, or a subset of non-dominated solutions, depending on the optimization context. During the search process, specific *mating selection* and *replacement* mechanisms are then used in order to respectively (i) choose some parents from which new individuals are created using some variation *operators*, and (ii) then update the population (and the archive) by incorporating the newly generated solutions. Apart from the variations operators used to generate new individuals, the dynamics of an evolutionary process in terms of convergence towards the Pareto front and diversity of computed solutions is tightly related to how the selection and replacement are designed. We can distinguish three main classes of approaches as described in the following.

PARETO DOMINANCE BASED APPROACHES. Algorithms from this family mostly rely on a dominance relation in order to 'compare' different solutions at the selection and replacement steps. For instance, we can cite the well-established and popular NSGA-II [Deb+02] algorithm which uses non-dominating sorting in order to rank the solutions in the population, as well as other local search approaches [Lie+09] such as Pareto Local Search (PLS) [PCS04; LTZ04]. For the sake of illustration, the high level code of a basic PLS algorithm is given in the template of Algorithm 7: from the current archive (containing only currently non-dominated solutions) a non-visited solution is selected, its neighbors according to a neighborhood structure are evaluated, and those that are non-dominated are simply added to the archive, and so on, until the neighborhood of all solutions in the archive are explored.

INDICATOR BASED APPROACHES. Algorithms in this family rely on a quality indicator in order to guide the multiobjective search. The goal is in general to converge towards a solution set optimizing the underlying indicator. The so-called Indicator-Based Evolutionary algorithm (IBEA) [ZK04] is one of the most representative algorithm in this family together with the so-called SMS-MOEA [BNE07b]. For the sake of illustration, we provide in Algorithm 8 a high level algorithmic template. Notice that to a solution x is attributed a real-valued

Algorithm 7: Pareto Local Search (PLS) using an unbounded archive.

```
1  x_0 ← initial (non-visited) solution;
2  A ← {x_0};
3  repeat
4  |  Select a non-visited solution x ∈ A;
5  |  Explore neighboring solutions in N(x) and mark x as visited;
6  |  A ← replace A with non-dominated solutions from A ∪ N(x);
7  until All solutions are visited;
```

Algorithm 8: Indicator-Based Evolutionary Algorithm (IBEA). The 'fitness' of a solution is defined by function h using an indicator-based contribution I, e.g., in IBEA, $h(x) = \sum_{x' \in P \setminus \{x\}} (-e^{-I(x',x)/\kappa})$, κ a constant, and I relates to the epsilon indicator.

```
1  P ← initial population ;
2  repeat
3  |  P' ← Select solutions from P;
4  |  Q ← generate new solutions from P' using a variation operator;
5  |  Evaluate the solution in Q using the fitness function h;
6  |  P ← Replace P by solutions from P ∪ Q;
7  until stopping condition is satisfied;
```

Algorithm 9: MOEA/D. μ (single-objective) subproblems $g(.|\lambda_i)$ are defined with respect to every search direction (weight vector) λ_i.

```
1  (λ^1, ..., λ^μ) ← generate μ initial weights/directions;
2  ∀i ∈ {1 ... μ}, B(i) ← the T-neighborhood of sub-problems i, i.e., the closest weights to λ_i;
3  (x^1, ..., x^μ) ← generate initial population of size μ;
4  repeat
5  |  for i ∈ {1 ... μ} do
6  |  |  Select x and x' randomly in {x_j : j ∈ B(i)};
7  |  |  y ← mutation_crossover_repair(x, x');
8  |  |  for j ∈ B(i) do
9  |  |  |  if g(y|λ_j) is better than g(x_j|λ_j) then
10 |  |  |  |  x_j ← y;
11 until stopping condition is satisfied;
```

number $h(x)$ which depends on an indicator function I. This function is to be viewed as associating a fitness value to every individual and hence allowing to compare them, and to eventually prefer one over the other at the selection and replacement steps.

DECOMPOSITION BASED APPROACHES. In our work, we are mainly concerned by this family of approaches, also referred to as aggregation-based or scalarizing-based. They rely on the reformulation of the original multi-objective problem in a number of single- or multi- objective smaller (sub)-problems that are solved either cooperatively or independently. Among the different existing algorithms, e.g., cMOGA [MIG01], MOGLS [Jas02; IM98], MSOPS [Hug03], etc, the so-called Multiobjective Optimization Evolutionary Algorithm based on Decomposition (MOEA/D) [ZL07] is one of the most popular algorithms that has been attracting a lot of attention from the community in the last few years. In the rest of this section, we provide a more throughout description of MOEA/D for which the basic template is given in Algorithm 9.

The idea of MOEA/D is to cooperatively search for good-performing solutions in multiple regions of the Pareto front by decomposing the multi-objective problem into a number of *scalarized* single-objective sub-problems. Many different scalarizing functions [Mie99] can

be used for this purpose. Popular examples are the weighted sum (*W*) and the weighted Tchebycheff (T) functions defined below:

$$WS(x,\lambda) = \sum_{i=1}^{m} \lambda_i \cdot f_i(x) \quad , \quad T(x,\lambda) = \max_{i\in\{1,\dots,m\}} \lambda_i \cdot \left|z_i^\star - f_i(x)\right|$$

where $\lambda = (\lambda_1,\dots,\lambda_m)$ is a weighting coefficient vector such that $\lambda_i \geqslant 0$ for all i, and $z^\star = (z_1^\star,\dots,z_m^\star)$ is a *utopian* point dominating all other points, i.e., $\forall i, \forall x, z_i^\star > f_i(x)$. T (resp. WS) is to be minimized (resp. maximized).

Given such a scalar function g, and for each sub-problem $i \in \{1,\dots,\mu\}$, the goal of MOEA/D is to approximate the solution x with the best scalarizing function value $g(x,\lambda^i)$. For that purpose, it maintains a population $P = (x^1,\dots,x^\mu)$, each individual corresponding to a good-quality solution for one sub-problem. For each sub-problem $i \in \{1,\dots,\mu\}$, a set of *neighbors* $B(i)$ is defined, which is typically the set of T closest weighting coefficient vectors using the euclidian distance. To evolve the population, subproblems are optimized cooperatively based on this neighborhood relation. At a given iteration corresponding to one sub-problem i, two solutions are selected at random from $B(i)$, and an offspring solution y is created by means of variation operators (e.g., mutation, crossover, repair functions, etc). Then, for every sub-problem $j \in B(i)$, if y improves over j's current solution x^j then y replaces it, i.e., for $g = T$, if $g(y,\lambda^j) < g(x^j,\lambda^j)$ then set $x^j = y$. The algorithm continues looping over sub-problems, optimizing them one after the other, until a stopping condition is satisfied.

4.1.3 MOEA/D: Motivation, Challenges and Literature Overview

GENERAL MOTIVATION. Most of the work described in this chapter deals with the design and analysis of decomposition based algorithms, in particular within the MOEA/D framework. Our motivation for adopting such an approach can be explained from different orthogonal perspectives. First, one of the most appealing and distinctive feature of the MOEA/D framework is its simplicity. Simplicity, in addition of being a desirable aspect in general, leads in the case of decomposition to both efficient and effective solving procedure. In fact, decomposition follows the relatively simple and well-established "divide-and-conquer" paradigm. Breaking a multi-objective problem into several '*smaller*' sub-problems solved in a *cooperative* manner is thus a natural outcome to reduce the complexity and the difficulty inherent to the tackled problems. For instance, compared to other dominance or indicator based approaches, the computational flow of MOEA/D does not require sophisticated or computationally expensive operations, such non-dominated sorting, hypervolume computations, etc. Moreover, decomposition enables parallelism, in the sense that the computations inherent to different sub-problems could be intuitively distributed over a possibly large-scale computing environment. As such, the distributed nature of a decomposition-based approach, when dealing with large-scale or computationally intensive problems, opens novel opportunities for designing new parallel solving algorithms that can be deployed over a real massively parallel platform with reduced re-engineering efforts for practitioners. Considering 'smaller' subproblems also allows to incorporate the lessons learnt from the past more easily in a smart manner. This increases the flexibility of the designed approaches, as can be witnessed by the numerous versions and extensions of MOEA/D for a wide range of optimization problems and scenarios; see e.g. [MOE]. For instance, existing evolutionary operators, as well as machine learning techniques, which are extensively studied in the single objective setting, could be wisely coupled within a decomposition-based optimization approach, either to ease the solving of each sub-problem and to guide the search in a high-dimensional space, or to help identifying the best possible interactions and coordination rules between the involved sub-problems. Of course, decomposition comes with other challenges that have to be accurately addressed.

GENERAL CHALLENGES. Generally speaking, three main challenges are addressed in our work as discussed in the following paragraphs. As for any other optimization technique, setting up an effective decomposition-based approach relies on the design and integration of several components that can be configured in many different ways. The specification of these components and their combination is crucially important. We can classify decomposition components into two high-level (and coarse-grained) categories. The first one enables to define how to decompose the original problem into a set of sub-problems both in the objective and in the variable space, thus specifying the regions where it would be more interesting to focus the search for promising solutions. The components from the second category enable to effectively solve the sub-problems and to guide the search within each of the so-defined region. The first central challenge addressed in our work is the coordination of these two categories of components, their joint specification, and the interaction between the computations occurring at the different regions defined by the decomposition in order to reduce the computational cost of the underlying global solving procedure while improving its effectiveness.

As mentioned before, and since the solving process is by essence distributed among different cooperating sub-problems, decomposition also implies the distribution of the underlying computations among possibly large-scale computing resources. Nevertheless, considering a computing platform as a simple physical medium to be used in a straightforward manner can inevitably result in incompatibility issues between the designed algorithms and their effective deployment and parallelization. A more appropriate approach is thus to think the distribution of the computational flow till the beginning at the time the different components of a decomposition-based multi-objective optimization algorithm are specified, which constitute the second general challenge addressed in our work.

Finally, we are also interested in a more global approach where decomposition is one choice that can eventually complement several other possible ones. The challenge is then to be able to better understand what makes decomposition different from other approaches and tools, and for which optimization problems or scenarios. The ultimate (global) goal of our work is in fact to strengthen the existing solving procedures, optimization methodologies, algorithms, etc, and to improve their efficiency, quality, robustness, etc.

In the rest of this section, we provide a more focused literature overview in an attempt to better position our contributions all along the large panorama of studies on decomposition and the MOEA/D framework in particular.

BRIEF HISTORY AND LITERATURE OVERVIEW. The origins of the original MOEA/D framework [ZL07] go back to the cellular multi-objective genetic algorithm (C-MOGA) and its related variants [MIG00a; MIG00b]. The main differences rely in the fact that the MOEA/D framework allowed to (i) strengthen the concept of neighboring cooperating subproblems, by explicitly addressing the issue of local neighborhood-based mating selection and replacement, and (ii) to point in a more comprehensive way the importance of the scalarizing functions used to define the cooperating subproblems and to guide the search accordingly. Following these two aspects, several other studies followed, motivated by the success and effectiveness of such a framework, e.g., a variant of MOEA/D was the winner of the CEC 2009 conference competition dealing with continuous multi-objective benchmark instances [ZLL09; LZ09]. A recent comprehensive survey can be found in [Tri+17], where different lines of research along the MOEA/D framework are classified and discussed thoroughly. Following the classification of [Tri+17], we are mainly concerned with the following four critical aspects: (i) the weight vector specification which is related to the decomposition methodology, (ii) the mating selection and replacement mechanisms, (iii) the computational effort underlying the MOEA/D algorithm, and (iv) the design and integration of variation operators. In the following, some related work are highlighted very briefly in order to allow the reader to better position our work and contributions.

In the conventional MOEA/D algorithm, some initial weighted vectors need to be generated. Standard techniques using a simplex-lattice design [DD98] are not always appropriate, since they may guide the search to a non-uniform distribution of the solutions along the Pareto front. Alternative methods were hence considered for this purpose, e.g., generalized decomposition [GPF13], two layer approaches [Li+15b], and others [Zap+15]. Besides the standard weighted sum and Tchebychev scalarizing functions, other types of scalarizing approaches were also considered, e.g., penalty based schemes [Sat15a; YJJ17], angle based schemes [Che+16]. More importantly, it was shown that adapting the weighting functions/vectors can lead to substantial improvements. For instance, mixing different scalarizing functions simultaneously and adaptively is considered in [Ish+10; Ish+09]. In [WZZ15], the impact of using the L_p norm is analyzed, and an online adaptive approach to estimate an accurate value of the p parameter is derived in [WZZ16]. Moreover, alternative decomposition strategies using region decomposition [Wan+16; LGZ14] and reference vectors [Che+16] were recently proposed in order to derive enhanced convergence and diversity search dynamics. In our work, we consider to conduct an in-depth analysis of using different scalarizing functions in decomposition, and provide a alternative perspective to understand the fundamental difference between different ones. Weight adaptation is also one of our concerns and we address this issue form a distributed local perspective.

Tightly related to this aspect, extensive research was conducted in order to study, improve and extend, the basic mating selection and replacement mechanisms of MOEA/D. In fact, we remark that mating selection within MOEA/D is performed exclusively among neighbors. Assuming that nearby sub-problems have similar solutions, the neighborhood size is critical for an accurate exploration/exploitation balance. Moreover, the replacement mechanism can lead to a situation where several neighbors are replaced by the same offspring. It can also lead to the situation where a solution which does improve a subproblem not belonging to the neighborhood, is discarded. This can imply a loss in both diversity and convergence, and likely a loss in performance. Different techniques can be adopted to deal with such issues, by typically adapting the neighborhood size, adjusting the neighborhood with respect to the selected subproblem, restricting the number of replacements, using different neighborhoods, etc. Among the different variants that one can find, a notable one called MOEA/D-DE, can be considered as a state-of-the-art [LZ09], and was extensively studied in the context of continuous and difficult benchmark functions using the well-established differential evolution operator as a crossover. Two simple modifications to basic MOEA/D are actually considered in MOEA/D-DE. The first modification uses an extra probability parameter δ allowing parents to be selected from the whole population instead of solely the T-neighborhood. The second one uses an extra parameter nr to bound the number of neighbors that can be replaced by a newly generated offspring. In our work, we consider different novel selection and replacement strategy to improve the behavior of MOEA/D while highlighting in a more comprehensive manner what makes one strategy better than the other. We also designed selection and replacement to be compatible with a distributed design.

The third critical aspect in MOEA/D is the specification of the computational effort devoted to the solving of every single subproblem. In the original MOEA/D variant, all subproblems are in fact treated equally. This might lead to some issues, typically when the Pareto front has a complicated shape, or when different regions of the Pareto front have different degrees of difficulty [Qi+14; ZLL09; Li+15a]. This is actually tightly related to the specification of weight vectors. Generally speaking, two kind of approaches are used: dynamically allocating the computational resources in terms of the number of function evaluations attributed to each subproblem, and adapting the weight vectors themselves to better guide the search. Besides, when the function evaluation is expensive, different studies can be found on the use of surrogates and meta-models to speed-up the search, e.g., [ZC13; Zha+10]. In our work, we mainly addressed the question of how to distribute the computational flow of MOEA/D over different parallel processes, and how to design novel variants allowing a high level parallelism. This is actually one research issue that was considered scarcely in the past.

The last critical aspect is the specification of the variation operators used to create new candidate solutions. There were in this respect several studies both to incorporate existing widely used operators and techniques (differential evolution, local search, particle swarm, ant colony, estimation of distribution, etc) and also to design improved variants that can take benefits from the cooperative process of solving neighboring sub-problems, e.g., see [Tri+17]. Some work also considered to study the combination of different operators, and also the adaptive on-line selection [Li+02] or off-line tuning [BLS15] of search operators within MOEA/D. The interesting observation here is that the design of accurate variation operators and their intelligent combination is essential when tackling different problem instances or domains. This is also correlated with the specification of the other components such as the choice of the scalarizing function and the mating selection and replacement, which all together allow to efficiently improve the population quality. In our work, we also worked actively in incorporating novel variation operators both for combinatorial domains, where the community still lacks much knowledge on a systematic way of handling decomposition, and also for continuous domains where we were mainly interested by machine learning inspired techniques.

In the rest of this chapter, we shall provide a more detailed description of our contributions covering explicitly or implicitly most of the previously discussed aspects. Our contributions are actually structured following three lines. In Section 4.2, we focus on weight vector specification, mating selection and replacement, in a standard sequential setting. In Section 4.3, we focus on the benefits of the high level parallelism allowed by decomposition. In Section 4.4, we address the design and the incorporation of search operators.

4.2 CONTRIBUTION #1: DESIGN COMPONENTS OF DECOMPOSITION BASED APPROACHES

The first research line adopted in our work concerns the design and analysis of the algorithmic components that can be plugged into a decomposition based approach. In the following, we first provide an overview of our contribution on analyzing the impact of using specific scalarizing functions. Then-after, we describe our contributions on designing improved variants of MOEA/D based on novel mating selection and replacement mechanisms.

4.2.1 On the Impact of the Scalarizing Functions

Let us recall that there exist many ways of *decomposing* a multiobjective optimization problem using a (set of) single-objective *scalarizing functions*, including the prominent examples of *weighted sum* (WS), *weighted Chebychev* (T), or *augmented weighted Chebychev* (S_{aug}) [Mie99]. For most of them, theoretical results, especially about which Pareto-optimal solutions are attainable, exist [Mie99; Kaloo] but they are typically of too general nature to allow for statements on the actual search performance of (stochastic) optimization algorithms. Within the MOEA/D framework, such functions were mostly studied by evaluating their ability to provide a good approximation set and by comparing the relative performances of the underlying multi-objective algorithms, e.g., [IAN13; Sat15b]. In our work [Der+14a], we instead abstract away from any particular scalarizing function, and rather focus in understanding which general properties of them influence the search behavior of EMO algorithms. We argue by means of experimental investigations that it is not the actual choice of the scalarizing function or their parameters that makes the difference in terms of performance, but rather the general properties of the resulting lines of equal function values. To this end, and

Table 3: Overview of some scalarizing functions, and the corresponding angles of the lines of equal function values with the standard Pareto dominance cone.

scalar function	S_{gen} parameters	opening angles	reference
$WS(z) = w_1\|z_1^* - z_1\| + w_2\|z_2^* - z_2\|$	$\alpha = 0, \epsilon = 1$	$\theta_1 = \arctan\left(-\frac{w_1}{w_2}\right)$ $\theta_2 = \frac{\pi}{2} + \arctan\left(\frac{w_1}{w_2}\right)$	[Mie99]
$T(z) = \max\{\lambda_1\|z_1^* - z_1\|, \lambda_2\|z_2^* - z_2\|\}$	$\alpha = 1, \epsilon = 0$	$\theta_1 = 0$ $\theta_2 = \pi/2$	[Mie99]
$S_{aug}(z) = T(z) + \epsilon\,(\|z_1^* - z_1\| + \|z_2^* - z_2\|)$	$\alpha = 1$, $w_1 = w_2 = 1$	$\theta_1 = \arctan\left(-\frac{\epsilon}{\lambda_1+\epsilon}\right)$ $\theta_2 = \frac{\pi}{2} + \arctan\left(\frac{\epsilon}{\lambda_2+\epsilon}\right)$	[Mie99]
$S_{norm}(z) = (1-\epsilon)T(z) + \epsilon\, WS(z)$	$\alpha = 1 - \epsilon$, $w_i = 1/\lambda_i$	$\theta_1 = \arctan(-\frac{\epsilon w_1}{(1-\epsilon)\lambda_2+\epsilon w_2})$ $\theta_2 = \frac{\pi}{2} + \arctan(\frac{\epsilon w_2}{(1-\epsilon)\lambda_1+\epsilon w_1})$	[Der+14a]

for problems with 2 objectives, we consider the following general scalarizing function that covers the special cases of WS[1] , T, and S_{aug} functions:

$$S_{gen}(z) = \alpha \cdot \max\{\lambda_1 \cdot |z_1^* - z_1|, \lambda_2 \cdot |z_2^* - z_2|\} + \epsilon\,(w_1 \cdot |z_1^* - z_1| + w_2 \cdot |z_2^* - z_2|)$$

where $z = (z_1, z_2)$ is the objective vector of a feasible solution, $z^* = (z_1^*, z_2^*)$ a utopian point, $\lambda_1, \lambda_2, w_1$, and $w_2 > 0$ scalar weighting coefficients indicating a search direction in objective space, and $\alpha \geqslant 0$ and $\epsilon \geqslant 0$ parameters to be fixed. For more details about the mentioned scalarizing functions and their relationship, we refer to Table 3 where we consider a case of S_{gen} that combines WS and T with a single parameter ϵ: the normalized $S_{norm}(z) = (1-\epsilon)T(z) + \epsilon WS(z)$ where $\alpha = 1 - \epsilon$ and $\epsilon \in [0, 1]$.

One important property of a scalarizing function turns out to be the shape of its sets of equal function values, that is the curve in the objective space where all points of the curve have the same scalar value with respect to the considered configuration of the scalarizing function. The following proposition, leveraging previous work [Mie99], states that these equi-function-values are given by two straight lines characterized by the opening angles θ_i they form with the f_1-axis, as illustrated in Fig. 16.

Proposition 1 *Let z^* be a utopian point, $\lambda_1, \lambda_2, w_1$, and $w_2 > 0$ scalar weighting coefficients, $\alpha \geqslant 0$ and $\epsilon \geqslant 0$, where at least one of the latter two is positive. Then, the polar angles between the equi-utility lines of S_{gen} and the f_1-axis are $\theta_1 = \arctan(-\frac{\epsilon w_1}{\alpha\lambda_2+\epsilon w_2})$ and $\theta_2 = \frac{\pi}{2} + \arctan(\frac{\epsilon w_2}{\alpha\lambda_1+\epsilon w_1})$.*

We are able to provide empirical evidence on the fact that the dynamics of the search process is rather 'independent' of the scalarizing function under consideration or its parameters. Instead, we show that the search process is guided by the positioning of the lines of equal function values in the objective space—described by the opening angle, i.e., θ_1 and θ_2. Using the ρMNK landscapes [Ver+13; AT07] as a benchmark problem, Fig. 16 shows three typical exemplary executions of a single objective $(1+\lambda)$-EA, endowed a standard bit-flip mutation, and optimizing the single objective scalar function S_{gen}. The typical initial solution maps around the point $z = (0.5, 0.5)$ in the objective space, which is the average objective vector for a random solution of ρMNK landscapes. One can see that the evolution of the current solution can be explained by the combination of two effects. The first one is given by the bit-flip mutation operator, that produces more offspring in a particular direction compared to the other ones, due to the underlying characteristics of the ρMNK landscapes under consideration. The second one is given by the lines of equal function values, i.e., the current solution moves perpendicular to the equi-fitness lines, following the gradient direction in the objective space. We can remark that the search process is mainly guided by the lower part of the cones of equal function values when the direction is above the initial solution, and *vice versa*. When the direction angle δ is smaller (resp. larger) than $\pi/4$, the dynamics of the search process are better captured by the opening angle θ_1 (resp. θ_2). Geometrically, the

1 Contrary to the standard literature, our formalization assumes minimization and we therefore have included the utopian point $\bar{z}$ that is typically assumed to be $\bar{z} = (0, 0)$ for minimization.

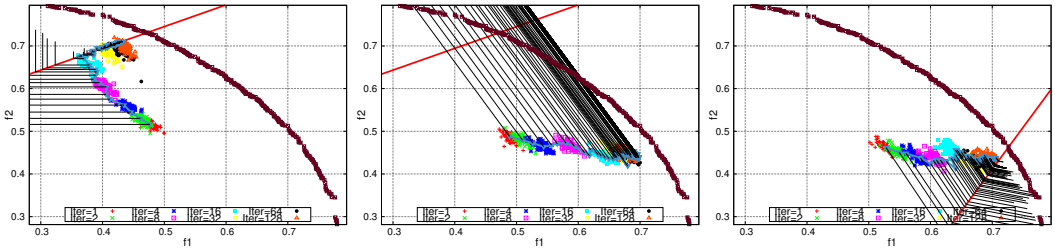


Figure 16: Exemplary runs of a $(1+\lambda)$-EA for different direction angles δ (straight line) and different ϵ-values using S_{norm} and ρMNK landscapes ($\rho = -0.7$). Shown are the best known Pareto front approximation, the offspring at some selected generations, the evolution of the parent, and the lines of equal function values. Left: $\epsilon = 0$, $\delta = \frac{3}{10} \cdot \frac{\pi}{2}$. Middle: $\epsilon = 1$, $\delta = \frac{3}{10} \cdot \frac{\pi}{2}$. Right: $\epsilon = 0.6$, $\delta = \frac{7}{10} \cdot \frac{\pi}{2}$.

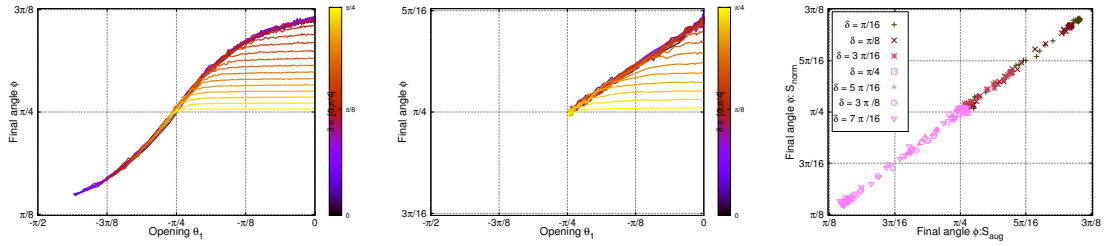


Figure 17: Left (resp. Middle): scatter plots showing final angle $\phi(\epsilon)$ and opening $\theta_1(\epsilon)$ for $\rho = -0.7$ and S_{norm} (resp. S_{aug}). Every color is for a fixed δ and variable ϵ. Right: Scatter plot showing $(\phi(S_{norm}), \phi(S_{aug}))$.

optimal solution with respect to a scalarizing function correspond to the intersection of one of the 'highest' lines of equal fitness values in the gradient direction and the feasible region of the objective space. Although the above description is mainly intuitive, a more detailed analysis can support this general idea.

In Fig. 17, we show the scatter plot of the final angle ϕ, that a solution found by the $(1+\lambda)$-EA forms with the f_1-axis, as a function of the opening angle θ_1 for different direction angles[2] $\delta \in [0, \pi/4]$. The scatter plot gives a set of values $(\theta_1(\epsilon), \phi(\epsilon))$ for different ϵ-values in S_{norm} and in S_{aug}. From Proposition 1, for a given direction angle δ, the opening angle θ_1 belongs to the interval $[\delta - \pi/2, 0]$ for S_{norm}, and to the interval $[-\pi/4, 0]$ for S_{aug}. Independently of the scalarizing function, when the direction angle is between 0 and around $3\pi/16$ (blue color), the value of ϕ is highly correlated with the opening angle θ_1. For such directions, a simple linear regression confirms this observation and allows us to explain the relation between the opening angle and the final angle. Interestingly, this is independent of the definition of the scalarizing function, and depends mainly on the property of the lines of equal function values. This tells us that the lines of equal fitness values are guiding the search process following the gradient direction given by the opening angle in the objective space. In Fig. 17 (Right), we can see that the obtained final angles are equivalent when the opening angle is the same, even for different direction angles and/or scalarizing functions. In fact, we observe that the final angles obtained are very similar for the scalarizing functions S_{norm} and S_{aug} if δ is the same for both functions and the ϵ-values are chosen in order to have matching opening angles. Whatever the δ- and ϵ-values, the points are close to the line $y = x$, which shows that independently of the scalarizing function, the final angle is strongly correlated to the opening angle, and not to a particular scalarizing function.

2 Roughly speaking, the direction angle is the angle between the line defined by the weight vector in the objective space and the f_1-axis.

To summarize, our empirical investigations allows us to show that it is the opening of the lines of equal function values that explicitly guides the search towards a specific region of the objective space. In particular, when combining multiple scalarizing search processes to compute a whole approximation set, these lines play a crucial role to achieve diversity. While our results are in some sense natural and intuitive and consider simple search procedures, they make a fundamental step towards strengthening the understanding of the properties and dynamics of more complex algorithmic settings. They also rise new interesting issues that were hidden by the complex design of well-established algorithms. For instance, we are able to highlight the importance of a *non-uniform* configuration of scalarizing functions (i.e., different parameters for different weighting coefficient or in different search direction) which, to the best of our knowledge, was not considered so-far. Moreover, other types of opening angles can be directly defined without necessarily using a particular scalarizing function. For instance, this can offer more flexibility when tuning decomposition-based algorithms, e.g., defining the opening angles without being bound to a fixed closed-form definition, but adaptively, with respect to the current search state. In particular, this suggests that common techniques and paradigms in on-line and off-line parameter setting are worth to be investigated to set the opening angles, instead of designing new scalarizing functions.

4.2.2 Improving Mating Selection and Replacement in MOEA/D

A GENERATIONAL MOEA/D APPROACH. As mentioned in the related work section, MOEA/D could suffer from a lack of diversity due to the locality of its selection and replacement mechanism. In [Mar+14], we show that this can also be caused by the fact that in MOEA/D, sub-problems are optimized iteratively in a sequence. As can be seen in the template of Algorithm 9, parents are selected randomly from the neighborhood of the sub-problem being processed. Thus, it might happen that a solution with the potential of producing a good offspring, gets never selected for reproduction. Additionally, because a neighbor's solution might be replaced as soon as a better offspring is found, this solution gets actually no chance to survive in the population. To increase the chance for a solution to survive in the population, we investigate the idea of evolving the whole population *simultaneously* by optimizing all subproblems in a generational manner as detailed in Algorithm 10.

Contrary to MOEA/D where a single offspring is generated at each iteration, our framework is basically a $(\mu + \mu)$-EA where the first stage consists in generating μ offsprings and the second stage consists in updating the whole population for the next round. The first stage corresponds to mating selection where *one* new offspring is created for *every* subproblem. Specifically, we consider two alternatives: (i) either the solution of the current subproblem is *always* selected to be a parent and hence included for variation ($x = s$), or (ii) parents are picked randomly from neighbors in the usual way MOEA/D proceeds ($x = c$). Here, s refers to a *Selfish* policy, whereas s refers to a *Collective* strategy. Only when all subproblems are treated and all μ new offspring solutions are created, the second stage of replacement occurs. In this stage, the subproblems are processed iteratively and we again consider two alternatives: (i) either the solution of a subproblem is compared to the offspring created at this subproblem ($y = s$), or (ii) the solution of the current subproblem is compared to the offsprings created in all neighboring subproblems ($y = c$). Notice that Algorithm 10 is fully compatible with MOEA/D and some other variants as in [LZ09], i.e., parameters δ and nr.

By combining the different selfish and collective generational selection and replacement policies, we are able to show substantial improvement over MOEA/D[3], as illustrated in Fig. 18 rendering the "anytime" behavior of designed algorithms for different ρMNK landscapes having different objective correlations. In particular, we can see that all combinations are able to make improvements, with MOEAD-**sc** and MOEAD-**cc** being consistently better than MOEA/D. We also remark that MOEAD-**sc** and MOEAD-**cc** are more systematically improving

3 Notice that [LZ09] was the winner of the CEC 2009 competition, and is considered as a reference state-of-the-art variant of MOEA/D for difficult continuous benchmark problems.

Algorithm 10: MOEAD-xy ($x, y \in \{s, c\}$), a generational MOEA/D.

Input: $\{\lambda^1, \cdots, \lambda^\mu\}$: weight vectors w.r.t sub-problems; g: a scalarizing function; $B(i)$: the neighbors of sub-problem $i \in \{1, \cdots, \mu\}$; $P=\{p^1, \cdots, p^\mu\}$: the initial population.

```
while Stopping Condition do
    for i ∈ {1, ··· , μ} do
        if rand(0, 1) < δ then B_i ← B(i) ;                          /* Neighborhood Setting */
        else B_i ← P ;
        if x = s then                                                /* Selfish mating selection */
            k ← i;
        else if x = c then                                           /* Collective mating selection */
            k ← rand(B_i);
        ℓ ← rand(B_i); while ℓ = k do ℓ ← rand(B_i);
        if rand(0, 1) < cr then                                      /* Variation operators */
            o^i ← crossover(p^k, p^ℓ); o^i ← mutation(o^i);
        else o^i ← mutation(p^k) ;
        if o^i is infeasible then repair(o^i);
    for i ∈ {1, ··· , μ} do c_i ← 0;
    for i ∈ {1, ··· , μ} do                                          /* Environmental replacement */
        if y = s then                                                /* Selfish replacement */
            p' ← o^i;
            if g(p', λ^i) better than g(p^i, λ^i) then p^i ← p' ;
        else if y = c then                                           /* Collective replacement */
            shuffle(B_i);
            for j ∈ B_i do
                p' ← o^j;
                if c_j < nr then
                    if g(p', λ^i) better than g(p^i, λ^i) then p^i ← p'; c_j ← c_j + 1 ;
```

upon MOEAD-ss for test instances having conflicting objectives; whereas MOEAD-ss is able to outperform its competitors as the objective correlation gets higher. Actually, we can show that our strategies induce different intensification/diversification trade-offs both at the local level of every single-objective scalarized subproblem; but also at a more global level when considering the whole approximation set. When a selfish (resp. collective) mating selection is considered, the probability that a solution in the population gets selected for reproduction is 1 (resp. $1-(1-1/T)^T$). This means that all our strategies imply diversified offsprings since no solution in the current population gets replaced before exploring its potential. At the replacement stage, if a collective strategy is adopted, then the single-objective search at every subproblem is intensified since the probability that a locally improving solution can be found is higher. But this might increase the number of copies in the current approximation set. When a selfish replacement is considered, it is more likely that the number of copies is minimized; but at the price of delaying the advance of the population towards the front. For correlated objectives, and since the front is not too large, it is sufficient that only few solutions are able to approach the front in order to get good overall performance. Thus, a selfish replacement can be accurate. This is not the case for anti-correlated objectives where both the local improvements and the global spread of solutions are crucial.

Besides, we consider in [AD15] to study the behavior of the MOEA/D framework, including our generational designed approach, when applied to other sophisticated combinatorial optimization problems. Our goal is to gain a more in-depth understanding of the benefits and limitations of decomposition-based evolutionary algorithms, compared to other approaches, such dominance based algorithms, and to provide enhanced algorithmic components allowing for optimal performance. In the rest of this section, we provide a bench of results we are able to obtain, and we describe an alternative replacement strategy which is shown to be highly effective. All results in the rest of this section are with respect to the Fuzzy Job Shop Scheduling Problem (FJSP) with two objectives for which a formal definition and related work can be found in [AD15].

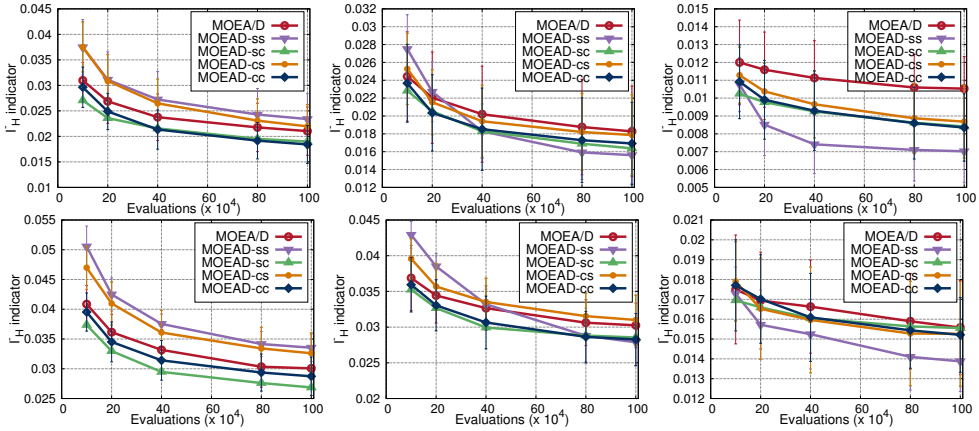


Figure 18: Convergence plots of MOEAD-xy w.r.t. hypervolume difference and for different ρMNK landscapes (with fixed bit string size $N = 128$ and two objectives $M = 2$). Columns are respectively for instances with objective correlation $\rho \in \{-0.7, 0.0, 0.7\}$. Rows are respectively for instances with a degree of non-linearity $K \in \{4, 8\}$. $\mu = 128$, $T = 8$, $\delta = 1.0$, $cr = 1.0$, $nr = 0$ and $g = T$ in Algorithm 10.

Table 4: Comparing MOEA/D *vs.* NSGA-II using FJSP instances. Average indicator value and the standard deviation (in parentheses, with a multiplicative factor of 10^{-1}) are reported. Gray cells indicate that the algorithm in the corresponding column is significantly better than the other one using a statistical t-test with confidence 0.05.

instance	Hypervolume Indicator		Epsilon Indicator	
	MOEA/D	NSGA-II	MOEA/D	NSGA-II
ABZ7	$0{,}382_{(0{,}38)}$	$0{,}423_{(0{,}32)}$	$0{,}179_{(0{,}39)}$	$0{,}141_{(0{,}32)}$
ABZ8	$0{,}373_{(0{,}31)}$	$0{,}428_{(0{,}35)}$	$0{,}218_{(0{,}33)}$	$0{,}164_{(0{,}36)}$
ABZ9	$0{,}349_{(0{,}44)}$	$0{,}415_{(0{,}33)}$	$0{,}244_{(0{,}47)}$	$0{,}180_{(0{,}35)}$
FT10	$0{,}549_{(0{,}48)}$	$0{,}591_{(0{,}39)}$	$0{,}199_{(0{,}50)}$	$0{,}163_{(0{,}40)}$
FT20	$0{,}203_{(0{,}24)}$	$0{,}241_{(0{,}18)}$	$0{,}110_{(0{,}23)}$	$0{,}076_{(0{,}18)}$
La21	$0{,}427_{(0{,}33)}$	$0{,}449_{(0{,}37)}$	$0{,}158_{(0{,}34)}$	$0{,}142_{(0{,}40)}$
La24	$0{,}429_{(0{,}36)}$	$0{,}456_{(0{,}39)}$	$0{,}181_{(0{,}38)}$	$0{,}162_{(0{,}40)}$
La25	$0{,}440_{(0{,}30)}$	$0{,}445_{(0{,}32)}$	$0{,}163_{(0{,}32)}$	$0{,}165_{(0{,}33)}$
La27	$0{,}228_{(0{,}24)}$	$0{,}254_{(0{,}29)}$	$0{,}184_{(0{,}26)}$	$0{,}166_{(0{,}33)}$
La29	$0{,}235_{(0{,}31)}$	$0{,}254_{(0{,}27)}$	$0{,}201_{(0{,}36)}$	$0{,}190_{(0{,}30)}$
La38	$0{,}608_{(0{,}51)}$	$0{,}683_{(0{,}34)}$	$0{,}199_{(0{,}48)}$	$0{,}132_{(0{,}31)}$
La40	$0{,}618_{(0{,}45)}$	$0{,}691_{(0{,}36)}$	$0{,}204_{(0{,}43)}$	$0{,}145_{(0{,}35)}$

MOEA/D VS NSGA-II. Surprisingly, when comparing the performance of conventional MOEA/D to the popular (dominance based) NSGA-II algorithm [Deb+00] (See Table 4), we can see that NSGA-II is substantially outperforming MOEA/D. To fully understand this huge difference between the two approaches, we show in Fig. 19 the evolution of the average number of different solutions (in the objective space) in the population maintained by both approaches as a function of the number of function evaluations for one representative instance. We observe that NSGA-II is able to maintain more different solutions than MOEA/D in the first stages of the run; or equivalently, that the number of copies in MOEA/D population is increasing very abruptly. We clearly attribute this to the fact that as soon as a good solution is found in MOEA/D, it will immediately replace all solutions from the neighbor subproblems. Although this strategy could speed-up convergence, it obviously prevents the evolutionary operators to produce improving offsprings in diversified regions in the objective space, which provides another evidence on the lack of diversity that basic MOEA/D can suffer.

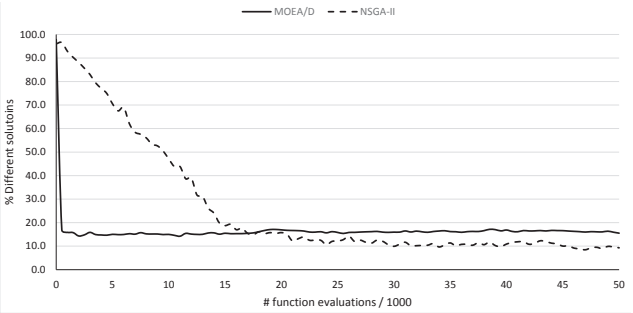


Figure 19: Evolution of the percentage of different objective vectors in the population for MOEA/D and NSGA-II.

RECURSIVE OPPORTUNISTIC REPLACEMENT. Although some existing variants of MOEA/D, e.g., MOEA/D-n_r[4] [LZ09], are able to enhance the original MOEA/D and to obtain highly competitive results compared to NSGA-II, we found that they still do not handle the diversity issue in an accurate manner. We there-by consider to study the behavior of our previously described generational approach MOEAD-**xy** [Mar+14] as well as a new variant, called MOEA/D-RO, in which population diversity is managed in a more explicit manner. More specifically, we directly check solution diversity, locally at the neighborhood level, and we manage to design a non-oblivious replacement. First, we simply do not allow an offspring to replace a solution if there already exists a solution having the same objective values in the corresponding neighborhood. Every time this condition is satisfied and the replacement is activated with respect to an offspring, say y, and a neighboring subproblem solution, say x^j, we do the following. The offspring y becomes the new current solution for subproblem j, but the previous solution x^j is not discarded if it can improve the solution of other neighbors. Hence, we recursively check whether there is an opportunity that solution x^j replaces a solution j' in the neighborhood $B(j)$ of subproblem j. If such a solution is found, a new replacement is activated and so on until no improvement is observed. Notice that with this strategy we do maintain diversity but we also attempt to improve convergence since we heuristically check whether a solution can serve for some subproblems before discarding it from the population.

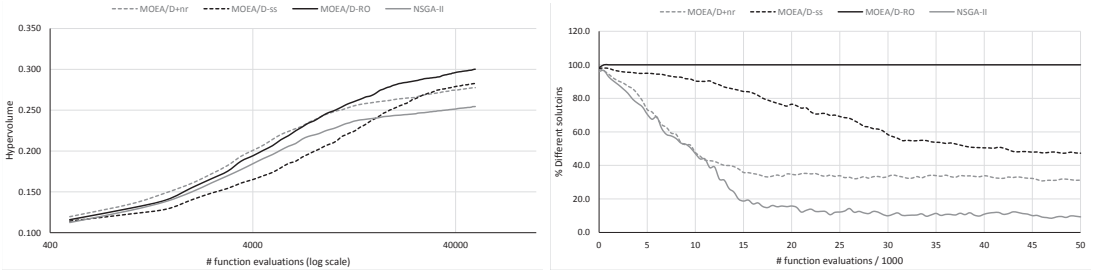
In Table 5, we can see that for most instances but one, the MOEA/D-RO strategy provides the best results, followed by our MOEAD-**ss** generational approach, for which no significant statistical difference is found with MOEA/D-RO. A huge difference between MOEA/D-RO and MOEAD-**ss** can however be elicited when examining more carefully their runtime behavior. In Fig. 20, we provide an illustration of this claim. In the first sub-figure, the Y-axis represents the average hypervolume obtained by each algorithm, while the X-axis measures the number of calls to the evaluation function (in the logarithmic scale). Overall, even though MOEAD-**ss** and MOEA/D-RO obtain similar hypervolume values at the end, we see that the evolution of both in time is different. For MOEA/D-RO, MOEA/D-n_r, and even NSGA-II, they obtain good results even with a short amount of function evaluations, while MOEAD-**ss** is much slower converging to good results. Actually, in the first stages MOEAD-**ss** performs worse than MOEA/D-n_r, but it is able to outperform it after a given number of evaluations. In the right part of Fig. 20, we see the number of different objective vectors that are maintained within the population during the evolution of the algorithms. The MOEA/D-RO has a 100% of different solutions, which is to contrast with the other considered algorithms. In our opinion, this diversity property is what makes one variant of MOEA/D better than the other depending on the available amount of computational effort.

To conclude on our contributions with respect to the mating selection and replacement of MOEA/D, we would like to comment that the very local nature of the neighborhood

4 In this variant, the number of replacement is controlled by a parameter nr, where a new solution can replace at most nr old ones in the T-neighborhood.

Table 5: Comparison of MOEA/D-RO [AD15], MOEAD-ss [Mar+14], MOEA/D-n_r [LZ09], and NSGA-II [Deb+00], using FJSP instances. ($\mu = 100$, $T = 100$, $nr = 1$, $\delta = 1$ when it holds).

instance	Hypervolume Indicator				Epsilon Indicator			
	MOEA/D-n_r	MOEAD-ss	MOEA/D-RO	NSGA-II	MOEA/D-n_r	MOEAD-ss	MOEA/D-RO	NSGA-II
ABZ7	0,435 (0,35)	0,446 (0,27)	**0,456** (0,30)	0.423 (0,32)	0,128 (0,37)	0,112 (0,28)	**0,106** (0,34)	0,141 (0,32)
ABZ8	0,446 (0,33)	0,459 (0,32)	**0,477** (0,33)	0,428 (0,35)	0,144 (0,34)	0,129 (0,34)	**0,112** (0,34)	0,164 (0,36)
ABZ9	0,425 (0,41)	0,435 (0,35)	**0,445** (0,38)	0,415 (0,33)	0,167 (0,42)	0,156 (0,38)	**0,148** (0,48)	0,180 (0,35)
FT10	0,617 (0,45)	0,634 (0,41)	**0,643** (0,42)	0,591 (0,39)	0,135 (0,46)	0,117 (0,43)	**0,109** (0,47)	0,163 (0,40)
FT20	0,238 (0,20)	**0,249** (0,20)	0,243 (0,23)	0,241 (0,18)	0,079 (0,18)	**0,068** (0,18)	0,073 (0,20)	0,077 (0,18)
La21	0,470 (0,27)	0,456 (0,29)	**0,478** (0,33)	0,449 (0,37)	0,119 (0,29)	0,132 (0,32)	**0,112** (0,34)	0,142 (0,40)
La24	0,472 (0,47)	0,491 (0,30)	**0,507** (0,36)	0,456 (0,39)	0,141 (0,48)	0,122 (0,30)	**0,108** (0,34)	0,162 (0,40)
La25	0,472 (0,35)	0,485 (0,30)	**0,494** (0,32)	0,445 (0,32)	0,136 (0,37)	0,120 (0,30)	**0,115** (0,38)	0,165 (0,33)
La27	0,270 (0,29)	0,274 (0,22)	**0,279** (0,28)	0,254 (0,29)	0,142 (0,35)	**0,134** (0,24)	0,134 (0,29)	0,166 (0,33)
La29	0,278 (0,31)	0,283 (0,32)	**0,300** (0,31)	0,254 (0,27)	0,158 (0,34)	0,148 (0,34)	**0,134** (0,35)	0,190 (0,30)
La38	0,706 (0,36)	0,709 (0,36)	**0,721** (0,30)	0,683 (0,34)	0,112 (0,31)	0,111 (0,34)	**0,101** (0,29)	0,132 (0,31)
La40	0,684 (0,41)	0,697 (0,35)	**0,726** (0,31)	0,691 (0,36)	0,147 (0,37)	0,135 (0,34)	**0,109** (0,44)	0,145 (0,35)



(a) Hypervolume evolution for La29

(b) Diversity evolution for La29

Figure 20: Runtime evolution of the hypervolume (Left) and the % of different solutions in the population (Right), using MOEA/D-RO [AD15], MOEAD-ss [Mar+14], MOEA/D-n_r [LZ09], and NSGA-II [Deb+00]), and for one representative FJSP instance.

relation defined among subproblems can imply very specific dynamics with respect to the global population. In fact, by a *wave* effect, what happens at the neighborhood level, starts spreading until eventually impacting the whole population. This simple observation is actually very difficult to elicit in a more formal or empirical manner, but its is at the heart of MOEA/D. Depending on how the wave is moving from one subproblem to another, and how aggressive it is, one can obtain different trade-offs in terms of diversity and convergence. Mating selection combined with the evolutionary operators enables to control the wave magnitude, that is how good a new solution is likely to be with respect to one subproblem when the population is fixed. Replacement combined with the scalarizing function enables to prevent the wave to vanish too quickly and to continue moving, iteration after iteration, towards interesting unexplored or efficient regions. In our opinion, being able to control this complex behavior using simple local rules is the key in designing a highly effective decomposition-based evolutionary optimization procedure.

4.3 CONTRIBUTION #2: DISTRIBUTED DECOMPOSITION-BASED APPROACHES

The second research line adopted in our work consists in exploring the decentralized nature of decomposition-based approaches. Our general goal is two-fold. On the one side, since the divide-and-conquer paradigm underlying decomposition can expose a high level parallelism, we target the design of cooperative search algorithms with improved approximation quality. On the other side, the parallelism exposed by such algorithms would eventually allow one to tackle large-scale problems, to speed-up the search and to take much benefits from the increased availability and computing power of nowadays large-scale dis-

Algorithm 11: DLBS – high level code to be executed by every distributed node i.

```
1 x^i ← initial solution corresponding to node v^i;
2 repeat
      // communicate positions
3     z^i ← (z^i_1, z^i_2) the position of solution x^i in the bi-objective space, z^i = f(x^i);
4     Send z^i to neighboring nodes;
5     Z^i ← receive neighboring positions;
      // variation
6     x ← Mutation(x^i);
      // replacement
7     if LF^{Z^i}(x) > LF^{Z^i}(x^i) then
8         x^i ← x;
9 until Stopping condition is satisfied;
```

tributed platforms. The first contribution described in this section concerns the design of novel adaptive and local rules that can dynamically adjust the weight directions used in multi-objective decomposition [Der+14b]. The second contribution is on the design of novel parallel variants of MOEA/D and the analysis of their potential in solving large optimization problems [Der+15].

4.3.1 A distributed localized and adaptive approach

In this section, we consider the design of an adaptive distributed mechanism [Der+14b] for the proper setting of the weight vectors used in decomposition based approaches. Let us notice that this is in line with other work we had on the same subject [DBL13], where weight vectors are defined as a function of the objective values of the whole population, and which is not discussed here for the sake of conciseness. A unique of the contribution presented in the following is the design of a cooperative and distributed approach which is inherently *local*; meaning that it is thought to be independent of any global knowledge, thus making it particularly suitable for a large-scale effective deployment.

DLBS: DISTRIBUTED LOCALIZED BI-OBJECTIVE SEARCH. Given a number of distributed computing nodes, our aim is to self-coordinate them locally, in order to cooperatively and adaptively search different regions of the Pareto front. The algorithm designed for bi-objective optimization problems is summarized in the high level template of Algorithm 11. For the clarity of the representation, we only consider the setting where each compute node is actually evolving one unique solution, i.e., we consider a one-to-one mapping between the distributed nodes and the solutions of the population. The nodes/population are then organized following a distributed *line* where every node/solution, except those being at the two extremes of the line, have exactly two distinct neighbors. According to this logical line structure, we design *local* rules based solely on the relative positions of neighboring solutions in the objective space. This can be viewed as an island based approach, where the information exchanged between neighboring nodes does not involve any migration. The designed rules are then based on the definition of *localized (scalar) fitness functions*, denoted $\mathcal{LF}$, to be optimized locally, and allowing every distributed node to focus on a different region of the objective space based on the position of its neighbors.

LOCALIZED FITNESS FUNCTIONS. The choice of the localized fitness function $\mathcal{LF}$ is the key ingredient of our approach. We study two alternatives allowing every distributed node to focus on the sub-region being orthogonal to the positions of its neighbors. This is with the exception of the two extreme nodes where the search is simply guided by the values of the corresponding objective value. The two designed scalar functions are summarized in Fig. 21. The first scalar function, denoted $\mathcal{LF}_{\text{OD}}$, is based on a weighted sum. Given a candidate

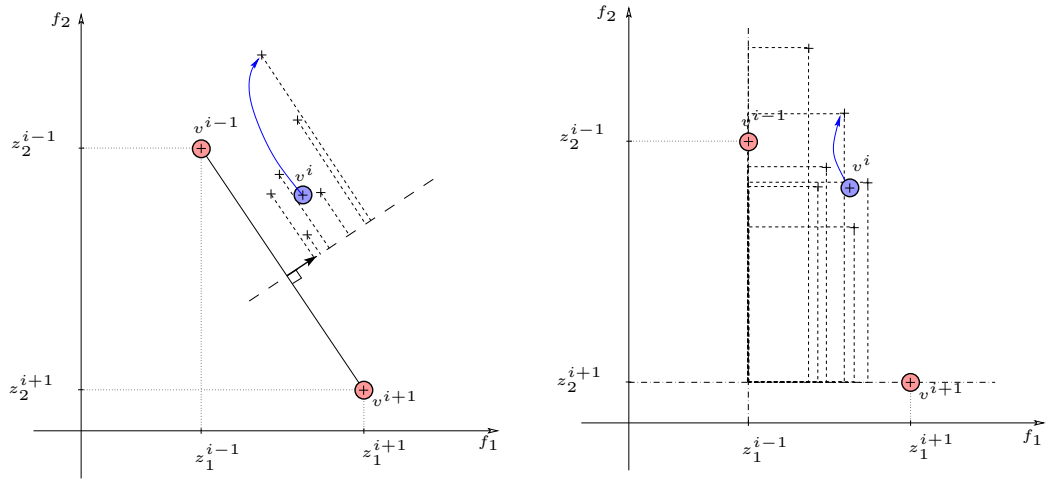


Figure 21: Illustration of the replacement mechanism in DLBsusing the localized fitness function $\mathcal{LF}_{OD}$ (left) and $\mathcal{LF}_{H}$ (right). The crosses without circle are the candidate solutions and the arrow shows the selected candidate solutions that replace the current one.

solution x at some distributed node i, it is scored according to the following function, where Z^i is the couple of neighbors' objective values.

$$\mathcal{LF}_{OD}^{Z^i}(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x);\ \text{where } w_1 = z_2^{i-1} - z_2^{i+1}, w_2 = z_1^{i+1} - z_1^{i-1}$$

Notice that notation OD stands for *Orthogonal Direction* and it is inspired by the dichotomic scheme proposed by [AN79], while being strictly local. The second scalar function, denoted by $\mathcal{LF}_H$, is based on the hypervolume indicator. The score of a candidate solution x at some distributed node i is defined as following.

$$\mathcal{LF}_{H}^{Z^i}(x) = \begin{cases} (f_1(x) - z_1^{i-1}) \cdot (f_2(x) - z_2^{i+1}) & \text{if } f_1(x) \geqslant z_1^{i-1} \text{ and } f_2(x) \geqslant z_2^{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Using the $\mathcal{LF}_H$ function, we intuitively hypothesis that by selecting candidate solutions maximizing the local hypervolume contribution at each node, the global hypervolume of the new set of solutions is likely to be better than the previous one. Notice however that only the local coordinate of neighbors in the objective space are used when defining the local hypervolume contribution at each distributed node, i.e., they serve to define a reference point. Moreover, it may happen that all solutions generated in the candidate set of one distributed node have a $\mathcal{LF}_H$-value of 0, e.g., when they are all dominated by at least one neighboring position. In this special case, we use the $\mathcal{LF}_{OD}$ function in order to avoid a random selection and make the current solutions evolving closer to the Pareto front.

The two previously defined functions are then to be maximized locally using standard evolutionary operators. They differs essentially by the lines of *local* equi-fitness values they are able to imply (see Section 4.2.1), and other localized fitness functions could have been considered as well. Our main goal is in fact to study the effectiveness of such locally defined functions in evolving the solutions distributively towards high quality approximation sets. A representative bench of results is hence discussed in the following.

SELECTED EXPERIMENTAL RESULTS. We study the performance and the behavior of the designed distributed localized functions using a broad range ρMNK landscape instances with different values for N (the bit-string size), K (the degree of non-linearity) and ρ (the objective correlation). We also consider the scalability of our approach when using a variable number

Table 6: DLBS – relative performance using the hypervolume for different ρMNK landscapes (same holds for the epsilon indicator). The value in each cell corresponds to the number of other algorithms that significantly (p-value= 0.05) outperform the algorithm in the corresponding column and for the instance given in the row. In braces, the hypervolume indicator value. $\mu = 128$.

ρ	N	K	DLBS_{OD}		DLBS_{H}		PIWS		HEMO	
−0.7	128	4	**0**	(1.846)	**0**	(1.919)	**2**	(2.365)	3	(3.667)
−0.7	128	8	**0**	(1.914)	**0**	(1.984)	**2**	(2.275)	**2**	(2.375)
−0.7	256	4	**0**	(1.529)	**1**	(1.618)	**2**	(1.779)	3	(4.190)
−0.7	256	8	**0**	(1.580)	**1**	(1.680)	**2**	(1.771)	3	(2.906)
−0.7	512	4	**0**	(0.985)	**1**	(1.107)	**2**	(1.253)	3	(3.352)
−0.7	512	8	**0**	(1.248)	**1**	(1.318)	**2**	(1.461)	3	(2.836)
0.0	128	4	**1**	(1.778)	**1**	(1.876)	3	(2.491)	**0**	(1.406)
0.0	128	8	**1**	(1.677)	**2**	(1.821)	3	(2.178)	**0**	(1.043)
0.0	256	4	**0**	(1.272)	**2**	(1.390)	3	(1.613)	**0**	(1.284)
0.0	256	8	**1**	(1.219)	**2**	(1.349)	3	(1.582)	**0**	(0.667)
0.0	512	4	**0**	(1.038)	**1**	(1.115)	3	(1.339)	**0**	(1.068)
0.0	512	8	**1**	(1.107)	**2**	(1.214)	3	(1.379)	**0**	(0.822)
+0.7	128	4	**1**	(1.518)	**2**	(1.651)	3	(2.277)	**0**	(1.255)
+0.7	128	8	**0**	(0.629)	**2**	(0.743)	3	(0.968)	**0**	(0.567)
+0.7	256	4	**1**	(0.618)	**2**	(0.695)	3	(0.804)	**0**	(0.378)
+0.7	256	8	**1**	(0.526)	**2**	(0.609)	3	(0.721)	**0**	(0.329)
+0.7	512	4	**1**	(0.521)	**2**	(0.571)	3	(0.647)	**0**	(0.252)
+0.7	512	8	**1**	(0.556)	**2**	(0.623)	3	(0.673)	**0**	(0.316)

of distributed nodes, which corresponds to the population size μ. Two baseline algorithms called PIWS and HEMO are used for the sake of comparison. In PIWS, we consider a set of uniformly distributed weight vectors. The weight vectors are fixed and do not change in the course of optimization. Each distributed node then execute multiple independent rounds with its assigned weight vector and uses a weighted sum fitness function to select candidate solutions. This allows us to appreciate the impact of our localized strategies on approximation quality and also the impact of distributed communications on running time. In HEMO, we use a $(\mu + \lambda)$ variant of SMS-EMOA [BNE07a], with a one-shot replacement strategy, which is a *sequential* and *global* hypervolume-based evolutionary multi-objective optimization algorithm. This allows us to appreciate how efficient our local strategies are compared with a global strategy having a full global knowledge of the search state, i.e., the whole current population.

As illustrated in Table 6 providing a bench of results, our weighted sum based variant DLBS_{OD} is better than the hypervolume based variant DLBS_{H}. This is actually to be interpreted in lights of our study on the impact of the scalarizing functions (Section 4.2.1); but from a more local and dynamic perspectives since the lines of equi-fitness values implied by the two localized fitness functions are not fixed but evolve dynamically. In Fig. 22, we show the trajectory of solutions in the objective space when using DLBS_{OD}, which give an illustration of the dynamics of the search process and the way the adaptive strategy could lead to well spread and improved solutions. When comparing DLBS to PIWS, we can see that DLBS performs substantially better independently of the localized fitness function that is considered. This is attributed to the local information exchanged in our cooperative and adaptive strategies; which is to contrast with PIWS where search directions are static. When comparing DLBS to HEMO, we found that DLBS_{OD} is better when having conflicting objectives ($\rho < 0$), where as HEMO is better for highly correlated objectives ($\rho > 0$). Based on a very local information, DLBS can find relatively well diversified solutions when the objectives are in conflicts; however, a more global algorithm like HEMOis able to find better solutions in a more focused area of the objective space which is intuitively a good option when the objectives are correlated. At this point, one should notice that the previous discussion holds when comparing DLBS to HEMO using the same fixed number of function evaluations; but with no considerations to execution time. With no surprise, it turns out that the running time of DLBS is impressively better than the running time of the inherently

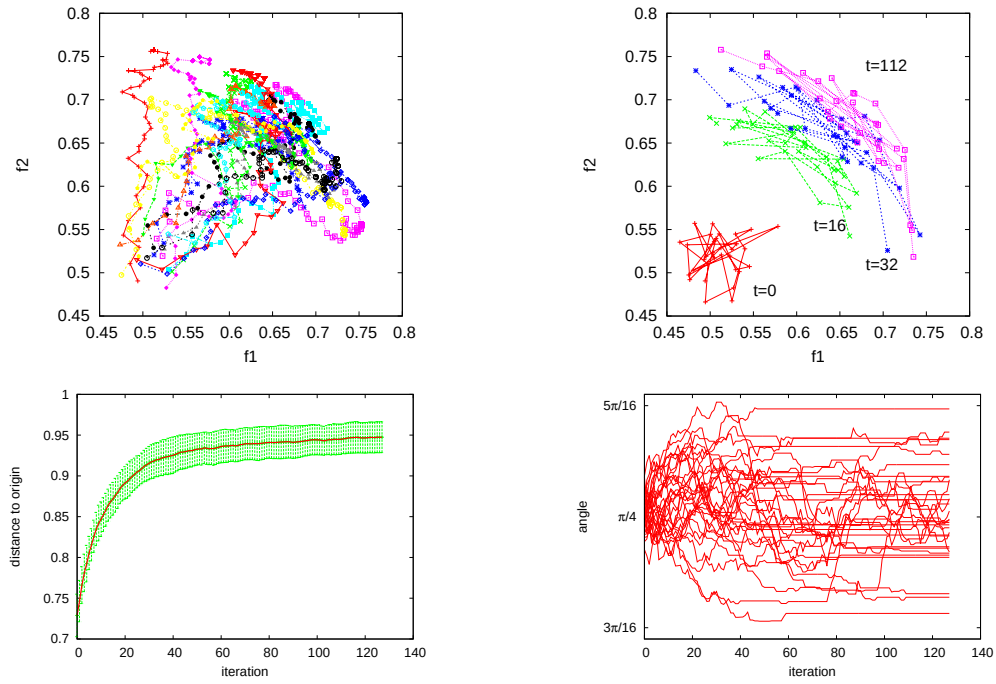


Figure 22: Dynamics of DLBS$_{OD}$ for a ρMNK landscape with $K = 4$, $\rho = 0.0$, $N = 128$ and $n = \mu = 128$ distributed nodes. Top-left: Evolution of the nodes trajectory (not all 128 trajectories are shown). Top-right: Evolution of the neighborhood graph. Bottom-left: Evolution of the average distance (and standard deviation) between node positions and the origin in the objective space. Bottom-right: Evolution of node angles.

sequential and global HEMO algorithm (especially because HEMOrequires global operations such as hypervolume computation, etc).

To conclude our analysis, we remark that the original motivation of the DLBS approach is to show how adapting the search directions distributively, and relying on a very local information, can lead to effective and scalable decomposition-based parallel algorithms. Our results allows us to support that such a methodology is promising and should be investigated more extensively. For instance, the DLBS approach could be combined with more sophisticated mating selection and replacement strategies as in sequential MOEA/D. Sharing/Migrating solutions among distributed nodes is also another limitation of the DLBS approach. These different possibilities raise other challenges specific to the parallelization of the computational flow of other standard decomposition-based algorithms such as MOEA/D. The next sections describe our contributions in this respect.

In Fig. 23, we illustrate the parallel performance of DLBS using a message passing distributed implementation. Notice that since only four real values are exchanged between neighboring nodes, the size of messages used to effectively implement DLBS does not depend on the solution encoding or problem size; which is technically speaking an interesting feature that allows us to scale our approach very efficiently. In particular, we are able to show substantial speedups depending on problem size, which is tightly related to the cost of the fitness function evaluation. The parallel efficiency, that is the computation to communication cost, is already around 90% for instances of size $N = 512$, and reaches more than 95% for large instances of size $N = 2048$. We also obtain linear acceleration factors when comparing our message passing implementation of DLBS to a sequentially simulated implementation of the same algorithm using a singe compute entities. This shows that from a purely parallel perspective, our DLBS approach is able to scale efficiency both as a function of problem size and number of distributed resources.

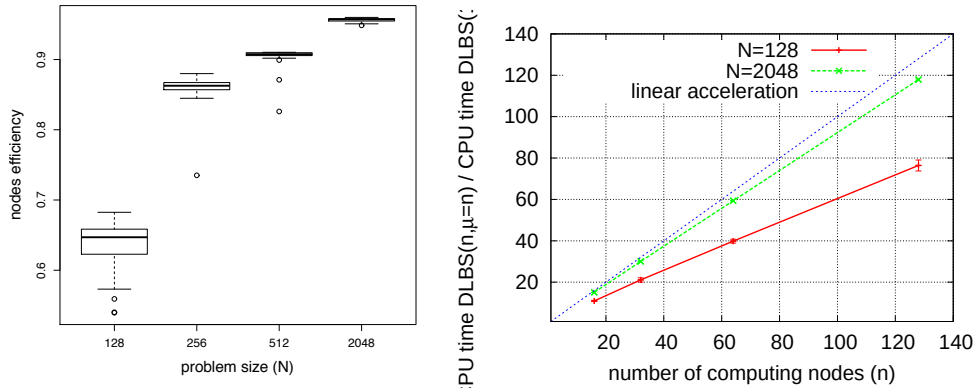


Figure 23: Parallel efficiency and scalability of DLBS. Left: Average ratio of computing time over execution time; showing the impact of the fitness function evaluation time as a function of N ($\mu = n = 128$). Right: Average acceleration ratio of DLBS with respect to the population size μ and $n = \mu$ distributed nodes.

4.3.2 Designing Parallel Multiobjective Decomposition

In this section, we are concerned with the design of parallel decomposition-based multi-objective algorithms. It is worth-noticing that one can find an extensive literature on designing parallel/distributed multi-objective solving methods [VZL03; CLV07; Tal+08; BAE09]. Two interdependent issues are usually addressed: (i) how to decrease the computational complexity of a specific multi-objective algorithms and (ii) how to make parallel processes cooperate to improve the quality of the Pareto set approximation, see e.g., [ZL02; DZJ03; CS04; MTC06; TYG06; MBS07; HYM07; Dur+08; Fig+10; Mos10; Dor+13]. Reviewing all the literature is out-of-the-scope of this document, However, let us comment that parallel and cooperative techniques implicitly come with the idea of *decomposing* the search into a number of sub-problems so that a diversified set of solutions, in terms of Pareto front quality, can be obtained. The main challenge is on defining efficient strategies to either divide the search space or the objective space. For instance, the population induced by a particle swarm multi-objective algorithm is divided [MBS07] into sub-swarms which are then coordinated through a master-slave approach. The diffusion model [VZL03] and the island model [Tom05; ZL02; Dor+11] have also been extensively adopted to design distributed cooperative methods. In the so-called cone separation techniques [Bra+04; SUZ05; BAE09], the objective space is divided into regions distributed over some islands.

Surprisingly, investigating to what extent a well established algorithm such as MOEA/D, where decomposition of the objective space is explicit, can be redesigned to run efficiently in large-scale parallel compute environments is not yet fully addressed; although one can find few attempts in this direction [ND10; Dur+11; MI14]. We attribute this to the following two questions : (i) How to maintain the search ability of the MOEA/D framework when attempting to break the dependencies in the computational flow of its original implementation and (ii) How to deal with the fine-grained parallelism that is likely to be encountered when effectively deploying the so-obtained variants at a large distributed scale? The challenge standing behind the previous questions is to derive novel parallel algorithms presenting a good balance between approximation quality and speed-up in the largest scales where fine-grained workload can drastically prevent high performance, especially in the scenario where communication cost is non-negligible compared to the objectives' evaluation cost. The work described in the following is mainly an attempt to address this challenge while gaining more insights into the best practices one has to follow when adopting a parallel decomposition-based approach.

DESIGN BOTTLENECK AND OVERVIEW OF EXISTING PARALLEL MOEA/D VARIANTS. The mating selection and the replacement in MOEA/D are performed using the information coming from the T-neighborhood of each subproblem. Hence, this creates a dependency between subproblems when attempting to evolve their respective solutions in parallel. One the other side solving the so-defined subproblems independently would offer obvious parallelism; but is not accurate of optimal performance. This issue was first studied in [ND10; Dur+11] where the authors investigated the intuitive idea that non-overlapping sub-problems, i.e., sub-problems having disjoint T-neighbors, could be processed in parallel. It is in particular shown in [ND10] that interesting speed-ups can be obtained at the price of significantly deteriorating the approximation quality compared to sequential MOEA/D. In [MI14], a parallel variant of MOEA/D based on the island model is investigated. In such a model, every island evolves a sub-population of individuals with respect to some sub-problems. Selected individuals are then sent to other islands during a migration phase. The parallel efficiency of the such a model is demonstrated with a 8-core shared-memory machine and a specific thread-based implementation. However, it is well understood that scaling up such a thread based approach while maintaining its accuracy can suffer several shortcomings; mainly because concurrent shared-memory read/write operations are no more possible, and distributed communication is typically many orders of magnitude more costly, which can then be prohibitive for quality and/or parallel efficiency.

A FINE-GRAINED (MESSAGE PASSING) PARALLEL MOEA/D. Understating how the quality of MOEA/D is affected by parallelism, and what speed-ups can be attained when facing fine-grained parallelism at large distributed scales, is not yet fully accomplished. Our work departs from the previously mentioned studies in several aspects; but it also retains insightful lessons learnt from them. As in [ND10; Dur+11], the idea of handling overlapping neighbors is a key point for scalability and high-quality approximation. As in [MI14], we get also inspired by the island model, but we do not explicitly rely on the conventional concept of islands (migration, etc) which allows us finely optimize our approach when facing fine-grained parallelism. Actually, we basically rely on our contribution on deriving a generational variant of MOEA/D (Section 4.2.2) in order to incorporate parallelism while maintaining a good approximation set.

Our parallel MOEA/D scheme, called MP-MOEA/D, is summarized in Algorithm 12 to be executed independently in parallel by *every* processing unit (PU), i.e., all variables are local and not shared in any way. A one-to-one mapping between subproblems and PUs is considered while assuming seemingly the same T-neighborhood relation between subproblems. Notice that this is a harsh assumption which allows us to fairly study the scalability of our scheme with very fine-grained computations. The computations performed by every PU are then divided in two stages. The first stage is performed locally without any communication, whereas the second stage requires communication with neighbors. The goal of every PU is two-fold: (i) to identify an improving solution for its own sub-problem, and (ii) to check whether an improving solution is found w.r.t. neighboring sub-problems. For this purpose, every PU maintains *locally* a representative *copy* of the solution of each neighbor. A PU then performs the same selection and variation mechanisms as in the conventional MOEA/D, with essentially three main modifications: (i) the PU current solution is always selected for reproduction as in MOEAD-**SC** (Section 4.2.2), (ii) the number of iterations during which new offspring solutions are generated is controlled by a parameter t_{max}, and (iii) since the current remote solutions of neighbors are not available locally, a PU simply checks whether any newly generated offspring does improve any of the local copies maintained for every neighbor. The idea here is that, if the local copies are sufficiently up-to-date, then the protocol has the ability to concurrently generate a 'good' offspring and to correctly detect any improvement on behalf of neighboring PUs. The improving offspring solutions are momentarily saved in order to be shared in the communication stage which allows to update and synchronize the local copies.

Algorithm 12: MP-MOEA/D: High level code for *every* distributed node i

Input: $B(i)$: neighboring sub-problems; λ^j for every $j \in B(i)$: neighbors' weight vectors;
// Neighbors' local copies initialization
INITIALIZE$\left(\cup_{j\in B(i)} x^j, z^\star\right)$; flag $\leftarrow 0$;
while STOPPING CONDITION **do**
  // Stage #1: Local computations
  for $j \in B(i) \setminus \{i\}$ **do** $y^j \leftarrow x^j$; ;
  Repeat t_{max} **times:**
    // Mating selection and variation
    $\ell \leftarrow rand(B(i) \setminus \{i\})$;
    $y \leftarrow$ CROSSOVER_MUTATION_REPAIR(x^i, x^ℓ);
    // Local Replacement
    if $g(y, \lambda^i) < g(x^i, \lambda^i)$ **then**
      $x^i \leftarrow y$; flag $\leftarrow 1$;
    // Check for neighbors' improvements
    for $j \in B(i) \setminus \{i\}$ **do**
      if $T(y, \lambda^j) < T(y^j, \lambda^j)$ **then** $y^j \leftarrow y$; ;
  // Stage #2: Distributed update
  DISTRIBUTED_LOCAL_COPIES_UPDATE (flag);

After the local computation stage, the second stage involving the update of local copies is activated. This stage is quite technical and is not detailed in Algorithm 12. It is designed to handle the following two main situations that can occur in the first stage: (i) an improving solution with respect to the PU's own sub-problem has been identified, (ii) improving solution(s) for one (or more) neighboring sub-problem(s) have been identified. In the first case, the PU has to notify its neighbors so that they can update their local copies with its new current solution. In the second case, a PU has to notify the corresponding neighbors so that they can update their own solutions with a new improving offspring. Symmetrically, a PU checks whether these situations occur at one (or more) neighbor(s) before resuming a new stage of local computations.

The technical implementation of the above described update mechanism is to be handled carefully since this is where fine-grained parallelism can prevent scalability. The fastest the states of PUs are updated with fresh information from neighbors, the better should be the approximation quality. On the other side, synchronizing PUs distributively implies a non-negligible communication cost that might even dominate the cost of local computations. This is precisely why we have introduced the parameter t_{max}, which offers the possibility of controlling the relative cost of local computations by setting the communication frequency. Moreover, we consider both a synchronous and asynchronous message passing implementations where all (local copies) distributed update operations are aggregated into a single message in order to reduce the number of messages transmitted over the network. In the *synchronous* case, every PU sends a message with accurate information to its neighbors, and then blocks waiting for their respective messages to be received. This synchronous implementation provides the guarantee that all PUs will be updated with fresh information; however, additional acknowledgements have to be issued in order to avoid deadlock situations, which has the drawback of introducing idle times. An *asynchronous* implementation allows us to reduce idle times by removing the remote synchronization costs; however, it does not guarantee that the local copies of PUs are up-to-date. Hence, it can lead to the situation where the evolutionary optimization is eventually resumed for several rounds with outdated information, which can constitute a penalty in terms of approximation quality.

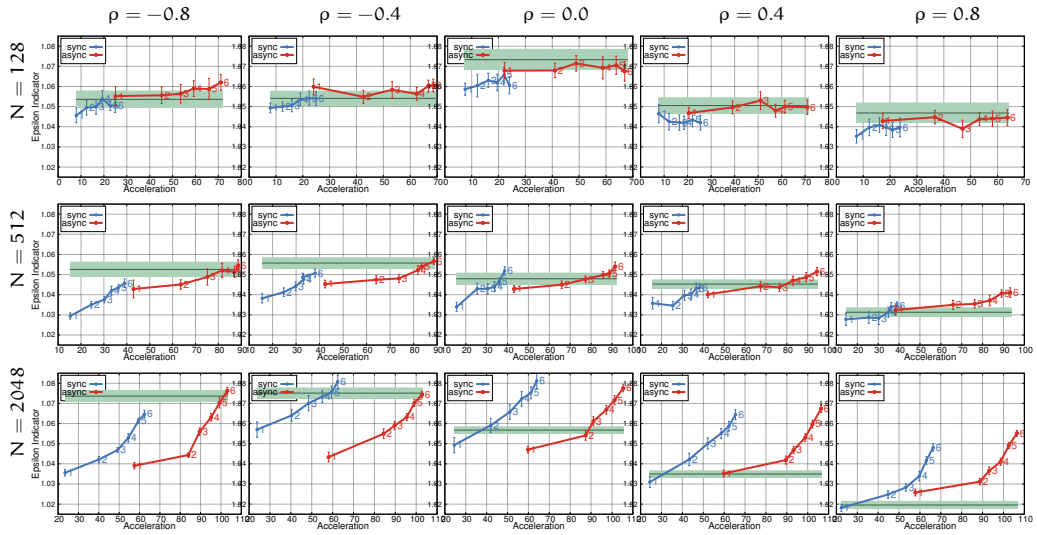


Figure 24: Acceleration *vs.* Quality (Epsilon indicator) as a function of t_{max} for different ρMNK landscapes with $K = 4$ and $M = 2$.

SELECTED EXPERIMENTAL RESULTS. Our experimental investigations using a message passing implementation reveal that our MP-MOEA/D can actually achieve a non-trivial trade-off in terms of approximation quality and parallel efficiency. This is illustrated in Fig. 24 for different ρMNK landscapes where we consider approximation quality (indicator's value) and acceleration as the *couple of goals* one would like to achieve *simultaneously*. Here the acceleration is measured as the ratio between the running time of sequential MOEA/D and the parallel running time of MP-MOEA/D using 128 compute cores. Each point represents the acceleration (x-axis) and the approximation quality (y-axis) corresponding to one value of the update frequency parameter t_{max} (reported as a label in the figure), and where the horizontal line with thick contour represents the average approximation quality obtained by the sequential MOEA/D (with a 95% confidence interval). Since the quality indicators are to be minimized, the points being below this line indicate that MP-MOEA/D is competitive compared to MOEA/D. Similarly, since acceleration is to be maximized (with an ideal value at 128), points being farther on the right-side of each subfigure indicate better parallel efficiency.

We can observe that approximation quality rather drops with higher t_{max}-values, whereas acceleration becomes better. The interesting observation is that there exists values for which quality is very competitive to sequential MOEA/D with substantial improvements in acceleration. In particular, when considering small-size instances, for which relatively high speed-ups are more difficult to achieve since the evaluation function cost becomes relatively low compared to communication cost, we observe that the impact of t_{max} on quality is less pronounced. Hence, larger t_{max}-values have the overall effect of significantly improving acceleration without a substantial drop in terms of speed-up. Notice for example that speed-ups of up to 70 can be obtained with a quality being similar to MOEA/D for $N = 128$ whereas a speed-up of only 10 to 20 is obtained with $t_{max} = 1$. Secondly, the obtained trade-offs depend on the correlation between the objective functions, especially for large-size instances. In fact, the more correlated the objective functions, the harder it is to obtain higher acceleration without a significant drop in performance. We attribute this to the fact that for such an objective correlation, an improving solution found at some PU with respect to a given sub-problem, is more likely to dominate neighboring solutions, and thus to subsequently produce new improving offsprings that speed-up the convergence of the optimization of neighboring sub-problems. This is actually related to the replacement dynamics

in MOEA/D and to the wave effect discussed previously at the end of Section 4.2.2. Hence, communicating improving offspring solutions immediately, as they are discovered at every PU, is more critical for approximation quality. This is less likely to happen when dealing with anti-correlated objective functions for which the size of the Pareto front is larger, and where diversity can balance this 'side'-effect. Notice also that the more conflicting the objectives, and the larger the instances, the more an asynchronous implementation dominates the synchronous one, both in approximation quality and acceleration. For example, for such configurations, asynchronous MP-MOEA/D is able to attain a near linear acceleration while being as good as MOEA/D in terms of approximation quality.

4.4 OTHER RELATED CONTRIBUTIONS

In our research, we were also interested in other challenges that do not relate solely to decomposition techniques, in the sense that they are common to other search paradigms and concern other important aspects of multi-objective optimization in general [Lie+17a; Mon+17; Shi+17b; Sag+17; Cua+17; Der+16; Der+16; Sag+16; Bas+16; Mar+15a; Mar+15b; Dro+14; TBD13]. Firstly, whether we are given a discrete or a continuous multi-objective problem is an important aspect one has to take into account. In particular, the (evolutionary) operators one has to consider and the way they are effectively incorporated into a general-purpose multi-objective search process is of crucial importance. Secondly, a multi-objective search process has very specific dynamics that are difficult to elicit especially because of the curse of dimensionality. In the following, we provide a brief overview on some of our recent work on these aspects. In Section 4.4.1, we highlight our contributions on incorporating local search procedures in multi-objective decomposition for combinatorial domains [Der+16; Der+16; Shi+17b]. In Section 4.4.2, we highlight our contributions on designing adaptive evolutionary operators for multi-objective optimization [Mar+15b; Sag+16; Sag+17].

4.4.1 Connecting Decomposition and Local Search

GENERAL MOTIVATION. Local search (Ls) is at the corner-stone of different advanced (single-objective) meta-heuristics and evolutionary algorithms that have been extensively studied and tuned by the optimization community especially when dealing with *combinatorial* optimization problems. The algorithmic components, the parameters and the variants of the MOEA/D framework are however very often investigated for *continuous* problems. Although some adaptations exist for the combinatorial setting [Cha+08; KZB13; KZB14], we can safely claim that no systematic and comprehensive studies considering the incorporation of Ls ingredients within MOEA/D can be found. In this respect, we argue that room for new research investigations exists in order to design novel multi-objective search algorithms based on both local search and decomposition for the efficient solving of difficult combinatorial optimization problems. Generally speaking, we are interested not only in bringing standard single-objective local search techniques into the framework of MOEA/D, but also in investigating how decomposition techniques can be used to improve local search based multi-objective algorithms, such as the well-established Pareto-Local Search (PLS) algorithm, belonging to the family of dominance-based algorithms (See section 4.1.2).

INCORPORATING LS INTO MOEA/D. The class of (single-objective) local search heuristics [HS04] encompasses several algorithms having different components and different degrees of complexity. A common ingredient being at the basis of a successful algorithm in this class is the neighborhood exploration and the move strategy. In fact, the two basic components of Ls are: (i) the definition of at least one neighborhood relation/structure, providing for every single solution a set of neighboring solutions that can be derived by performing little changes or perturbations, and (ii) the setting of the move strategy, that is how to explore those neigh-

boring solutions and how to guide the search process when iteratively moving from one solution to another. Standard and typical move strategies in a single-objective setting are as follows. In a *best-improvement* (or steepest descent) move strategy, the neighbor that improves the most the evaluation function is selected at each iteration. In a *first-improvement* move strategy, the first improving neighbor which is found is immediately selected. This strategy avoids to systematically generate and evaluate the whole neighborhood. Moreover, the neighborhood structure can be used as a an evolutionary mutation operator when some few neighboring solutions are sampled at random, and then the improving ones (if any) are considered for a possible move. Hence, a *random* strategy can be considered as well, where a random neighbor is generated at each iteration, and replaces the current solution iff there is an improvement.

Given that the MOEA/D framework transforms the original problem into multiple single objective ones, integrating such move strategies is actually a natural outcome. However, important design technicalities have to be explicitly and carefully specified for optimal performance. In particular, one has to manage the crucial exploration/exploitation aspects when creating new solutions and replacing old ones from and into the population in order to guarantee a reasonable balance between the diversity of the population and the convergence toward the Pareto front. In [Der+16], we study how the replacement flow of MOEA/D can be adapted and hybridized to support simple local search move strategies. Despite their simplicity, the obtained algorithm variants are shown to have very different search dynamics and behavior. Our study is actually conducted by designing a new set of bi-objective traveling salesman problem (TSP) instances[5] with tunable objective correlations, hence leveraging existing standard ones, e.g., [LT10; PS09]. We are for instance able to show that the ranks of the different move strategies (in terms of approximation quality) depends strongly on the objective correlation factor. We also study the anytime performance (that is the approximation quality using different amount of computing budgets) of the different local search variants and consider their behavior with respect to other MOEA/D specific parameters, such as the population size, the maximum number of replaced solutions, and the probability of parent selection. Our results revealed strong evidence on the need of adaptive (online) mechanism to select and combine different move strategies on line in the MOEA/D framework. This indicates that incorporating Ls into MOEA/D is still in its very infancy beginning, and hence, would deserve much more attention and research investigations.

BRINGING DECOMPOSITION INTO PLS ... IN PARALLEL. Local search is not restricted to the solving of (multiple) single-objective (sub-)problems and can be applied as a whole concept to multi-objective optimization problems as well. As illustrated previously in the template of Algorithm 7, Pareto-Local Search (PLS) can be viewed as a local search operating at a set level and stopping naturally after reaching a Pareto local optimum set [PSS07]. Although the basic PLS illustrated in Algorithm 7 enables to obtain high quality approximation sets, it is well known that its convergence speed is low and several strategies have been proposed [DT12; DLS15; Gei11; Lie+12] in order to overcome this issue. Actually, PLS has three main problem-independent components that were shown to be crucially important for its anytime performance [DLS15; Lie+12]: (i) the selection step that is what next solution to choose to explore from the archive, (ii) the neighborhood exploration which related directly to the move strategies discussed in the previous section, and (iii) the acceptance criterion that is how to update the archive which is tightly related to the replacement mechanism. We can also remark that with the exception of the exploration step, the two other strategies need to have full knowledge of the archive, which makes it rather challenging to derive a high level parallel version of PLS. In [Shi+17b], we study how we can get inspiration from decomposition-based techniques in order to improve the different PLS components, and in the same time to be able to parallelize it.

5 The instances and their description are made available online at the MoCObench repository dedicated to multiobjective combinatorial optimization problem instances.

We consider to decompose the objective space evenly into several small regions based on some weight vectors as in [LGZ14]. In the bi-objective case, this consists in delimiting a small region of the objective space using a reference point and two consecutive (neighboring) lines passing through the reference point as in standard MOEA/D. Then, we consider to run *in parallel* several cooperating PLS processes, each one operating in one of the so-defined regions. When a PLS process finds a solution out-side the boundaries of its region defined by decomposition, it simply ignore it unless no solutions within the boundaries exist in its local archive. The selection and replacement steps of basic PLS are also redesigned accordingly. The components of every parallel PLS process are updated with respect to the weight vector corresponding to the region where it is expected to operate. A weighted sum is hence used as a scalar function allowing every PLS process to have a total order on the solutions it can generate by local search. Instead of selecting or replacing a solution from the archive based on dominance, every parallel PLS process uses the scalar function (parametrized by the corresponding weight vector) to rank solutions and hence to differentiate between them. This allows us not only to coordinate the parallel PLS processes locally by simply using different weight vectors, but also to reduce drastically the size of the archive maintained globally. Our experimental investigations on standard instances of the well-established multiobjective Unconstrained Binary Quadratic Programming (mUBQP) problem, for which conventional PLS is known to provide high quality results, show that bringing decomposition into PLS, is beneficial both at the selection and replacement levels to improve convergence, while enabling a very efficient high level parallel design that was un-explored until now. Of course, this opens the door to further investigations both to improve the anytime behavior of PLS and also to deploy PLS on large scale distributed environments and for large scale instances.

4.4.2 Design of Adaptive Evolutionary Operators

In MOEA/D, an evolutionary operator applied at a subproblem is not only acting selfishly to improve the solution of its current sub-problem by "stealing information" from others, but it also behaves in an altruistic way by helping to improve the solutions of neighbors. In some sense, this allows to impact the distribution of solutions that can be obtained in every iteration, and hopefully helps in searching promising regions. From a very general perspective, the most important aspect for a successful evolutionary multi-objective algorithm is perhaps its ability to generate new promising solutions and hence to push the population effectively towards the Pareto front. While the internal algorithm-specific mechanisms, such as the population structure, the mating selection, the replacement, etc, are undoubtedly extremely important, the design of powerful and accurate evolutionary operators is another challenging issue. In our work, we are interested in this aspect as well [Mar+15b; Sag+16; Sag+17; Mar+15a; Dro+14]; and in particular, in techniques and algorithms that are able to capture the dependencies between the variables and to exploit the information obtained so far in order to adapt and to control the generation of new candidate solutions online during the search. This is highlighted in the next paragraphs for continuous domains.

INJECTING CMA-ES IN MOEA/D. In [Mar+15b], we investigate new opportunities offered by the flexibility of MOEA/D in incorporating the well-established CMA-ES (Covariance Matrix Adaption Evolution Strategy) [HO01]. Our interest in combining the CMA-ES with MOEA/D stems from two sources. On the one hand, CMA-ES has been shown to be among the best performing single-objective blackbox algorithms, with typically superior performance to other popular evolutionary operators such as Differential Evolution and other numerical optimizers—especially when the problems are difficult and the budgets are not too small [Han+10]. On the other hand, recent investigations [Han11] showed that external solutions can be easily *injected* into the algorithm to gain information from good solutions that are not sampled directly by the algorithm itself. Both aspects together make the CMA-ES with solution injection a highly interesting candidate to be used within a multi-objective

decomposition based framework like MOEA/D, where the population is precisely structured to enable single-objective subproblems to directly share interesting solutions. We therefore propose a novel variant of MOEA/D where CMA-ES is used as the core single-objective evolution engine and where the injection idea allows us to incorporate information from neighboring scalarizing problems into the search distributions. Notice that several multi-objective versions of the CMA-ES algorithm exist [IHR07; ISH07] which, however, do not resemble the framework of MOEA/D but instead aim at maximizing the hypervolume of a solution set in the framework of indicator-based algorithms. Besides being able to obtain competitive results, in particular compared to multi-objective CMA-ES, our investigations highlight novel promising alternatives by either leveraging the existing single-objective CMA-ES related variants (e.g., restart conditions, population size control, etc) and/or by exploring decomposition specific features (e.g. parallelization is enabled in a quite straightforward manner, the choice of appropriate scalarizing functions with respect to single-objective CMA-ES, etc). Our goal is in fact beyond beating existing variants of MOEA/D or multi-objective CMA-ES on some benchmarks; but more importantly, we aim at gaining a more fundamental understanding of what makes a multi-objective evolutionary process effective and under what conditions in line with other recent studies [CML17; Cas+17].

LEARNING VARIABLE IMPORTANCE TO GUIDE RECOMBINATION. In the same research line, we consider the idea of learning, in an online fashion based on statistical modeling, which variables affect convergence to the Pareto front. The rational behind this idea is that different subsets of variables may influence convergence towards different objective subspaces, while others may influence diversity. This can be the case for instance when dealing with many objective problems, where convergence is crucially important for an optimization process to approach the high dimensional Pareto Front. Although it is not clear that a sharp separation of variables always exists (a complex and *a priori* unknown interaction of several variables usually affect both convergence and diversity), this kind of approach can allow to extract a valuable knowledge of the problem being tackled, and in particular, help designing improved and finely tuned evolutionary operators. In [Sag+16; Sag+17], we propose to use the Pareto ranking induced by non-dominated sorting [Deb+00] as the score to render how good solutions are with respect to convergence. We then bias standard variation operators accordingly in order to help finding Pareto optimal solutions as soon as possible, and hence to improve the algorithm convergence. We use random forest [Bre01], a machine learning algorithm, in order to perform a regression of the Pareto rankings, in terms of non-dominated sorting, over decision variables at each iteration. From fitting the statistical regression model, we obtain estimates of the variable importance, which we later use to select the variables that will undergo variation. Besides showing that such an approach is able to achieve a significantly better convergence on some well-established continuous benchmark instances, our investigations suggest that the design of machine learning-enhanced evolutionary operators is one promising research direction that can definitely help catching the complexity of multi-objective optimization problems.

We conclude this section by emphasizing the importance of designing accurate evolutionary operators (both for continuous and discrete domains). For multi-objective optimization problems, where besides the size of the Pareto front and its dimensionality which are in fact a big issue, the design of a relevant evolutionary operator that can accommodate simultaneously to the properties of the considered objectives and to the current state of the population is extremely important. In our opinion, incorporating the lesson learnt from single objective optimization can only be a first step and much research remain to do in order to come out with evolutionary operators designed specifically to deal with the multi-objective nature of given problem.

4.5 CONCLUSIONS AND PERSPECTIVES

SUMMARY. In this chapter, we provided an overview of our research contributions on evolutionary multi-objective optimization, with a particular focus on decomposition based approaches. Our work can be viewed along different tightly related challenges. First, we addressed the analysis of the impact of using some particular scalarizing function and shed more light on the corresponding search behavior. By proposing novel mating selection and replacement mechanisms, and more importantly by conducting extensive empirical investigations to compare and understand the implied search dynamics and performance relatively to other existing techniques, we are able to provide a more fundamental understanding of the trade-off in terms of diversity and convergence that a decomposition based framework such MOEA/D allows to obtain. The other critical component for a successful evolutionary algorithm, is the design of efficient variation operators. For this purpose, we investigated the hybridization of decomposition with other search paradigms that were mostly considered in the single objective setting, in an attempt to design more effective search procedures. For discrete problems which constitute our central focus, we consider to connect local search with decomposition in order to design novel and systematic high level approaches. In fact, we argue that the community lacks much knowledge on the benefits of tackling a combinatorial multi-objective problem with decomposition, which can essentially be attributed to the lack of insights into the accurate choice of the standard tools from combinatorial optimization and their effective integration into decomposition. For continuous problems, we consider adaptive and machine learning based techniques in order to guide the search efficiently. For instance, our first investigations on leveraging the state-of-the-art CMA-ES single objective optimizer and the underlying search paradigms, i.e., stochastic model based sampling and cooperative adaption, are in our opinion a very interesting research path to follow in order to design novel multi-objective specific search operators. Additionally, one important aspect of our work consists in thinking decomposition in a very local manner, in the sense, that we paid much attention in designing local cooperative rules in order to enhance solution quality, and also to enable parallelism to take full benefits from large scale distributed and parallel resources. In our opinion, the high level parallelism exposed by the divide-and-conquer paradigm underlying decomposition is in fact a strong feature that will allow to design more powerful and more effective search procedures and to tackle increasingly complex and costly optimization problems.

UNDERSTANDING MULTI-OBJECTIVE SEARCH. Decomposition is to be viewed as a tool and not as a goal. In this respect, we are also interested in other fundamental research issues which relates to the understanding and the eliciting of multi-objective search behavior and dynamics. In fact, despite the number of available algorithms, their skillful design and their flexibility when applied to a large spectrum of problems, a key ingredient to make them efficient and effective lies in the choice of their components in order to be specifically adapted to the multi-objective optimization problem being tackled. This might even depend on the intrinsic properties of the problem instance being considered. For instance, why solving a sub-problem cooperatively is fundamentally more effective than solving it independently? Is it because cooperation implies better search behavior or is it because some hidden aspects with respect to the problem at hand makes it more simple? Or is it because of the two aspects? Why different operators are expected to provide different performance and for which problem instances? What makes it difficult to tackle optimization problems with many objectives? Is it solely because the intractability and curse of dimension? Many other questions can be raised with respect to multi-objective optimization and evolutionary algorithms in general. In this respect, there is evidence that a principled approach using new tools and techniques dedicated to the understanding of the behavior of existing algorithms in light of the properties of the multi-objective problem under study, are needed. Such tools and techniques exist in the single-objective optimization literature, where a number

of paradigms, for instance from the fitness landscape analysis, have proved to be extremely helpful in attaining such a goal. Motivated by their success and their accuracy, there was recently several studies leveraging the single-objective case and pushing fitness landscape analysis a step toward the development of new statistical methodologies and the identification of general-purpose characteristics and features that fit the multi-objective nature of a given optimization problem; including some of our recent contributions that were not discussed in this document [Lie+17a; Mon+17]. However, still a relatively huge gap remains between the design of multi-objective randomized search heuristics and the fundamental understanding of their effectiveness with respect to the properties of the tackled problem. This represent a difficult challenge that is timely to address in order to avoid being trapped in ad-hoc or hyper-specialized optimization methodologies.

UNDERSTANDING AND DESIGNING DECOMPOSITION. The previous research perspective can be considered more thoroughly in the context of decomposition based approaches. The overall research objective is to design an appropriate algorithmic framework adopting the evolutionary decomposition paradigm and its algorithmic Gestalt. Firstly, a key point is to identify the different possible decomposition strategies, and to systematically study their behavior as a function of the target problem properties in order to strengthen them with new and accurate algorithmic components. Secondly, we should keep in mind that it can be unwise and uneconomic to always ignore the knowledge and experience gained in the past. In this respect, decomposition makes it more flexible to incorporate existing solving technique although an in-depth re-design could be necessary to fit the multi-objective nature of the tackled problems. For instance, and as argued before, decomposition is clearly one key technique to enable scalability, given that the original multi-objective optimization problem is broken into smaller single objective sub-problems. However, decomposition can occur not only in the objective space, but also in the variable space which we will have to address in the future in order to fully cope with the possibly complex and large scale nature of optimization problems. Thirdly, we need to better understand the relation between the structural properties of an optimization problems and the parameters/components used in a decomposition based algorithm in order to consolidate the scientific foundations of such an approach. In fact, we argue that we need to develop a comprehensive methodology informing about what makes a decomposition effective and efficient (and why), and what are the intrinsic features that makes it effective or not for a multi-objective problem. As a byproduct, one concrete and possible solving approach is then to adapt state-of-the-art off-line and on-line automatic algorithm design and portfolio based techniques (See Chapter 3) viewed as high level optimization tools to efficiently search the space of possible configurations (e.g., in terms of the number of weight vectors, the type of aggregation function to choose, the constraints to define sub-problems, the parameters to use for every sub-problem, etc). Some research in this line already exist but it is still in its very infancy beginnings. We in fact need innovative generic tools, especially taking inspiration from existing high level autonomous and machine learning inspired search paradigms, that can be applied specifically to multi-objective optimization, thus eventually ending with autonomous cross-domain decomposition solvers with minimum burden for non-expert end-users.

MAKING IT PARALLEL AND ... LOCAL. Taking advantage from the decentralized nature of decomposition, combined with the compute power of modern massively parallel platforms, is the other obvious perspective of our work. In particular, the possible sources of parallelism exposed naturally by decomposition are good candidates for effective problem solving on massively parallel compute platforms. Parallelism is then to be viewed as a key element that will open the doors towards new algorithmic concepts that would not rise otherwise. An interesting research perspective is to address the heterogenous and hierarchical nature of modern devices and platforms in order to achieve scalability and high performance within decomposition. Different parallelization paradigms and models, different programming

languages and libraries, different implementation strategies will be required. Our message passing and fine-grained parallel MOEA/D provides some hints on some critical issues to be address. When considering a more generalized decomposition approach for solving possibly large scale problems, the challenge of achieving high performance and scalability without decreasing search performance is even more difficult to address. In this respect, future work will have to focus on identifying the different sources of parallelism induced by decomposition (both in objective and decision space) independently of their effective parallelization. For instance, scalability in general implies imbalanced computations. In the case of decomposition, different sub-problems can be expected to require variable computational efforts to be solved; and high level parallelism can provide alternative solving approaches. Some work on estimating the relative difficulty of each sub-problem exist in the sequential setting and using simple scalarizing functions; which is based on collecting information about the estimated progress rate on-line during the search process. However, to our best, no distributed approach considering adaptive and on-line cooperative learning between different parallel processes has been investigated so-far. Our work on distributed adaptive algorithm selection and machine learning inspired technique can be coupled with decomposition in order to locally adapt not only the effort to solve some subproblems but also the algorithmic components or parameters that are used in the cooperative and distributed solving process.

5 LOOKING AT THE FUTURE

In this document, we presented an overview of our research work at the crossroad of distributed and high performance computing, general-purpose search optimization algorithms in general. The last few years were in fact relatively rich in collaborations and contributions which allowed us to address a number of research topics related to both exact and heuristic search, sequential and distributed algorithms, evolutionary algorithms and related machine learning inspired approaches, single- and multi- objective optimization. The number of perspectives and future work that were discussed previously in this document with respect to each chapter, tell much about the complexity of optimization problems and the need to cross-fertilize the knowledge from a relatively wide range of fields in computational science before ending with a global, effective, and unified optimization methodology. In this respect, my research objectives are mostly motivated by the proposal of principled approaches for the design, analysis and understanding of optimization problems and algorithms.

It is our opinion in fact that in order to face the ever-increasing complexity of nowadays optimization problems and related applications, what is needed is not 'yet another' algorithm, but a determined and significant attempt to reformulate our fundamental understanding of the difficulty of solving optimization problems, right across the wide spectrum of optimization techniques and algorithms. There is evidence that standard optimization approaches need to be renovated by taking inspiration from other tightly-related fields in computational intelligence, statistics, machine learning, and distributed computing, in order to come out with new cutting-edge innovative and effective techniques and algorithms. Our ambition in the future is to build on the success of general purpose search heuristics, the maturity of meta-modeling and machine learning techniques, the advent of new analytics-driven methodologies in computational intelligence, and the availability of modern parallel computing facilities, in order to design, analyze, and evaluate novel algorithms and techniques in the context of large scale, heterogenous and cross-domain optimization problems. This is discussed in more details in the following paragraphs, which are tightly related to the current research activities that we are carrying out both within our newly created research team, our newly created France/Japan International associated Lab on massive optimization and computational intelligence (LIA-MODO), our ANR France Hong Kong bilateral project on big multi-objective optimization (bigMO), and in collaboration with several colleagues, without whom this piece of research would not be possible (see e.g., Appendix A.3).

MASSIVE AND BIG OPTIMIZATION PROBLEMS AND ALGORITHMS. Following the evolution of modern computational science, the field of optimization is inevitably shifting rapidly to the 'big' era where the large-scale nature of applications implies optimization problems, models and algorithms, increasingly large-scale and heterogeneous, coming from various applications and domains, with a large number of decision variables and conflicting objective functions of different nature, and with multiple sources of uncertainty. For instance, many optimization problems within the context of sustainable systems, multidisciplinary engineering design and innovation are increasingly complex, and involve large-size instances, cross-domain formulations and heterogeneous objectives, as well as multiple sources of uncertainty, for instance due to heavy simulations or missing data. Such characteristics lead to *massive* optimization problems, and raise new important and difficult scientific challenges for researchers and practitioners, that traditional approaches will hardly succeed when facing them. What is needed is to push the boundaries of existing optimization approaches, to go beyond the small- or medium- scale problems investigated so far in the literature, and to

design innovative flexible general-purpose and computationally intelligent algorithms able to efficiently and effectively tackle such massive optimization problems. Although some research dealing with the aforementioned characteristics can be found, the global difficult challenge is to propose a *unified* approach. In particular, we are interested in developing and setting up the foundations of cutting-edge autonomous solvers able to globally and jointly address the challenges encountered in problems from massive optimization following the after-mentioned aspects. *Large-scale optimization problems*, which commonly involve hundreds of variables that induce a large increase in the solution space where the optimization algorithm operates. *Any-objective optimization problems*, where one, multiple, or many criteria are to be simultaneously optimized, typically leading to a significant increase in the number of optimal trade-offs to be identified. *Cross-domain optimization problems*, where one has to deal with continuous, integer, categorical variables, or even more complex structures such as permutations, strings, trees, or graphs, that may be mixed among themselves. *Expensive optimization problems*, where the propagation of environmental parameters or the requirement of heavy simulations makes it already computationally demanding to obtain the quality of one single candidate solution at the evaluation stage.

The general goal is hence to foster the next generation of optimization algorithms for solving such problems from the incoming "big" optimization era by precisely investigating the modeling, the algorithmic resolution as well as the fundamental and experimental analysis of massive optimization problems. Arguing that such massive optimization problems raise new challenges, in particular because of (a) their dimensionality in terms of variables, (b) of objectives, (c) their heterogeneity, and (d) their expensive and uncertain nature, our research perspectives strive after jointly addressing these aspects, and can be viewed following four interconnected scientific objectives.

LANDSCAPE-AWARE OPTIMIZATION ALGORITHMS. The class of optimization problems encountered in real-life complex application domains is wide and heterogeneous. This explains the plethora of (ad-hoc) optimization techniques specialized in solving a particular problem formulation. On the contrary, general-purpose optimization methods such as Branch-and-Bound (complete, but quickly impractical for large-size problems) and general-purpose search heuristics from computational intelligence (e.g. stochastic local search, metaheuristics, evolutionary algorithms) constitute upper-level methodologies that can be used as guiding strategies in designing underlying optimization algorithms. Our goal precisely lies in the foundation, analysis and intelligent design of enhanced general-purpose optimization algorithms, search paradigms and their design principles, as well as innovative ways of combining them. However, being effective and efficient in solving the target problem always requires a proper configuration and adaptation of the general-purpose optimization approach. As such, most algorithms continue to be designed on the basis of intuition, and require an intensive phase of trials and errors for parameter setting. One way of addressing this in practice is to rely on parameter tuning in order to automatically configure an optimization algorithm by finding the most appropriate parameter setting, specialized for a given set of problem instances. Complementarily, we aim at avoiding hyper-specialized approaches, and at improving the way we develop optimization algorithms by incorporating a more fundamental approach in their design process. Our goal is to understand the difficulties a given optimization approach has to face, and what makes it efficient, independently of the target application, by deriving high-level and relevant features able to catch problem difficulty by means of tools from fitness landscape analysis, statistics and machine learning data analysis. Such an analytics-driven methodology, based on fitness landscape analysis and extensive benchmarking efforts, would allow, not only to understand what makes a problem difficult or an optimization approach efficient, but also to predict the algorithm performance, to select the most appropriate configuration from an algorithm portfolio, and to adapt and improve the algorithm design for unknown optimization domain and prob-

lem instances. This can for instance lead to the establishment of cross-domain autonomous solver that can address the challenges of massive and big optimization problem.

MODEL–ASSISTED AND SIMULATION OPTIMIZATION. In expensive optimization, evaluating the quality of a candidate solution is particularly demanding computationally speaking and might even be uncertain or subject to noise. This is typically the case when the evaluation step corresponds to the result of a (black-box) complex system simulation, or because of the large number of environmental parameters encountered in multidisciplinary engineering design and innovation, as well as sustainable systems. In this context, existing algorithms from optimization and computational intelligence suffer from slow convergence, and their scalability then raises new scientific challenges. To overcome this, one interesting approach is to rely on surrogate models and machine learning algorithms in order to predict the solutions quality without necessarily and systematically computing their (expensive/uncertain) objective value(s). The goal here is to accelerate the convergence of the optimization process and to improve the quality of final solutions. More particularly, different issues have to be addressed including: the suitability of advanced statistical and machine learning meta-models for large-scale optimization, the choice of the output to be predicted by these meta-models, their prediction accuracy and their parameter sensibility, the uncertainties and inaccuracies occurring in their responses, the choice of the data set from which the meta-model learns from, and the integration of the learning phase within the optimization process. Estimation-of-distribution and other model-assisted computational intelligence algorithms will also have to be strengthened and renovated. This consist in explicitly modeling the key features (such as variable interactions) that impact solutions quality, and to use this model as an algorithm component in order to produce new candidate solutions with an expected improved quality. The challenge for such techniques is mainly to deal with the characteristics of massive optimization problems and to scale accurately along the objective and decision spaces. Finally, because of the target application context, the computational cost of designed approaches is anyway prohibitive. As a consequence, distributed approaches are mandatory for addressing these different issues, with an effective parallelization on high performance computing platforms, which comes with additional specific challenges as discussed later.

DECOMPOSITION–BASED OPTIMIZATION ALGORITHMS. Given the large-scale nature of the target applications and the underlying optimization problems, in terms of the number of variables and objectives, a natural answer is to decompose the original global massive optimization problem to be solved into several sub-problems for which solutions are computed and aggregated taking inspiration from the divide and conquer paradigm. However, setting up an effective decomposition-based optimization approach relies on the design and integration of several components that are to be configured accurately. Firstly, we need to better define the set of sub-problems to be solved cooperatively, by decomposing the original problem into a set of sub-problems within a smaller region of the variable space and/or the objective space, so as to increase the efficiency of the optimization process. One key aspect is to be able to interconnect the possibly large-scale variable space with the possibly large scale objective space and to find a high level and unified decomposition methodology, e.g., based on landscape features. Secondly, novel cooperative computational intelligence algorithms and mechanisms are still needed in order to solve each sub-problem, and to specify the local rules of interaction and cooperation governing the global optimization process. The basic idea is to view the solving of an optimization problem as a complex system operating at different local parts (the sub-problems), so that the overall global computational power is eventually larger than the sum of its parts. In our opinion, this simple principle is *the* key for the successful solving of massive optimization problems. This will in particular allow to continue taking full benefits from the large-scale distributed and parallel compute facilities. In addition, decomposition-based optimization approaches will open important challenges in the design of new ways of combining general-purpose search heuristics (e.g. stochastic

local search, meta-heuristics, evolutionary algorithms) with exact algorithms (e.g. branch-and-bound, mathematical programming, dynamic programming) and/or machine learning algorithms in order to solve massive optimization problems. In fact, there is still a gap to fill between all of these techniques and decomposition can constitute the bridge to interconnect them in a flexible and effective manner.

DECENTRALIZED OPTIMIZATION ALGORITHMS. The goal here is to push forward the design, study, and validation of generic approaches for massive and big optimization, through the investigation of appropriate techniques that can fit in the large-scale and distributed nature of modern compute facilities. On the one hand, the power of modern and massively parallel compute platforms is becoming both huge and increasingly available for the community. Distributed large-scale high-speed interconnected CPUs, together with multi-core processors, many-core accelerators and co-processors, are without a doubt increasingly popular and widely deployed, not only in high-performance dedicated clusters and grids, but also in non-expert oriented environments such as distributed workstations and cloud compute facilities. On the other hand, the characteristics of massive optimization give rise to difficult challenges, beyond the ability of commonly-used optimization algorithms. In this respect, there is evidence that parallel optimization and evolutionary computing will play a crucially important role in order to foster the next generation of optimization techniques, and to accelerate their impact. The challenge is then to foster the cross-fertilization of optimization algorithms, distributed algorithms and high performance and massively parallel computation. Expert knowledge about parallel computing helps in creating and deploying parallel algorithms for different types of architectures and devices, e.g., cloud, multi-core, GPUs, etc. However, this implies the need for a careful definition of proper benchmarks, software tools, and metrics to measure the behavior of algorithms in a meaningful way. From a purely parallel and high performance point-of-view, the deployment of the designed massive optimization approaches over a real parallel computing testbed poses several issues that need to be addressed. In our opinion, the main challenge is the scalability with respect to the number of computing resources which has to be addressed by integrating the advances made by the high performance community in order to avoid ad-hoc problem specific approaches, that will anyway fail to follow the rapid evolution of large scale modern compute facilities. Moreover, a conceptual separation between 'physical' parallelism and decentralized/distributed algorithms (whether implemented in parallel or not) is needed not only to better and fairly analyze the resulting algorithms, but also to offer a high level algorithmic design that can still be adapted to the ever-evolving large scale compute environments with the minimum re-engineering efforts. In particular, all the previously-mentioned issues, such as sub-problem solving, expensive evaluation, or model-assisted approaches, have be though while taking their effective parallelization into account, and the potential gain of having a large distributed computing power available.

To conclude, and as one can guess given the large spectrum of the target future challenges and objectives, my research follows a global and collective methodology that I am constantly developing, as evidenced by my collaborations and projects which strongly nourished my work and experience. My scientific ambition is in fact to contribute to the emergence of a new generation of optimization algorithms that are able to adapt to the complexity and heterogeneity of current and future systems and applications whose scales and costs pose as many new challenges as research opportunities.

A | EXTENDED CV

A.1 ACADEMIC POSITION

A.1.1 General Information

Birthday ◇ **20 November 1977, Maharès, Tunisia**
Family ◇ **Married (1 child)**
Research Labs ◇ **CRIStAL, CNRS UMR 9189, Lille, France**
Inria Lille – Nord Europe
The France/Japan MODO International Associated Lab
Contact ◇ **(0033) (03) 28 77 85 82 / 59 35 86 47** (office time)
(0033) (07) 81 57 12 97
Web ◇ **http://cristal.univ-lille.fr/~derbel**

A.1.2 Education and Academic Milestones

Since 2017 ◇ Member and Co-founder of the France/Japan International Associated Laboratory LIA MODO

2016-2017 ◇ CRCT at the national level (CNU Section 27), 1/2 year Sabbatical permission

2016-2020 ◇ PEDR (Bonus for doctoral supervision and research, rank: A)

2012-2016 ◇ PES (Bonus for research excellence, rank: A)

2010-2015 ◇ Co-director (and Co-creation) of MOCAD, Master 2 speciality, Computer Science Department, IEEA, Univ. Lille

Since 2007 ◇ Associate Professor, Univ. of Lille, France
BONUS (previously DOLPHIN) research group
CRIStAL CNRS UMR 9189, Inria Lille – Nord Europe

2006-2007 ◇ Assistant professor (ATER), Univ. Provence Aix-Marseille 1, France
MOVE team, LIF Laboratory, CNRS UMR 7279

2002-2006 ◇ PhD in Computer Science, LaBRI, CNRS UMR 5800, Univ. of Bordeaux 1, France
Title: Local aspect in Distributed Algorithms
Jury: P. Fraigniaud (Reviewer), D. Peleg (Reviewer), C. Gavoille (President), G. Melançon, Y. Métivier (Co-supervisor), M. Mosbah (Co-supervisor)
Team: Combinatorics and Algorithm (Theme: Distributed Algorithms)

2005-2006 ◇ Assistant professor (ATER), IUT Univ. of Bordeaux 1 (Computer Science Department)

2002 ◇ Engineer/MSc in Computer Science, ENSEIRB High School, Univ. of Bordeaux 1, France

1998 ◇ High preparatory school (Math MP*), Lycée du Parc, Lyon

A.2 DOCTORAL AND STUDENT SUPERVISION

A.2.1 PhD Students Supervision

▷ Mathieu DJAMAÏ

- Title: *Peer-to-Peer Branch-and-Bound in the Grid*
- Date : 01/10/2009 - 11/03/2013
- Co-supervisor: N. Melab
- Grant: MESR (French Government), Univ. Lille, EDSPI (doctoral school)
- Co-authored publications: IPDPS [DDM11a], ICSCS [DDM11b], [DDM13], finalist of the SCALE challenge (4th IEEE International Scalable Computing Challenge) at CCGRID (11th International Symposium on Cluster, Cloud and Grid, Newport Beach, USA, 2011).
- Research engineer, France.

▷ Trong Tuan VU

- Title: *Heterogeneity and locality-aware work stealing for large scale Branch-and-Bound irregular algorithms*
- Date: 01/10/2011 - 12/12/2014
- Co-supervisor: N. Melab
- Grant: Inria CORDI, HEMERA Project
- Co-authored publications: FGCS [VD16], CCGRID [VD14], LION [VDM13], CLUSTER [Vu+12]
- Current position: Research engineer, London, UK

▷ Christopher JANKEE

- Title: *Optimization and Distributed Adaptive Metaheuristics in a Parallel environment*
- Date: 01/10/2014 - xx/02/2018
- Supervisors: C. Fonlupt and S. Verel, Université du Liottoral Côte d'Opale, Calais
- Co-authored publications: PPSN [Jan+16], EA [Jan+15; Jan+17b], IJCCI [Jan+17a]

A.2.2 Invited and External PhD Student Supervision

▷ Oliver CUATE. PhD student CINVESTAV-IPN, Mexico (2013–). Bilateral ECOS Nord (France) / ANUIES (Mexico) project (see Section A.3). *Pareto exploration in many-objective optimization*. Supervisor: O. Schütze, Cinvestav, MX. Doctoral visit from Apr. to Jun. 2016. Co-authored publications: [Cua+17].

▷ Miyako SAGAWA. PhD student at Shinshu University, Nagano, Japan (2014-2018). JSP-MEXT and S3-BBO France/Japan bilateral project (see Section A.3). *Learning variable importance for many-objective optimization problems*. Supervisors: H. Aguirre and K. Tanaka, Shinshu Univ., JP. Doctoral visits: from October to November 2014 and from Apr. to Jun. 2016. Co-authored publications: [Sag+17; Sag+16].

▷ Martin DROZDIK. PhD student at Shinshu University, Nagano, Japan (2011-2015). JSP-MEXT France/Japan bilateral projects (see Section A.3). *Improvements, understanding and performance of multi-objective Differential Evolution*. Supervisors: H. Aguirre et K. Tanaka, Univ. Shinshu, JP. Doctoral visit: from Nov. 2013 to Sep. 2014. Co-authored publications: [Dro+14].

- Hiba YAHYAOUI. PhD student at the Univ. of Jendouba, Tunisia (2014-2017). *Adaptive metaheuristics with multiple neighborhoods*. Supervisor: S. Krichen. Doctoral visits: three month doctoral visits in 2013, 2014 and 2015. Co-authored publications: [Yah+15].
- Juan Jose PALACIOS ALONSO. PhD student at the Univ. of Oviedo, Spain (2012-2015). *Metaheuristic strategies for scheduling under uncertainty*. Supervisor: Camino Rodriguez Vela. Doctoral visit: from Oct. to Dec. 2014. Co-authored publications: [AD15].

A.2.3 Postdoctoral Supervision

- Saul ZAPOTECAS-MARTINEZ. JSPS-MEXT project (see Section A.3). *Decomposition-based multi-objective optimization*. Postdoctoral stay (1 year): from Nov. 2014 to Mar. 2015 and from Jun. 2015 to Dec. 2015. Co-authored publications: [Mar+15a; Mar+15b].
- Asim ALI. HEMERA Inria project (see Section A.3). *Peer-to-Peer algorithms for large scale combinatorial optimization*. Postdoctoral stay (1 year): from Oct. 2010 to Sep. 2011. Co-authored publications: [Vu+12].

A.2.4 Master 2 Student Supervision

- Alexandre VERKYNDT. 6 months internship (2016). Subject: *Expensive multi-objective decomposition using surrogates*.
- Alexandre VERKYNDT. 6 months project (2015). Subject: *Decomposition based multi-objective optimization*.
- Antoine ASSEMAN. 6 months project (2014). Subject: *Parallel Pareto Local Search*.
- Gauvain MARGUET. 6 months internship (2014). Subject: *Parallel multi-objective decomposition*.
- Gauvain MARQUET. 6 months internship (2013). Subject: *Evolutionary multi-objective decomposition*.
- Ghazi TEKAYA. 4 months internship (2012). Subject: *Routing in sensor networks*.
- Hiba YAHYAOUI. 6 months internship (2012). Subject: *Bringing Order in Variable Neighborhood Search*.
- Rémi DEGRUSON. 6 months project (2012). Subject: *Benchmarking multiobjectivization algorithm on COCO*.
- Dhoha GHRAB. 9 months internship (2011). Subject: *Graph coloring and hierarchical routing in sensor networks*.
- Mahmoud HAMMOUDA. 4 months internship (2009). Subject: *Self-optimization in radio networks*.
- Mathieu DJAMAÏ. 5 months internship (2009). Subject: *fully distributed Branch-and-Bound*.

A.2.5 Master 1 Student Supervision

- Valentin OWCZAREK. 4 months project (2015). Subject: *Extending and benchmarking SOO on COCO*.
- Delphine POUX. 4 months project (2015). Subject: *Greedy optimization algorithms for an agriculture problem*.

- Yoann DUFRESNE. 4 months project (2015). Subject: *Localized multi-objective optimization*.
- Luis Diego ARENAS PIMENTEL. 4 months internship (2010). Subject: *Radio network algorithm simulation and visualization*.
- Abhishek SINGH. 4 months internship (2009). Subject: *Distributed Combinatorial Optimization in Telecommunications*.
- Nicolas GOUVY and Pamela WATTEBLED. 4 months internship (2009). Subject: *Weighted graph distributed algorithm simulation and visualization*.

A.3 FUNDED PROJECTS AND SCIENTIFIC ANIMATION

A.3.1 Funded projects and scientific responsibilities

Funded projects

- ANR PRCI BigMO France / Hong Kong
  - Role: Principal investigator (Coordinator)
  - Title: BigMO / Big Multi-objective Optimization
  - Date: 2017 - 2021 (4 years)
  - Grant: co-funded by the ANR (FR) and the RGC (HK) agencies. HK coordinator: Qingfu Zhang. 240,000 Euros. (Seemingly the same budget is available for the partner in Hong Kong).
- International Associated Laboratory LIA MODO Lille / Shinshu
  - Role: Participant and co-founder
  - Title: MODO / Frontiers on Massive Optimization and Computational Intelligence
  - Date: 2017 - 2021 (4 years renewable)
  - Grant: local LIA between Univ. Lille (FR) and Shinshu University (JP). Manpower and budget are fixed on an annual basis by the institutions of the two partners. Co-directors: Hernan Aguirre (JP) et Arnaud Liefooghe (FR).
- Bilateral ECOS Nord / ANUIES, France / Mexico, project
  - Role: Participant and co-writer
  - Title: many-objective evolutionary optimization: applications to engineering and smart cities.
  - Date: 2016 - 2020 (4 years)
  - Grant: co-funded by the ECOS Nord (FR) and the ANUIES (MX) programs. Joint PhD funding; two weeks stay (senior researcher), and two-month stay (student junior researcher) per year.
- University Internationalization project
  - Role: Participant
  - Title: Massive optimization
  - Date: 2016 - 2018 (3 years)
  - Grant: Univ. Lille 1. 12,000 Euros. Bilateral France / Japan project. Principle Investigator: Arnaud Liefooghe.

▷ University BQR International project

- Role: Participant
- Title: Analytics Learning-driven multi-objective
- Date: 2016 (1 year)
- Grant: Univ. Lille 1. 3,000 Euros. Bilateral France / Japan project. Principle Investigator: Arnaud Liefooghe.

▷ PHC Procore bilateral, France / Hong Kong, project

- Role: Principal Investigator
- Title: Decomposition based multi-objective optimization
- Date: 2015 - 2017 (2 years)
- Grant: PHC Procore / RGC. 9,000 Euros per year. Same budget is available for the partner in Hong Kong.

▷ Ayame/Inria S3-BBO, France / Japan, associate team

- Role: Participant
- Title: Threefold scalability in any-objective black-box optimization
- Date: 2014 - 2017 (3 years renewable)
- Grant: JSPS (JP) and Inria (FR). Partners: Shinshu Univ. (JP), Tao (Inria Saclay, FR), Dolphin (Univ. Lille, Inria Lille, FR). 10,000 euros per year for the french partners and student internship support. Same budget is available for the partner in Japan. Coordinators: Hernan Aguirre (JP) and Anne Auger (FR).

▷ JSPS-MEXT bilateral, France / Japan, project

- Role: Participant
- Title: Global research on the framework of evolutionary solution search to accelerate innovation
- Date: 2013 - 2016 (2,5 years)
- Grant: JSPS (JP). Fundings for Univ. de Lille: 2,5 post-doctoral position; 1 year PhD stay. Participants: Univ. Shinshu (JP), Tao (Inria Saclay, FR), Dolphin (Univ. Lille, Inria Lille, FR), Univ. Du Littoral Côte d'Opale, Calais, FR. Principal investigator: K. Tanaka (JP).

▷ University BQR Emergent Research project.

- Role: Principal Investigator
- Title: Toward massive parallel optimization on hybrid P2P/GPU architectures
- Date: 2012 (1 year)
- Grant: Univ. Lille 1. 6,000 Euros.

▷ ADT Inria HEMERA large wingspan project

- Role: Coordinator of challenge A - COPS (1 over 14)
- Title: Large Scale Computing for Combinatorial Optimization Problems.
- Date: 2010 - 2014 (4 years)
- Grant: Inria. Three-year PhD fund (Trong Tuan Vu), one-year postdoc position (Asim Ali), et missions. Participants: several research teams involved in the Grid'5000 [Gri] experimental grid organized in 8 working groups and 14 scientific challenges. Coordinator: Christian Perez, Inria Lyon

- ▷ Inria STIC bilateral, France / Tunisia, project
  - Role: Principal Investigator
  - Title: Coloring and spanning structure for radio networks
  - Date: 2011 - 2013 (2 years)
  - Grant: Inria STIC. 10,000 Euros. Same budget is available for the partner in Tunisia.

Scientific animation

- ▷ Member of the restricted recruitment committee for associate professorship in computer science (CoS, section 27), University of Valenciennes, LAMIH, 2017.
- ▷ Member of the recruitment committee for associate professorship in computer science (section 27), University Lille 1, since 2010.
- ▷ Member of the Grid5000 GIS site-leaders committee (2013-2015)
- ▷ Member of the CLDD committee at Inria Lille (Commission Locale Développement Durable) (2012-2014)
- ▷ Member of the information system infrastructure working group, FIL, University Lille 1, since 2015.

A.3.2 Research dissemination and visibility

International visibility

- ▷ Associate Editor, IEEE Transactions on Systems, Man and Cybernetics: Systems, since 2016.
- ▷ Member of the IEEE CIS Task Force on Decomposition-based Techniques in Evolutionary Computation, since 2017.
- ▷ Co-Foundation of the International Associated Laboratory between the University of Lille (France) and Shinshu University (Japan): MODO / Frontiers on Massive Optimization and Computational Intelligence, inauguration, Jul. 2017. (see Section A.3)
- ▷ Establishment of the new moea/d website dedicated to multi-objective decomposition.
- ▷ Participation to the creation of the Inria associate team France / Japan (S3-BBO), 2014 (see Section A.3).
- ▷ Participation to the creation of the memorandum of understanding (MoU) between University of Lille and Shinshu University (Japan), Feb. 2014

Scientific Awards

- ▷ Our paper [Shi+17b] wins the best student paper award in SEAL 2017.
- ▷ Our paper [Mar+14] was nominated to the best paper award in PPSN 2014.
- ▷ Our paper [DT10b] wins the best paper award in ICDCN 2010.

International sabbatical and research visits

- One week visit to City University, Hong Kong, October 2017. Hosting professor: Qingfu Zhang.
- Three weeks visit to Shinshu University, Nagano, Jun. - Jul. 2017. Hosting professors: Kiyoshi Tanaka, Hernan Aguirre.
- Two months visit (CRCT) to IST (Instituto Superior Tecnico), University of Lisboa, Portugal; Jan. - Mar. 2017. Hosting professor: José Rui Figueira.
- One week visit to the University of Coimbra, Portugal, Feb. 2017. Hosting professor: Luis Paquete.
- One week visit to City University, Hong Kong, Oct. 2016. Hosting professor: Qingfu Zhang.
- One week visit to Shinshu University, Nagano, Japan, Dec. 2015. Hosting professors: Kiyoshi Tanaka, Hernan Aguirre.
- One week visit to Shinshu University, Nagano, Japan, May 2015. Hosting professors: Kiyoshi Tanaka, Hernan Aguirre.
- Two weeks visits to Shinshu University, Karuizawa, Japan, Dec. 2014. Hosting professors: Kiyoshi Tanaka, Hernan Aguirre.

Scientific organization committees

- Co-organization of the ADEMO (Advances in Decomposition-based Multiobjective Optimization) special session at CEC (International Congress on Evolutionary Computation). Vancouver Canada. 2016.
- Co-organization of the EMO@MCDM (Evolutionary Multi-objective Optimization) special session at MCDM (International conference on Multicriteria Decision Making). Malaga, Spain. 2013.
- Invited Editor EJOR (European Journal of Operation Research). Special issue on Evolutionary Multi-objective Optimization. 2013.
- Co-organization of the 8th school summer "Artificial Evolution". Quiberon, France. 2013.
- Co-organization of the (annual) grid'5000 summer school in Lille, 2010.
- Co-organization de annual grid'5000 day in Lille in 2011, 2010, 2009 and 2008.

Reviewing activities

- Member of the restricted recruitment committee for associate professorship in computer science (CoS, section 27), University of Valenciennes, LAMIH, 2017.
- Expert reviewer for the ANR french national agency (optimization, evolutionary algorithms, parallel and distributed algorithms) in 2010, 2014 and 2016.
- Expert reviewer for project proposal, University of Luxembourg, 2010.
- I am serving as a reviewer on a regular basis in a number of conferences and journals, e.g.,:
  - IEEE Transactions on Evolutionary Computation
  - IEEE Transactions on Systems Man and Cybernetics

- Applied Soft Computing
- Soft Computing
- Swarm and Evolutionary Computation
- Computers & Operations Research
- International Transactions in Operational Research
- The Computer Journal
- Computers & Industrial Engineering
- Computational Optimization and Applications
- RAIRO - Operations Research
- Simulation Modeling Practice and Theory
- GECCO (ACM Genetic and Evolutionary Computation Conference)
- CEC (IEEE Congress on Evolutionary Computation)
- EvoCOP (LNCS Evolutionary Computation in Combinatorial Optimization)

Main current senior collaborators

▷ International: Dr. H. Aguirre (Shinshu University, Japan), Pr. K. Tanaka (Shinshu University, Japan), Pr. Q. Zhang (City University, Hong Kong), Dr. O. Schuetze (CINVESTAV-IPN, Mexico), Pr. J. R. Figueira (IST, Univ. Lisboa, Portugal), Dr. L. F. C. Paquete (University of Coimbra, Portugal), Manuel Lopez-Ibanez (University of Manchester, UK), Saul Zapotecas Martinez (Mexico), Juan Jose Palacios Alonso (Spain)

▷ National: Pr. S. Verel (Univ. Littoral Côte d'Opale, Calais), Pr. C. Fonlupt (Univ. Littoral Côte d'Opale, Calais), Dr. Matthieu Basseur, Dr. Adrien Goëffon University of Angers (University of Angers), Dr. Dimo Brockhoff (CR, Inria Paris Saclay)

A.4 SUMMARY OF TEACHING ACTIVITIES

Responsibilities

▷ Co-creation (2010) and Co-responsible (2010-2015) of MOCAD (Complex Models, Algorithms, and Data), a Master 2 Speciality in the Computer Science Department, University of Lille.
This last year master speciality was proposed in the context of the 2010-2014 teaching offer. It replaces the previous existing research Master with a different organization and objective. The lectures proposed in MOCAD cover some topics addressed in a number of research teams in the CRIStAL Lab and Inria. We had the challenge of coordinating and organizing the lectures content, and promoting the speciality with respect to students and related organizations. A number of MOCAD student continue into the PhD program every year since its establishment.

▷ Creation, development and administration of the online student recruitment website at the Computer Science Department (FIL), since 2010.

▷ Responsible of different lectures in the first and second year Master, at the computer science department (See table below summarizing my main teaching involvements).

▷ Regular student project supervision.

Lectures Summary

Lecture	Type	Level	Year
Object-oriented design (COO)	TD/TP	L3 Info (FIL, IEEA, Univ. Lille)	2015/20–
Functional programming (PF)	TD/TP	L3 Info (FIL, IEEA, Univ. Lille)	2015/2016
Big-data Technology (TLDE)*	C/TD/TP	M2 Info MOCAD (FIL, IEEA, Univ. Lille)	2014/20–
Advanced Object-oriented design (COA)*	C/TD/TP	M1 Miage FA (FIL, IEEA, Univ. Lille)	2013/2015
Combinatorial Optimization (OC)*	C/TD/TP	M2 Info MOCAD (FIL, IEEA, Univ. Lille)	2010/20–
Cluster and grid computing (CGC)*	C/TD/TP	M2 Info TIIR (FIL, IEEA, Univ. Lille)	2009/2014
Algorithms and applications (AeA)*	C/TD/TP	M1 Info (FIL, IEEA, Univ. Lille)	2009/20–
Parallel and distributed programming (PPD)	TD/TP	M1 Info (FIL, IEEA, Univ. Lille)	2008/20–
Supervised Project (PJE)*	C/TD/TP	M1 Info (FIL, IEEA, Univ. Lille)	2007/20–
Design of distributed Web applications (CAR)*	C/TD/TP	M1 Miage (FIL, IEEA, Univ. Lille)	2007/2015
Object-oriented design (COO)*	C/TD/TP	M1 Miage (FIL, IEEA, Univ. Lille)	2007/2015
Network and Unix (STU)	TD/TP	M2 Info TIIR (FIL, IEEA, Univ. Lille)	2007/2009
Advanced Object-oriented design (COA)	TD/TP	M1 GMI (FIL, IEEA, Univ. Lille)	2007/2008
Networking	TD/TP	M1 Info (Univ. Marseille Aix 1)	2006/2007
Parallel and distributed programming	TD/TP	M1 Info (Univ. Marseille Aix 1)	2006/2007
Operation research	TP	M1 Info (Univ. Marseille Aix 1)	2006/2007
Data base administration	TD/TP	IUT 2 (Univ. Bordeaux 1)	2005/2006
Object-oriented design and UML	TD/TP	IUT 2 (Univ. Bordeaux 1)	2005/2006
Algorithms and programming	TD/TP	IUT 1 (Univ. Bordeaux 1)	2004/2005
Algorithms and data structures	TD	Engineer high school (ENSEIRB, Bordeaux)	2003/2005
Object-oriented programming*	C/TD/TP	Engineer high school (ENSEIRB, Bordeaux)	2003/2005
Human computer interfaces*	C/TP	Engineer high school (ENSEIRB, Bordeaux)	2003/2004

*: Responsible of the lecture organization
C: Main lecturer (Cours)
TD: Exercise session (Travaux dirigés)
TP: Practical programming session (Travaux pratique)

B PERSONNEL BIBLIOGRAPHY AFTER PHD

B.1 BOOK CHAPTER

[DDM13] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Large sclae P2P-Inspired Problem solving: a formal and experimental study." In: *Large Scale Network-Centric Distributed Systems*. Ed. by A. Y. Zomaya and H. Sarbazi-Azad. John Wiley & Sons, 2013, pp. 73–102.

B.2 INTERNATIONAL PEER REVIEWED JOURNALS

[VD16] Trong-Tuan Vu and Bilel Derbel. "Parallel Branch-and-Bound in multi-core multi-CPU multi-GPU heterogeneous environments." In: *Future Generation Comp. Syst.* 56 (2016), pp. 95–109.

[Jem+15] Imen Jemili, Dhouha Ghrab, Amine Dhraief, Abdelfettah Belghith, Bilel Derbel, Ahmed S. Al-Mogren, and Hassan Mathkour. "CHRA: a coloring based hierarchical routing algorithm." In: *J. Ambient Intelligence and Humanized Computing* 6.1 (2015), pp. 69–82.

[Der+14b] Bilel Derbel, Jérémie Humeau, Arnaud Liefooghe, and Sébastien Verel. "Distributed Localized Bi-objective Search." In: *European Journal of Operational Research* 239 (2014), pp. 731–743.

[DMZ10] Bilel Derbel, Mohamed Mosbah, and Akka Zemmari. "Sublinear Fully Distributed Partition with Applications." In: *Theory Comput. Syst.* 47.2 (2010), pp. 368–404.

[DG08] Bilel Derbel and Cyril Gavoille. "Fast deterministic distributed algorithms for sparse spanners." In: *Theor. Comput. Sci.* 399.1-2 (2008), pp. 83–100.

B.3 INTERNATIONAL CONFERENCES WITH COMMITTEE AND PROCEEDINGS

[Cua+17] Oliver Cuate, Bilel Derbel, Arnaud Liefooghe, El-Ghazali Talbi, and Oliver Schütze. "An Approach for the Local Exploration of Discrete Many Objective Optimization Problems." In: *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*. 2017, pp. 135–150.

[Jan+17a] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "Analysis of a batch strategy for a Master-Worker adaptive selection algorithm framework." In: *The 9th International Joint Conference on Computational Intelligence (IJCCI)*. 2017.

[Jan+17b] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "On the Design of a Master-Worker Adaptive Algorithm Selection Framework." In: *The 15th LNCS International conference on Artificial Evolution (EA)*. 2017.

[Lie+17a] Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka. "A Fitness Landscape Analysis of Pareto Local Search on Bi-objective Permutation Flowshop Scheduling Problems." In: *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*. 2017, pp. 422–437.

[Lie+17b] Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka. "Towards Landscape-Aware Automatic Algorithm Configuration: Preliminary Experiments on Neutral and Rugged Landscapes." In: *Evolutionary Computation in Combinatorial Optimization - 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*. 2017, pp. 215–232.

[Mon+17] Hugo Monzón, Hernán E. Aguirre, Sébastien Vérel, Arnaud Liefooghe, Bilel Derbel, and Kiyoshi Tanaka. "Closed state model for understanding the dynamics of MOEAs." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*. 2017, pp. 609–616.

[Sag+17] Miyako Sagawa, Hernán E. Aguirre, Fabio Daolio, Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, and Kiyoshi Tanaka. "Learning variable importance to guide recombination on many-objective optimization." In: *5th International Conference on Smart Computing and Artificial Intelligence (SCAI)*. 2017, to appear.

[Shi+17a] Jialong Shi, Qingfu Zhang, Bilel Derbel, and Arnaud Liefooghe. "A Parallel Tabu Search for the Unconstrained Binary Quadratic Programming problem." In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 2017, pp. 557–564.

[Shi+17b] Jialong Shi, Qingfu Zhang, Bilel Derbel, and Arnaud Liefooghe. "Using Parallel Strategies to Speed Up Pareto Local Search." In: *The 11th International Conference on Simulated Evolution and Learning*. 2017, to appear.

[Bas+16] Matthieu Basseur, Bilel Derbel, Adrien Goëffon, and Arnaud Liefooghe. "Experiments on Greedy and Local Search Heuristics for *d*dimensional Hypervolume Subset Selection." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. 2016, pp. 541–548.

[Der+16] Bilel Derbel, Arnaud Liefooghe, Qingfu Zhang, Hernán E. Aguirre, and Kiyoshi Tanaka. "Multi-objective Local Search Based on Decomposition." In: *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. 2016, pp. 431–441.

[Jan+16] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "A Fitness Cloud Model for Adaptive Metaheuristic Selection Methods." In: *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. 2016, pp. 80–90.

[LD16] Arnaud Liefooghe and Bilel Derbel. "A Correlation Analysis of Set Quality Indicator Values in Multiobjective Optimization." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. 2016, pp. 581–588.

[Sag+16] Miyako Sagawa, Hernán E. Aguirre, Fabio Daolio, Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, and Kiyoshi Tanaka. "Learning variable importance to guide recombination." In: *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, December 6-9, 2016*. 2016, pp. 1–7.

[AD15] Juan José Palacios Alonso and Bilel Derbel. "On Maintaining Diversity in MOEA/D: Application to a Biobjective Combinatorial FJSP." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*. 2015, pp. 719–726.

[Der+15] Bilel Derbel, Arnaud Liefooghe, Gauvain Marquet, and El-Ghazali Talbi. "A fine-grained message passing MOEA/D." In: *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. 2015, pp. 1837–1844.

[DP15] Bilel Derbel and Philippe Preux. "Simultaneous optimistic optimization on the noiseless BBOB testbed." In: *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. 2015, pp. 2010–2017.

[Jan+15] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies." In: *Artificial Evolution - 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*. 2015, pp. 83–96.

[Mar+15a] Saúl Zapotecas Martínez, Bilel Derbel, Arnaud Liefooghe, Hernán E. Aguirre, and Kiyoshi Tanaka. "Geometric Differential Evolution in MOEA/D: A Preliminary Study." In: *Advances in Artificial Intelligence and Soft Computing - 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, October 25-31, 2015, Proceedings, Part I*. 2015, pp. 364–376.

[Mar+15b] Saúl Zapotecas Martínez, Bilel Derbel, Arnaud Liefooghe, Dimo Brockhoff, Hernán E. Aguirre, and Kiyoshi Tanaka. "Injecting CMA-ES into MOEA/D." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*. 2015, pp. 783–790.

[Yah+15] Hiba Yahyaoui, Saoussen Krichen, Bilel Derbel, and El-Ghazali Talbi. "A Hybrid ILS-VND Based Hyper-heuristic for Permutation Flowshop Scheduling Problem." In: *19th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2015, Singapore, 7-9 September 2015*. 2015, pp. 632–641.

[Der+14a] Bilel Derbel, Dimo Brockhoff, Arnaud Liefooghe, and Sébastien Vérel. "On the Impact of Multiobjective Scalarizing Functions." In: *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. 2014, pp. 548–558.

[Dro+14] Martin Drozdik, Kiyoshi Tanaka, Hernán E. Aguirre, Sébastien Vérel, Arnaud Liefooghe, and Bilel Derbel. "An Analysis of Differential Evolution Parameters on Rotated Bi-objective Optimization Functions." In: *Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*. 2014, pp. 143–154.

[Mar+14] Gauvain Marquet, Bilel Derbel, Arnaud Liefooghe, and El-Ghazali Talbi. "Shake Them All! - Rethinking Selection and Replacement in MOEA/D." In: *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. 2014, pp. 641–651.

[VD14] Trong-Tuan Vu and B. Derbel. "Link-Heterogeneous Work Stealing." In: 14^{th} *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 2014, pp. 354–363.

[DBL13] Bilel Derbel, Dimo Brockhoff, and Arnaud Liefooghe. "Force-Based Cooperative Search Directions in Evolutionary Multi-objective Optimization." In: *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*. 2013, pp. 383–397.

[Ghr+13] Dhouha Ghrab, Bilel Derbel, Imen Jemili, Amine Dhraief, Abdelfettah Belghith, and El-Ghazali Talbi. "Coloring based Hierarchical Routing Approach." In: *Proceedings of the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013), Halifax, Nova Scotia, Canada, June 25-28, 2013*. 2013, pp. 188–196.

[Jem+13] Imen Jemili, Dhouha Ghrab, Abdelfettah Belghith, Bilel Derbel, and Amine Dhraief. "Collision aware coloring algorithm for wireless sensor networks." In: *2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013, Sardinia, Italy, July 1-5, 2013*. 2013, pp. 1546–1553.

[TBD13] Thanh-Do Tran, Dimo Brockhoff, and Bilel Derbel. "Multiobjectivization with NSGA-ii on the noiseless BBOB testbed." In: *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*. 2013, pp. 1217–1224.

[VDM13] Trong-Tuan Vu, Bilel Derbel, and Nouredine Melab. "Adaptive Dynamic Load Balancing in Heterogeneous Multiple GPUs-CPUs Distributed Setting: Case Study of B&B Tree Search." In: 7^{th} *International Conference on Learning and Intelligent Optimization (LION)*. 2013, pp. 87–103.

[DD12] Houda Derbel and Bilel Derbel. "On neighborhood tree search." In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. 2012, pp. 1261–1268.

[Vu+12] Trong-Tuan Vu, Bilel Derbel, Ali Asim, Ahcene Bendjoudi, and Nouredine Melab. "Overlay-Centric Load Balancing: Applications to UTS and B&B." In: 14^{th} *IEEE International Conference on Cluster Computing (CLUSTER)*. 2012, pp. 382–390.

[DV11] Bilel Derbel and Sébastien Verel. "DAMS: Distributed Adaptive Metaheuristic Selection." In: *Genetic And Evolutionary Computation Conference (GECCO)*. Dublin, Ireland: ACM, 2011, pp. 1955–1962.

[DDM11a] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Distributed B&B: A Pure Peer-to-Peer Approach." In: *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May 2011 - Workshop Proceedings*. 2011, pp. 1788–1797.

[DDM11b] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Impact of logical overlay upon a Pure P2P approach for the B&B algorithm." In: *IEEE Inter. Conf. on Systems and Computer Science (ICSCS'11)*. 2011.

[DT10a] Bilel Derbel and El-Ghazali Talbi. "Distributed Node Coloring in the SINR Model." In: *2010 International Conference on Distributed Computing Systems, ICDCS 2010, Genova, Italy, June 21-25, 2010*. 2010, pp. 708–717.

[DT10b] Bilel Derbel and El-Ghazali Talbi. "Radio Network Distributed Algorithms in the Unknown Neighborhood Model." In: *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3-6, 2010. Proceedings*. 2010, pp. 155–166.

[Der+09] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. "Local Computation of Nearly Additive Spanners." In: *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*. 2009, pp. 176–190.

[Der08] Bilel Derbel. "Local Maps: New Insights into Mobile Agent Algorithms." In: *Distributed Computing, 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings*. 2008, pp. 121–136.

[Der+08] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. "On the locality of distributed sparse spanner construction." In: *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*. 2008, pp. 273–282.

[DMGo8] Bilel Derbel, Mohamed Mosbah, and Stefan Gruner. "Mobile Agents Implementing Local Computations in Graphs." In: *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*. 2008, pp. 99–114.

C BIBLIOGRAPHY

[Ama] Amazon Inc. *High Performance Computing (HPC). URL: http://aws.amazon.com/ec2/hpc-applications/.*

[Gri] Grid'5000. *Grid'5000 French national experimental gird. URL: https://www.grid5000.fr/.*

[Int] Intel. *Intel Threading Building Blocks*.

[MOE] MOEA/D website. *Website dedicated to decomposition based multi-objective optimization. URL: https://sites.google.com/view/moead/home*.

[Top] Top500. *Top500 SuperComputers. URL: https://www.top500.com/.*

[Cas+17] Olacir R. Castro, Roberto Santana, José Antonio Lozano, and Aurora Pozo. "Combining CMA-ES and MOEA/DD for many-objective optimization." In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 2017, pp. 1451–1458.

[CML17] Josu Ceberio, Alexander Mendiburu, and José Antonio Lozano. "Are we generating instances uniformly at random?" In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 2017, pp. 1645–1651.

[Cer+17] Audrey Cerqueus, Xavier Gandibleux, Anthony Przybylski, and Frédéric Saubion. "On branching heuristics for the bi-objective 0/1 unidimensional knapsack problem." In: *J. Heuristics* 23.5 (2017), pp. 285–319.

[Cua+17] Oliver Cuate, Bilel Derbel, Arnaud Liefooghe, El-Ghazali Talbi, and Oliver Schütze. "An Approach for the Local Exploration of Discrete Many Objective Optimization Problems." In: *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*. 2017, pp. 135–150.

[Gmy+17] Jan Gmys, Mohand Mezmaz, Nouredine Melab, and Daniel Tuyttens. "IVM-based parallel branch-and-bound using hierarchical work stealing on multi-GPU systems." In: *Concurrency and Computation: Practice and Experience* 29.9 (2017).

[Jan17] Christopher Jankee. *PhD Thesis*. PhD Thesis. Calais, France, 2017.

[Jan+17a] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "Analysis of a batch strategy for a Master-Worker adaptive selection algorithm framework." In: *The 9th International Joint Conference on Computational Intelligence (IJCCI)*. 2017.

[Jan+17b] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "On the Design of a Master-Worker Adaptive Algorithm Selection Framework." In: *The 15th LNCS International conference on Artificial Evolution (EA)*. 2017.

[Lar+17] J. L. J. Laredo, F. Guinand, D. Olivier, and P. Bouvry. "Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing." In: *IEEE Transactions on Parallel and Distributed Systems* 28.2 (2017), pp. 517–529.

[Lie+17a] Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka. "A Fitness Landscape Analysis of Pareto Local Search on Bi-objective Permutation Flowshop Scheduling Problems." In: *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*. 2017, pp. 422–437.

[Lie+17b] Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka. "Towards Landscape-Aware Automatic Algorithm Configuration: Preliminary Experiments on Neutral and Rugged Landscapes." In: *Evolutionary Computation in Combinatorial Optimization - 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*. 2017, pp. 215–232.

[Mon+17] Hugo Monzón, Hernán E. Aguirre, Sébastien Vérel, Arnaud Liefooghe, Bilel Derbel, and Kiyoshi Tanaka. "Closed state model for understanding the dynamics of MOEAs." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*. 2017, pp. 609–616.

[Sag+17] Miyako Sagawa, Hernán E. Aguirre, Fabio Daolio, Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, and Kiyoshi Tanaka. "Learning variable importance to guide recombination on many-objective optimization." In: *5th International Conference on Smart Computing and Artificial Intelligence (SCAI)*. 2017, to appear.

[Shi+17a] Jialong Shi, Qingfu Zhang, Bilel Derbel, and Arnaud Liefooghe. "A Parallel Tabu Search for the Unconstrained Binary Quadratic Programming problem." In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 2017, pp. 557–564.

[Shi+17b] Jialong Shi, Qingfu Zhang, Bilel Derbel, and Arnaud Liefooghe. "Using Parallel Strategies to Speed Up Pareto Local Search." In: *The 11th International Conference on Simulated Evolution and Learning*. 2017, to appear.

[Tri+17] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh. "A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition." In: *IEEE Transactions on Evolutionary Computation* 21.3 (2017), pp. 440–462.

[Vu+17] Ky Khac Vu, Claudia D'Ambrosio, Youssef Hamadi, and Leo Liberti. "Surrogate-based methods for black-box optimization." In: *ITOR* 24.3 (2017), pp. 393–424.

[YJJ17] Shengxiang Yang, Shouyong Jiang, and Yong Jiang. "Improving the multiobjective evolutionary algorithm based on decomposition with new penalty schemes." In: *Soft Computing* 21.16 (2017), pp. 4677–4691.

[Bas+16] Matthieu Basseur, Bilel Derbel, Adrien Goëffon, and Arnaud Liefooghe. "Experiments on Greedy and Local Search Heuristics for *d*dimensional Hypervolume Subset Selection." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. 2016, pp. 541–548.

[Bis+16] Bernd Bischl et al. "ASlib: A benchmark library for algorithm selection." In: *Artificial Intelligence* 237.Supplement C (2016), pp. 41–58.

[CM16] Imen Chakroun and Nouredine Melab. "HB&B@GRID: An heterogeneous grid-enabled Branch and Bound algorithm." In: *International Conference on High Performance Computing & Simulation, HPCS 2016, Innsbruck, Austria, July 18-22, 2016*. 2016, pp. 697–704.

[Che+16] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. "A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization." In: *IEEE Transactions on Evolutionary Computation* 20.5 (2016), pp. 773–791.

[Der+16] Bilel Derbel, Arnaud Liefooghe, Qingfu Zhang, Hernán E. Aguirre, and Kiyoshi Tanaka. "Multi-objective Local Search Based on Decomposition." In: *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. 2016, pp. 431–441.

[GLS16] Adrien Goëffon, Frédéric Lardeux, and Frédéric Saubion. "Simulating non-stationary operators in search algorithms." In: *Applied Soft Computing* 38 (2016), pp. 257–268.

[Jan+16] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "A Fitness Cloud Model for Adaptive Metaheuristic Selection Methods." In: *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. 2016, pp. 80–90.

[LD16] Arnaud Liefooghe and Bilel Derbel. "A Correlation Analysis of Set Quality Indicator Values in Multiobjective Optimization." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. 2016, pp. 581–588.

[Lóp+16] M. López-Ibáñez, J. Dubois-Lacoste, L. Cáceres, M. Birattari, and T. Stützle. "The irace package: Iterated racing for automatic algorithm configuration." In: *Oper Res Pers* 3 (2016), pp. 43–58.

[Sag+16] Miyako Sagawa, Hernán E. Aguirre, Fabio Daolio, Arnaud Liefooghe, Bilel Derbel, Sébastien Vérel, and Kiyoshi Tanaka. "Learning variable importance to guide recombination." In: *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, December 6-9, 2016*. 2016, pp. 1–7.

[Ver16] Sébastien Verel. *Apport à l'analyse des paysages de fitness pour l'optimisation mono-objective et multiobjective*. HDR Thesis. Calais, France, 2016.

[VD16] Trong-Tuan Vu and Bilel Derbel. "Parallel Branch-and-Bound in multi-core multi-CPU multi-GPU heterogeneous environments." In: *Future Generation Comp. Syst.* 56 (2016), pp. 95–109.

[Wan+16] L. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao. "Constrained Subproblems in a Decomposition-Based Multiobjective Evolutionary Algorithm." In: *IEEE Transactions on Evolutionary Computation* 20.3 (2016), pp. 475–480.

[WZZ16] R. Wang, Q. Zhang, and T. Zhang. "Decomposition-Based Algorithms Using Pareto Adaptive Scalarizing Methods." In: *IEEE Transactions on Evolutionary Computation* 20.6 (2016), pp. 821–837.

[Wu+16] Xiuli Wu, Pietro Consoli, Leandro Minku, Gabriela Ochoa, and Xin Yao. "An Evolutionary Hyper-heuristic for the Software Project Scheduling Problem." In: *PPSN XIV*. Springer International Publishing, 2016.

[AD15] Juan José Palacios Alonso and Bilel Derbel. "On Maintaining Diversity in MOEA/D: Application to a Biobjective Combinatorial FJSP." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*. 2015, pp. 719–726.

[BLS15] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. "Comparing Decomposition-Based and Automatically Component-Wise Designed Multi-Objective Evolutionary Algorithms." In: *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings, Part I*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Cham: Springer International Publishing, 2015, pp. 396–410.

[CM15] Imen Chakroun and Nouredine Melab. "Towards a heterogeneous and adaptive parallel Branch-and-Bound algorithm." In: *J. Comput. Syst. Sci.* 81.1 (2015), pp. 72–84.

[Chi+15] Francisco Chicano, Andrew M. Sutton, L. Darrell Whitley, and Enrique Alba. "Fitness Probability Distribution of Bit-Flip Mutation." In: *Evolutionary Computation* 23.2 (2015). PMID: 24885680, pp. 217–248.

[Der+15] Bilel Derbel, Arnaud Liefooghe, Gauvain Marquet, and El-Ghazali Talbi. "A fine-grained message passing MOEA/D." In: *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. 2015, pp. 1837–1844.

[DP15] Bilel Derbel and Philippe Preux. "Simultaneous optimistic optimization on the noiseless BBOB testbed." In: *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. 2015, pp. 2010–2017.

[Di +15] Giacomo Di Tollo, Frédéric Lardeux, Jorge Maturana, and Frédéric Saubion. "An experimental study of adaptive control for evolutionary algorithms." In: *Applied Soft Computing* 35 (2015), pp. 359–372.

[DLS15] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. "Anytime Pareto local search." In: *European Journal of Operational Research* 243.2 (2015), pp. 369–385.

[Jan+15] Christopher Jankee, Sébastien Vérel, Bilel Derbel, and Cyril Fonlupt. "Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies." In: *Artificial Evolution - 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*. 2015, pp. 83–96.

[Jem+15] Imen Jemili, Dhouha Ghrab, Amine Dhraief, Abdelfettah Belghith, Bilel Derbel, Ahmed S. Al-Mogren, and Hassan Mathkour. "CHRA: a coloring based hierarchical routing algorithm." In: *J. Ambient Intelligence and Humanized Computing* 6.1 (2015), pp. 69–82.

[Li+15a] H. Li, M. Ding, J. Deng, and Q. Zhang. "On the use of random weights in MOEA/D." In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2015, pp. 978–985.

[Li+15b] K. Li, K. Deb, Q. Zhang, and S. Kwong. "An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition." In: *IEEE Transactions on Evolutionary Computation* 19.5 (2015), pp. 694–716.

[Mar+15a] Saúl Zapotecas Martínez, Bilel Derbel, Arnaud Liefooghe, Hernán E. Aguirre, and Kiyoshi Tanaka. "Geometric Differential Evolution in MOEA/D: A Preliminary Study." In: *Advances in Artificial Intelligence and Soft Computing - 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, October 25-31, 2015, Proceedings, Part I*. 2015, pp. 364–376.

[Mar+15b] Saúl Zapotecas Martínez, Bilel Derbel, Arnaud Liefooghe, Dimo Brockhoff, Hernán E. Aguirre, and Kiyoshi Tanaka. "Injecting CMA-ES into MOEA/D." In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*. 2015, pp. 783–790.

[Muñ+15] Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges." In: *Information Sciences* 317 (2015), pp. 224–245.

[Sat15a] Hiroyuki Sato. "Analysis of inverted PBI and comparison with other scalarizing functions in decomposition based MOEAs." In: *Journal of Heuristics* 21.6 (2015), pp. 819–849.

[Sat15b] Hiroyuki Sato. "Analysis of inverted PBI and comparison with other scalarizing functions in decomposition based MOEAs." In: *J. Heuristics* 21.6 (2015), pp. 819–849.

[Sor+15] Jorge A Soria-Alcaraz, Gabriela Ochoa, Adrien Göeffon, Frédéric Lardeux, and Frédéric Saubion. "Combining Mutation and Recombination to Improve a Distributed Model of Adaptive Operator Selection." In: *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer. 2015, pp. 97–108.

[Sud15] Dirk Sudholt. "Parallel Evolutionary Algorithms." In: *Springer Handbook of Computational Intelligence*. Ed. by Janusz Kacprzyk and Witold Pedrycz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 929–959.

[WZZ15] Rui Wang, Qingfu Zhang, and Tao Zhang. "Pareto Adaptive Scalarising Functions for Decomposition Based Algorithms." In: *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings, Part I*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Cham: Springer International Publishing, 2015, pp. 248–262.

[Yah+15] Hiba Yahyaoui, Saoussen Krichen, Bilel Derbel, and El-Ghazali Talbi. "A Hybrid ILS-VND Based Hyper-heuristic for Permutation Flowshop Scheduling Problem." In: *19th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2015, Singapore, 7-9 September 2015*. 2015, pp. 632–641.

[Zap+15] S. Zapotecas-Martinez, H. E. Aguirre, K. Tanaka, and C. A. C. Coello. "On the low-discrepancy sequences and their use in MOEA/D for high-dimensional objective spaces." In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2015, pp. 2835–2842.

[BP14] Petr Baudiš and Pet Pošìk. "Online Black-Box Algorithm Portfolios for Continuous Optimization." In: *Parallel Problem Solving from Nature–PPSN XIII*. Springer, 2014, pp. 40–49.

[Der+14a] Bilel Derbel, Dimo Brockhoff, Arnaud Liefooghe, and Sébastien Vérel. "On the Impact of Multiobjective Scalarizing Functions." In: *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. 2014, pp. 548–558.

[Der+14b] Bilel Derbel, Jérémie Humeau, Arnaud Liefooghe, and Sébastien Verel. "Distributed Localized Bi-objective Search." In: *European Journal of Operational Research* 239 (2014), pp. 731–743.

[DKB14] Thomas Desautels, Andreas Krause, and Joel W. Burdick. "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3873–3923.

[DD14] Benjamin Doerr and Carola Doerr. "The impact of random initialization on the runtime of randomized search heuristics." In: *GECCO '14 - Conference on Genetic and Evolutionary Computation*. ACM. Vancouver, Canada: ACM, 2014, pp. 1375–1382.

[Dro+14] Martin Drozdik, Kiyoshi Tanaka, Hernán E. Aguirre, Sébastien Vérel, Arnaud Liefooghe, and Bilel Derbel. "An Analysis of Differential Evolution Parameters on Rotated Bi-objective Optimization Functions." In: *Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*. 2014, pp. 143–154.

[Fer+14] Carlos M. Fernandes, Juan LJ Laredo, Juan Julian Merelo, Carlos Cotta, Rafael Nogueras, and Agostinho C. Rosa. "Shuffle and Mate: A Dynamic Model for Spatially Structured Evolutionary Algorithms." In: *Parallel Problem Solving from Nature–PPSN XIII*. Springer, 2014, pp. 50–59.

[Gar+14] Mario García-Valdez, Leonardo Trujillo, Juan Julián Merelo-Guérvos, and Francisco Fernández-de-Vega. "Randomized Parameter Settings for Heterogeneous Workers in a Pool-Based Evolutionary Algorithm." In: *Parallel Problem Solving from Nature–PPSN XIII*. Springer, 2014, pp. 702–710.

[KZB14] Liangjun Ke, Qingfu Zhang, and R. Battiti. "Hybridization of Decomposition and Local Search for Multiobjective Optimization." In: *IEEE Transactions on Cybernetics* 44.10 (2014), pp. 1808–1820.

[Kot14] Lars Kotthoff. "Algorithm Selection for Combinatorial Search Problems: A Survey." In: *AI Magazine* (2014).

[Ler+14] Rudi Leroy, Mohand Mezmaz, Nouredine Melab, and Daniel Tuyttens. "Work Stealing Strategies For Multi-Core Parallel Branch-and-Bound Algorithm Using Factorial Number System." In: *Programming Models and Applications on Multicores and Manycores (PMAM)*. Orlando, FL, USA: ACM, 2014, 111:111–111:119.

[Li+02] Ke Li, Alvaro Fialho, Sam Kwong, and Qingfu Zhang. "Adaptive Operator Selection With Bandits for a Multiobjective Evolutionary Algorithm Based on Decomposition." In: *IEEE Transactions on Evolutionary Computation* 18.1 (2014-02), pp. 114–130.

[LGZ14] H. L. Liu, F. Gu, and Q. Zhang. "Decomposition of a Multiobjective Optimization Problem Into a Number of Simple Multiobjective Subproblems." In: *IEEE Transactions on Evolutionary Computation* 18.3 (2014), pp. 450–455.

[ME14] KatherineM. Malan and AndriesP. Engelbrecht. "Fitness Landscape Analysis for Metaheuristic Performance Prediction." In: *Recent Advances in the Theory and Application of Fitness Landscapes*. Ed. by Hendrik Richter and Andries Engelbrecht. Vol. 6. Emergence, Complexity and Computation. Springer Berlin Heidelberg, 2014, pp. 103–132.

[MI14] Andrea Mambrini and Dario Izzo. "PaDe: A Parallel Algorithm Based on the MOEA/D Framework and the Island Model." In: *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. 2014, pp. 711–720.

[Mar+14] Gauvain Marquet, Bilel Derbel, Arnaud Liefooghe, and El-Ghazali Talbi. "Shake Them All! - Rethinking Selection and Replacement in MOEA/D." In: *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. 2014, pp. 641–651.

[Mez+14] Mohand Mezmaz, Rudi Leroy, Nouredine Melab, and Daniel Tuyttens. "A Multi-core Parallel Branch-and-Bound Algorithm Using Factorial Number System." In: *IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*. 2014, pp. 1203–1212.

[Mun14] Rémi Munos. "From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning." In: *Foundations and Trends in Machine Learning* 7.1 (2014), pp. 1–129.

[Qi+14] Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun, and Jianshe Wu. "MOEA/D with adaptive weight adjustment." In: *Evolutionary computation* 22.2 (2014), pp. 231–264.

[RE14] Hendrik Richter and Andries Engelbrecht, eds. *Recent Advances in the Theory and Application of Fitness Landscapes*. Emergence, Complexity and Computation. Springer, 2014.

[Sil+14] J.M.N. Silva, C. Boeres, L.M.A. Drummond, and A.A. Pessoa. "Memory aware load balance strategy on a parallel branch-and-bound application." In: *Concurrency and Computation: Practice and Experience* (2014).

[Vu14] Trong-Tuan Vu. *Heterogeneity and locality-aware work stealing for large scale Branch-and-Bound irregular algorithms*. PhD Thesis. Lille, France, 2014.

[VD14] Trong-Tuan Vu and B. Derbel. "Link-Heterogeneous Work Stealing." In: 14^{th} *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 2014, pp. 354–363.

[Bur+13] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. "Hyper-heuristics: a survey of the state of the art." In: *JORS* 64.12 (2013), pp. 1695–1724.

[Can+13] Caner Candan, Adrien Goëffon, Frédéric Lardeux, and Frédéric Saubion. "Parameter Setting with Dynamic Island Models." In: *Learning and Intelligent Optimization*. Springer, 2013, pp. 253–258.

[Cha13] I. Chakroun. *Parallel heterogeneous Branch and Bound algorithms for multi-core and multi-GPU environments*. PhD Thesis. Lille, France, 2013.

[DBL13] Bilel Derbel, Dimo Brockhoff, and Arnaud Liefooghe. "Force-Based Cooperative Search Directions in Evolutionary Multi-objective Optimization." In: *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*. 2013, pp. 383–397.

[Dja13] M. Djamai. *Algorithmes Branch-and-Bound Pair-a-Pair pour grilles de calcul*. PhD Thesis. Lille, France, 2013.

[DDM13] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Large sclae P2P-Inspired Problem solving: a formal and experimental study." In: *Large Scale Network-Centric Distributed Systems*. Ed. by A. Y. Zomaya and H. Sarbazi-Azad. John Wiley & Sons, 2013, pp. 73–102.

[DB13] Bernabé Dorronsoro and Pascal Bouvry. "Cellular genetic algorithms without additional parameters." In: *The Journal of Supercomputing* 63.3 (2013), pp. 816–835.

[Dor+13] Bernabé Dorronsoro, Grégoire Danoy, Antonio J. Nebro, and Pascal Bouvry. "Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution." In: *Computers & Operations Research* 40.6 (2013). Emergent Nature Inspired Algorithms for Multi-Objective Optimization, pp. 1552–1563.

[Ghr+13] Dhouha Ghrab, Bilel Derbel, Imen Jemili, Amine Dhraief, Abdelfettah Belghith, and El-Ghazali Talbi. "Coloring based Hierarchical Routing Approach." In: *Proceedings of the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013), Halifax, Nova Scotia, Canada, June 25-28, 2013*. 2013, pp. 188–196.

[GPF13] Ioannis Giagkiozis, Robin C. Purshouse, and Peter J. Fleming. "Generalized Decomposition." In: *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*. 2013, pp. 428–442.

[GW13] Olga Goussevskaia and Roger Wattenhofer. "Scheduling with interference decoding: Complexity and algorithms." In: *Ad Hoc Networks* 11.6 (2013), pp. 1732–1745.

[HW13] Y. Hamadi and C.M. Wintersteiger. "Seven Challenges in Parallel SAT Solving." In: *AI Magazine* 34.2 (2013), pp. 99–106.

[Her+13a] Juan F. R. Herrera, Leocadio G. Casado, Eligius M. T. Hendrix, Remigijus Paulavicius, and Julius Zilinskas. "Dynamic and Hierarchical Load-Balancing Techniques Applied to Parallel Branch-and-Bound Methods." In: 8^{th} *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. 2013, pp. 497–502.

[Her+13b] Juan F.R. Herrera, Leocadio G. Casado, Remigijus Paulavicius, Julius ilinskas, and Eligius M.T. Hendrix. "On a Hybrid MPI-Pthread Approach for Simplicial Branch-and-Bound." In: *IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum* (2013), pp. 1764–1770.

[Hus+13] Hameed Hussain et al. "A survey on resource allocation in high performance distributed computing systems." In: *Parallel Computing* 39.11 (2013), pp. 709–736.

[IAN13] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. "A Study on the Specification of a Scalarizing Function in MOEA/D for Many-Objective Knapsack Problems." In: *LION7*. 2013, pp. 231–246.

[JH13] V. Janjic and K. Hammond. "How to be a Successful Thief." In: 19^{th} *Inter. Conf. on Parallel Proc. (Euro-Par)*. 2013, pp. 114–125.

[Jem+13] Imen Jemili, Dhouha Ghrab, Abdelfettah Belghith, Bilel Derbel, and Amine Dhraief. "Collision aware coloring algorithm for wireless sensor networks." In: *2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013, Sardinia, Italy, July 1-5, 2013*. 2013, pp. 1546–1553.

[KZB13] Liangjun Ke, Qingfu Zhang, and R. Battiti. "MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and Ant Colony." In: *IEEE Transactions on Cybernetics* 43.6 (2013), pp. 1845–1859.

[KJL13a] S. Kouki, M. Jemni, and T. Ladhari. "Scalable Distributed Branch and Bound for the Permutation Flow Shop Problem." In: 8^{th} *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. 2013, pp. 503–508.

[KJL13b] S. Kouki, M. Jemni, and T. Ladhari. "Scalable Distributed Branch and Bound for the Permutation Flow Shop Problem." In: *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*. 2013, pp. 503–508.

[MMT13] M. Mezmaz, N. Melab, and D. Tuyttens. "A Multithreaded branch-and-bound algorithm for solving the flow-shop problem on a multicore environment." In: *Large Scale Network-Centric Distributed Systems*. Ed. by H. Sarbazi-Azad and A. Y. Zomaya. John Wiley & Sons, Inc., Hoboken, New Jersey, 2013. Chap. 3.

[Sar+13] L. Sarzyniec, T. Buchert, E. Jeanvoine, and L. Nussbaum. "Design and Evaluation of a Virtual Experimental Environment for Distributed Systems." In: 21^{st} *Inter. Conf. on Parallel, Distributed and Network-Based Processing*. 2013.

[TF13] R. Tanabe and A. Fukunaga. "Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms." In: *Evolutionary Computation (CEC), 2013 IEEE Congress on*. 2013, pp. 1263–1270.

[TBD13] Thanh-Do Tran, Dimo Brockhoff, and Bilel Derbel. "Multiobjectivization with NSGA-ii on the noiseless BBOB testbed." In: *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*. 2013, pp. 1217–1224.

[VHS13] Nadarajen Veerapen, Youssef Hamadi, and Frédéric Saubion. "Using Local Search with adaptive operator selection to solve the Progressive Party Problem." In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*. 2013, pp. 554–561.

[Ver+13] Sébastien Verel, Arnaud Liefooghe, Laetitia Jourdan, and Clarisse Dhaenens. "On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives." In: *Eur J Oper Res* 227.2 (2013), pp. 331–342.

[VDM13] Trong-Tuan Vu, Bilel Derbel, and Nouredine Melab. "Adaptive Dynamic Load Balancing in Heterogeneous Multiple GPUs-CPUs Distributed Setting: Case Study of B&B Tree Search." In: 7^{th} *International Conference on Learning and Intelligent Optimization (LION)*. 2013, pp. 87–103.

[ZC13] Saúl Zapotecas Martínez and Carlos A. Coello Coello. "MOEA/D Assisted by Rbf Networks for Expensive Multi-objective Optimization Problems." In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. Amsterdam, The Netherlands: ACM, 2013, pp. 1405–1412.

[Ben12] A. Bendjoudi. *Scalable and fault tolerant hierarchical B&B algorithms for Computational Grids*. PhD Thesis. Algiers, Algeria, 2012.

[BMT12a] A. Bendjoudi, N. Melab, and E. -G. Talbi. "Hierarchical Branch and Bound Algorithm for Computational Grids." In: *Future Generation Computer Systems (FGCS)* 28.8 (2012), pp. 1168–1176.

[BMT12b] A. Bendjoudi, N. Melab, and E-G. Talbi. "An adaptive hierarchical master-worker framework for grids: Application to B&B algorithms." In: *Journal of Parallel and Distributed Computing (JPDC)* 72.2 (2012), pp. 120–131.

[Can+12] Caner Candan, Adrien Goeffon, Frédéric Lardeux, and Frédéric Saubion. "A dynamic island model for adaptive operator selection." In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM. 2012, pp. 1253–1260.

[CM12] I. Chakroun and M. Melab. "An Adaptative Multi-GPU based Branch-and-Bound. A Case Study: the Flow-Shop Scheduling Problem." In: 14^{th} *IEEE International Conference on High Performance Computing and Communications*. 2012.

[DD12] Houda Derbel and Bilel Derbel. "On neighborhood tree search." In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. 2012, pp. 1261–1268.

[Dro+12] Maciej Drozdowski, Pawel Marciniak, Grzegorz Pawlak, and Maciej Plaza. "Grid Branch-and-Bound for Permutation Flowshop." In: *Parallel Processing and Applied Mathematics*. Vol. 7204. LNCS. Springer, 2012, pp. 21–30.

[DT12] Madalina M. Drugan and Dirk Thierens. "Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies." In: *J. Heuristics* 18.5 (2012), pp. 727–766.

[GL12] Adrien Goëffon and Frédéric Lardeux. "Autonomous Local Search Algorithms with Island Representation." In: *Learning and Intelligent Optimization*. Ed. by Youssef Hamadi and Marc Schoenauer. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 390–395.

[HMS12] Youssef Hamadi, Eric Monfroy, and Frédéric Saubion, eds. *Autonomous Search*. Springer, 2012.

[KJL12] Samia Kouki, Mohamed Jemni, and Talel Ladhari. "A Load Balanced Distributed Algorithm to Solve the Permutation Flow Shop Problem Using the Grid." In: *Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering*. CSE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 146–153.

[Lie+12] Arnaud Liefooghe, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and El-Ghazali Talbi. "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems." In: *J. Heuristics* 18.2 (2012), pp. 317–352.

[Mel+12] N. Melab, I. Chakroun, M. Mezmaz, and D. Tuyttens. "A GPU-accelerated B&B Algorithm for the Flow-Shop Scheduling Problem." In: 14^{th} *IEEE Conf. on Cluster Computing*. 2012.

[Och12] Gabriela Ochoa. "Hyper-heuristics and Cross-domain Optimization." In: *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO '12. Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 1197–1214.

[OD12] Lars Otten and Rina Dechter. "Advances in Distributed Branch and Bound." In: *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*. 2012, pp. 917–918.

[Pil+12] L.L. Pilla et al. "A Hierarchical Approach for Load Balancing on Parallel Multicore Systems." In: 41^{st} *Inter. Conf. on Parallel Processing (ICPP)*. 2012, pp. 118–127.

[PDB12] Frédéric Pinel, Grégoire Danoy, and Pascal Bouvry. "Evolutionary Algorithm Parameter Tuning with Sensitivity Analysis." In: *Security and Intelligent Information Systems: International Joint Conferences, SIIS 2011, Warsaw, Poland, June 13-14, 2011, Revised Selected Papers*. Ed. by Pascal Bouvry, Mieczysław A. Kłopotek, Franck Leprévost, Małgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybiński. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 204–216.

[SD12] Abdorreza Savadi and Hossein Deldari. "A Bridging Model for Branch-and-Bound Algorithms on Multi-core Architectures." In: 5^{th} *International Symposium on Parallel Architectures, Algorithms and Programming*. PAAP. 2012, pp. 235–241.

[SL12] Kate Smith-Miles and Leo Lopes. "Measuring instance difficulty for combinatorial optimization problems." In: *Comput Oper Res* 39.5 (2012), pp. 875–889.

[VMS12] Nadarajen Veerapen, Jorge Maturana, and Frédéric Saubion. "An exploration-exploitation compromise-based adaptive operator selection for local search." In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. 2012, pp. 1277–1284.

[Vu+12] Trong-Tuan Vu, Bilel Derbel, Ali Asim, Ahcene Bendjoudi, and Nouredine Melab. "Overlay-Centric Load Balancing: Applications to UTS and B&B." In: 14^{th} *IEEE International Conference on Cluster Computing (CLUSTER)*. 2012, pp. 382–390.

[BDW11] Mihai Budiu, Daniel Delling, and Renato F. Werneck. "DryadOpt: Branch-and-Bound on Distributed Data-Parallel Execution Engines." In: *International Parallel and Distributed Processing Symposium (IPDPS)*. Washington, DC, USA, 2011, pp. 1278–1289.

[Bur+11] Edmund K. Burke, Michel Gendreau, Gabriela Ochoa, and James D. Walker. "Adaptive Iterated Local Search for Cross-domain Optimisation." In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2011, pp. 1987–1994.

[DV11] Bilel Derbel and Sébastien Verel. "DAMS: Distributed Adaptive Metaheuristic Selection." In: *Genetic And Evolutionary Computation Conference (GECCO)*. Dublin, Ireland: ACM, 2011, pp. 1955–1962.

[DDM11a] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Distributed B&B: A Pure Peer-to-Peer Approach." In: *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May 2011 - Workshop Proceedings*. 2011, pp. 1788–1797.

[DDM11b] Mathieu Djamaï, Bilel Derbel, and Nouredine Melab. "Impact of logical overlay upon a Pure P2P approach for the B&B algorithm." In: *IEEE Inter. Conf. on Systems and Computer Science (ICSCS'11)*. 2011.

[Dor+11] Bernabé Dorronsoro, Grégoire Danoy, Pascal Bouvry, and Antonio J. Nebro. "Multi-objective Cooperative Coevolutionary Evolutionary Algorithms for Continuous and Combinatorial Optimization." In: *Intelligent Decision Systems in Large-Scale Distributed Environments*. Ed. by Pascal Bouvry, Horacio González-Vélez, and Joanna Kołodziej. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 49–74.

[Dur+11] Juan Durillo, Qingfu Zhang, Antonio Nebro, and Enrique Alba. "Distribution of Computational Effort in Parallel MOEA/D." In: *International Conference on Learning and Intelligent Optimization (LION)*. LNCS. 2011.

[ECG11] J.F.S. Estrada, L. G. Casado, and I Garcia. "Adaptive Parallel Interval Global Optimization Algorithms Based on their Performance for Non-dedicated Multicore Architectures." In: *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 2011, pp. 252–256.

[Gei11] Martin Josef Geiger. "Decision support for multi-objective flow shop scheduling by the Pareto Iterated Local Search methodology." In: *Computers & Industrial Engineering* 61.3 (2011), pp. 805–812.

[GL11] A. Goeffon and F. Lardeux. "Optimal One-Max Strategy with Dynamic Island Models." In: *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*. 2011, pp. 485–488.

[GF11] Yiyuan Gong and Alex Fukunaga. "Distributed island-model genetic algorithms using heterogeneous parameter settings." In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*. 2011, pp. 820–827.

[Han11] N. Hansen. *Injecting External Solutions Into CMA-ES*. Tech. rep. INRIA, 2011.

[HHL11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration." In: *Learning and Intelligent OptimizatioN*. 2011, pp. 507–523.

[LGH11] Z. Lü, F. Glover, and J.-K. Hao. "Neighborhood Combination for Unconstrained Binary Quadratic Problems." In: *MIC Post-Conference Book*. 2011, pp. 49–61.

[LHG11] Z. Lü, J.-K. Hao, and F. Glover. "Neighborhood analysis: a case study on curriculum-based course timetabling." In: *J. of Heuristics* 17 (2 2011), pp. 97–118.

[MIY11] S-J Min, C. Iancu, and K. Yelick. "Hierarchical Work Stealing on Manycore Clusters." In: 5^{th} *Conference on Partitioned Global Address Space Prog. Models*. 2011.

[Ope11] OpenMP. *OpenMP Application Program Interface*. 2011.

[RLP11] Kaushik Ravichandran, Sangho Lee, and Santosh Pande. "Work stealing for multi-core HPC clusters." In: 17^{th} *inter. conf. on Parallel processing (Euro-Par)*. 2011, pp. 205–217.

[Sar+11] Vijay A. Saraswat, Prabhanjan Kambadur, Sreedhar Kodali, David Grove, and Sriram Krishnamoorthy. "Lifeline-based global load balancing." In: 16^{th} *ACM Symp. on Principles and practice of parallel programming (PPoPP '11)*. 2011, pp. 201–212.

[Zha+11] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen. "Cloud versus in-house cluster: Evaluating Amazon cluster compute instances for running MPI applications." In: *Inter. Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 2011, pp. 1–10.

[BB10] Lucio Barreto and Michael Bauer. "Parallel Branch and Bound Algorithm - A comparison between serial, OpenMP and MPI implementations." In: *Journal of Physic: Conference series* 256.1 (2010), p. 012018.

[BR10a] O. Beaumont and AL. Rosenberg. "Link-heterogeneity vs. node-heterogeneity in clusters." In: *High Performance Computing (HiPC), 2010 International Conference on*. 2010, pp. 1–8.

[BR10b] O. Beaumont and A.L. Rosenberg. "Link-heterogeneity vs. node-heterogeneity in clusters." In: *International Conference on High Performance Computing (HiPC)*. 2010, pp. 1–8.

[Bur+10] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. "A Classification of Hyper-heuristic Approaches." In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. Boston, MA: Springer US, 2010, pp. 449–468.

[DMZ10] Bilel Derbel, Mohamed Mosbah, and Akka Zemmari. "Sublinear Fully Distributed Partition with Applications." In: *Theory Comput. Syst.* 47.2 (2010), pp. 368–404.

[DT10a] Bilel Derbel and El-Ghazali Talbi. "Distributed Node Coloring in the SINR Model." In: *2010 International Conference on Distributed Computing Systems, ICDCS 2010, Genova, Italy, June 21-25, 2010*. 2010, pp. 708–717.

[DT10b] Bilel Derbel and El-Ghazali Talbi. "Radio Network Distributed Algorithms in the Unknown Neighborhood Model." In: *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3-6, 2010. Proceedings*. 2010, pp. 155–166.

[Fia+10a] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michele Sebag. "Analyzing Bandit-based Adaptive Operator Selection Mechanisms." In: *Annals of Mathematics and Artificial Intelligence – Special Issue on Learning and Intelligent Optimization* 60 (2010). Ed. by R. Battiti et al., pp. 25–64.

[Fia+10b] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michele Sebag. "Analyzing bandit-based adaptive operator selection mechanisms." In: *Annals of Mathematics and Artificial Intelligence* 60.1-2 (2010), pp. 25–64.

[Fig+10] J. R. Figueira, A. Liefooghe, E.-G. Talbi, and A. P. Wierzbicki. "A Parallel Multiple Reference Point Approach for Multi-objective Optimization." In: *European Journal of Operational Research* 205 (2010), pp. 390–400.

[GB10] N. Gast and G. Bruno. "A Mean Field Model of Work Stealing in Large-scale Systems." In: *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*. 2010, pp. 13–24.

[Han+10] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posík. "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009." In: *GECCO workshop on Black-Box Optimization Benchmarking (BBOB'2010)*. Ed. by J. Branke et al. ACM, 2010, pp. 1689–1696.

[Ish+10] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima. "Simultaneous Use of Different Scalarizing Functions in MOEA/D." In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. GECCO '10. Portland, Oregon, USA: ACM, 2010, pp. 519–526.

[Kad+10] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. "ISAC – Instance-Specific Algorithm Configuration." In: *European Conference on Artificial Intelligence*. 2010, pp. 751–756.

[KJL10] Samia Kouki, Mohamed Jemni, and Talel Ladhari. "Deployment of Solving Permutation Flow Shop Scheduling Problem on the Grid." English. In: *Grid and Distributed Computing, Control and Automation*. Vol. 121. Communications in Computer and Information Science. Springer, 2010, pp. 95–104.

[Kra10] Oliver Kramer. "Evolutionary self-adaptation: a survey of operators and strategy parameters." In: *Evolutionary Intelligence* 3.2 (2010), pp. 51–65.

[LD10] Ailsa H. Land and Alison G. Doig. "An Automatic Method for Solving Discrete Programming Problems." English. In: *50 Years of Integer Programming 1958-2008*. Springer Berlin Heidelberg, 2010, pp. 105–132.

[LMS10] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. "Iterated local search: Framework and applications." In: *Handbook of Metaheuristics*. Springer, 2010, pp. 363–397.

[LT10] T. Lust and J. Teghem. "Two-phase Pareto local search for the biobjective traveling salesman problem." In: *Journal of Heuristics* 16.3 (2010), pp. 475–510.

[Mos10] Sanaz Mostaghim. "Parallel Multi-objective Optimization Using Self-organized Heterogeneous Resources." In: *Parallel and Distributed Computational Intelligence*. Vol. 269. Studies in Computational Intelligence. Springer, 2010, pp. 165–179.

[ND10] Antonio J. Nebro and Juan José Durillo. "A Study of the Parallelization of the Multi-Objective Metaheuristic MOEA/D." In: *International Conference on Learning and Intelligent Optimization (LION)*. LNCS. 2010, pp. 303–317.

[OD10] Lars Otten and Rina Dechter. "Load Balancing for Parallel Branch and Bound." In: *10th Workshop on Preferences and Soft Constraints*. 2010, pp. 51–65.

[QW10a] Jean-Noël Quintin and Frédéric Wagner. "Hierarchical work-stealing." In: 16^{th} *inter. conference on Parallel processing (Euro-Par)*. 2010, pp. 217–229.

[QW10b] J.-N. Quintin and F. Wagner. "Hierarchical Work-stealing." In: 16^{th} *Inter. Conf. on Parallel Processing (EuroPar)*. 2010, pp. 217–229.

[XHL10] Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. "Hydra: Automatically Configuring Algorithms for Portfolio-based Selection." In: *Conference on Artificial Intelligence*. 2010, pp. 210–216.

[Zha+10] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. "Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model." In: *IEEE Transactions on Evolutionary Computation* 14.3 (2010), pp. 456–474.

[Bia+09] Marco Biazzini, Balazs Banhelyi, Alberto Montresor, and Mark Jelasity. "Distributed Hyper-heuristics for Real Parameter Optimization." In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1339–1346.

[Bir09] Mauro Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer, 2009.

[BAE09] Lam T. Bui, Hussein A. Abbass, and Daryl Essam. "Local models — an approach to distributed multi-objective optimization." In: *Computational Optimization and Applications* 42.1 (2009), pp. 105–139.

[Der+09] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. "Local Computation of Nearly Additive Spanners." In: *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*. 2009, pp. 176–190.

[Din+09] James Dinan, D. Brian Larkins, P. Sadayappan, Sriram Krishnamoorthy, and Jarek Nieplocha. "Scalable Work Stealing." In: *ACM Conference on High Performance Computing Networking, Storage and Analysis (SC)*. Portland, Oregon, 2009, 53:1–53:11.

[EPS09] Yuri Evtushenko, Mikhail Posypkin, and Israel Sigal. "A framework for parallel large-scale global optimization." English. In: *Computer Science - Research and Development* 23.3-4 (2009), pp. 211–215.

[Fia+09] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michele Sebag. "Dynamic Multi-Armed Bandits and Extreme Value-based Rewards for Adaptive Operator Selection in Evolutionary Algorithms." In: *LION'09*. Ed. by T. Stuetzle et al. Vol. 5851. LNCS. Springer Verlag, 2009, pp. 176–190.

[Hut+09] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. "ParamILS: An Automatic Algorithm Configuration Framework." In: *J. Artif. Int. Res.* 36.1 (2009), pp. 267–306.

[Ish+09] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima. "Adaptation of Scalarizing Functions in MOEA/D: An Adaptive Scalarizing Function-Based Multiobjective Evolutionary Algorithm." In: *Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings*. Ed. by Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 438–452.

[LZ09] Hui Li and Qingfu Zhang. "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II." In: *IEEE Trans. Evolutionary Computation* 13.2 (2009), pp. 284–302.

[Lie+09] Arnaud Liefooghe, Salma Mesmoudi, Jérémie Humeau, Laetitia Jourdan, and El-Ghazali Talbi. "A Study on Dominance-Based Local Search Approaches for Multiobjective Combinatorial Optimization." In: *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics: Second International Workshop, SLS 2009, Brussels, Belgium, September 3-4, 2009. Proceedings*. Ed. by Thomas Stützle, Mauro Birattari, and Holger H. Hoos. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 120–124.

[Mat+09] Jorge Maturana, Álvaro Fialho, Frédéric Saubion, Marc Schoenauer, and Michele Sebag. "Extreme compass and dynamic multi-armed bandits for adaptive operator selection." In: *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE. 2009, pp. 365–372.

[Meh+09] Malika Mehdi, Mohand-Said Mezmaz, Nouredine Melab, El-Ghazali Talbi, and Pascal Bouvry. "P2P computing for large tree exploration-based exact optimisation." In: *IJGUC* 1.3 (2009), pp. 252–260.

[PS09] L. Paquete and T. Stützle. "Design and analysis of stochastic local search for the multiobjective traveling salesman problem." In: *Computers & Operations Research* 36.9 (2009), pp. 2619–2631.

[Smi09] Kate A. Smith-Miles. "Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection." In: *ACM Comput. Surv.* 41.1 (2009), 6:1–6:25.

[Tal09] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons, 2009.

[ZLL09] Q. Zhang, W. Liu, and H. Li. "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances." In: *2009 IEEE Congress on Evolutionary Computation*. 2009, pp. 203–208.

[Cas+08] L. G. Casado, J. A. Martinez, I. Garcia, and E. M. T. Hendrix. "Branch-and-Bound Interval Global Optimization on Shared Memory Multiprocessors." In: *Optimization Methods Software* 23.5 (2008), pp. 689–701.

[CC08] K. Chakhlevitch and P. I. Cowling. "Hyperheuristics: Recent Developments." In: *Adaptive and Multilevel Metaheuristics*. 2008, pp. 3–29.

[Cha+08] Pei Chann Chang, Shih Hsin Chen, Qingfu Zhang, and Jun Lin Lin. "MOEA/D for flowshop scheduling problems." In: *CEC*. 2008, pp. 1433–1438.

[DaC+08] Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michèle Sebag. "Adaptive operator selection with dynamic multi-armed bandits." In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. ACM Press, 2008, p. 913.

[Der08] Bilel Derbel. "Local Maps: New Insights into Mobile Agent Algorithms." In: *Distributed Computing, 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings*. 2008, pp. 121–136.

[DGo8] Bilel Derbel and Cyril Gavoille. "Fast deterministic distributed algorithms for sparse spanners." In: *Theor. Comput. Sci.* 399.1-2 (2008), pp. 83–100.

[Der+08] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. "On the locality of distributed sparse spanner construction." In: *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*. 2008, pp. 273–282.

[DMGo8] Bilel Derbel, Mohamed Mosbah, and Stefan Gruner. "Mobile Agents Implementing Local Computations in Graphs." In: *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*. 2008, pp. 99–114.

[Dur+08] Juan José Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. "A study of master-slave approaches to parallelize NSGA-II." In: *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. 2008, pp. 1–8.

[GRHo8] A. Goeffon, J.-M. Richer, and J.-K. Hao. "Progressive Tree Neighborhood Applied to the Maximum Parsimony Problem." In: *IEEE/ACM Trans. Comput. Biol. Bio.* 5 (2008), pp. 136–145.

[ITNo8] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. "Evolutionary many-objective optimization: A short review." In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. 2008, pp. 2419–2426.

[Meh+08] Malika Mehdi, Mohand-Said Mezmaz, Nouredine Melab, El-Ghazali Talbi, and Pascal Bouvry. "An Efficient Hybrid P2P Approach for Non-redundant Tree Exploration in B&B Algorithms." In: *Second International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2008), March 4th-7th, 2008, Technical University of Catalonia, Barcelona, Spain*. 2008, pp. 360–365.

[OPo8] Stephen Olivier and Jan Prins. "Scalable Dynamic Load Balancing Using UPC." In: 37^{th} *International Conference on Parallel Processing (ICPP)*. 2008, pp. 123–131.

[ÖBKo8] E. Özcan, B. Bilgin, and E. E. Korkmaz. "A comprehensive analysis of hyper-heuristics." In: *Intell. Data Anal.* 12 (2008), pp. 3–23.

[PRo8] J. Puchinger and G. Raidl. "Bringing order into the neighborhoods: relaxation guided variable neighborhood search." In: *J. of Heuristics* 14 (5 2008), pp. 457–472.

[Tal+08] El-Ghazali Talbi, Sanaz Mostaghim, Tatsuya Okabe, Hisao Ishibuchi, Günter Rudolph, and Carlos A. Coello Coello. "Parallel Approaches for Multiobjective Optimization." In: *Multiobjective Optimization – Interactive and Evolutionary Approaches*. Vol. 5252. LNCS. 2008, pp. 349–372.

[Xu+08] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "SATzilla: Portfolio-based Algorithm Selection for SAT." In: *J. Artif. Intell. Res. (JAIR)* 32 (2008), pp. 565–606.

[ZKT08] E. Zitzler, J. Knowles, and L. Thiele. "Quality Assessment of Pareto Set Approximations." In: *Multiobjective optimization – interactive and evolutionary approaches*. Vol. 5252. Lecture Notes in Computer Science. Springer, 2008. Chap. 14, pp. 373–404.

[AT07] H. E. Aguirre and Kiyoshi Tanaka. "Working principles, behavior, and performance of MOEAs on MNK-landscapes." In: *European Journal of Operational Research* 181.3 (2007), pp. 1670–1690.

[BNE07a] N. Beume, B. Naujoks, and M. Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume." In: *EJOR* 181.3 (2007), pp. 1653–1669.

[BNE07b] Nicola Beume, Boris Naujoks, and Michael Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume." In: *European Journal of Operational Research* 181.3 (2007), pp. 1653–1669.

[CLV07] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Second. Springer, 2007.

[Dic07] A. Diconstanzo. *Branch-and-Bound with peer-to-peer for large scale grids*. PhD Thesis. France: Ecole doctorale STIC, 2007.

[Din+07] J. Dinan, S. Olivier, G. Sabin, J. Prins, P. Sadayappan, and C.-W. Tseng. "Dynamic Load Balancing of Unbalanced Computations Using Message Passing." In: 21^{th} *IPDPS*. 2007, pp. 1–8.

[Eib+07] A. E. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and J. E. Smith. "Parameter Control in Evolutionary Algorithms." In: *Parameter Setting in Evolutionary Algorithms*. Springer, 2007, pp. 19–46.

[HYM07] Tomoyuki Hiroyasu, Kengo Yoshii, and Mitsunori Miki. "Discussion of parallel model of multi-objective genetic algorithms on heterogeneous computational resources." In: *ACM Genetic and Evolutionary Computation Conference*. 2007, pp. 904–904.

[IHR07] C. Igel, N. Hansen, and S. Roth. "Covariance Matrix Adaptation for Multi-objective Optimization." In: *Evolutionary Computation* 15.1 (2007), pp. 1–28.

[ISH07] Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. "Steady-State Selection and Efficient Covariance Matrix Update in the Multi-objective CMA-ES." English. In: *Evolutionary Multi-Criterion Optimization*. Springer Berlin Heidelberg, 2007, pp. 171–185.

[LLM07] Fernando G.. Lobo, Cludio F. Lima, and Zbigniew Michalewicz. *Parameter Setting in Evolutionary Algorithms*. 1st. Springer Publishing Company, Incorporated, 2007.

[Mez07] M. Mezmaz. *Une approche efficace pour le passage sur grilles de calcul de methodes dóptimisation combinatoire*. PhD Thesis. Lille, France, 2007.

[MMT07a] M. Mezmaz, N. Melab, and E-G. Talbi. "A Grid-based Parallel Approach of the Multi-Objective Branch and Bound." In: *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. PDP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 23–30.

[MMT07b] M.-S. Mezmaz, N. Melab, and E.-G. Talbi. "A Grid-enabled Branch and Bound Algorithm for Solving Challenging Combinatorial Optimization Problems." In: *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2007, pp. 1–9.

[MBS07] Sanaz Mostaghim, Juergen Branke, and Hartmut Schmeck. "Multi-objective particle swarm optimization on computer grids." In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2007, pp. 869–875.

[Oli+07] Stephen Olivier, Jun Huan, Jinze Liu, Jan Prins, James Dinan, P. Sadayappan, and Chau-Wen Tseng. "UTS: an unbalanced tree search benchmark." In: 19^{th} *inter. conf. on Languages and compilers for parallel computing (LCPC)*. 2007, pp. 235–250.

[PSS07] Luis Paquete, Tommaso Schiavinotto, and Thomas Stützle. "On local optima in multiobjective combinatorial optimization problems." In: *Annals of Operations Research* 156.1 (2007), p. 83.

[Thi07] Dirk Thierens. "Adaptive Strategies for Operator Allocation." In: *Param. Setting in EA*. Vol. 54. Springer, 2007, pp. 77–90.

[ZL07] Q. Zhang and H. Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731.

[AL06] Belarmino Adenso-Diaz and Manuel Laguna. "Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search." In: *Oper. Res.* 54.1 (2006), pp. 99–114.

[Bar06] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation*. Springer, 2006.

[Dre+06] Johann Dreo, Alain Petrowski, Patrick Siarry, and Eric Taillard. *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media, 2006.

[GS06] L. Gaspero and A. Schaerf. "Neighborhood Portfolio Approach for Local Search Applied to Timetabling Problems." In: *J. of Mathematical Modelling and Algorithms* 5 (2006), pp. 65–89.

[GK06] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media, 2006.

[HR06] B. Hu and G. Raidl. "Variable neighborhood descent with self-adaptive neighborhood-ordering." In: 7^{th} *EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Meta.* 2006.

[Jon06] Kenneth A. De Jong. *Evolutionary computation: A Unifed Approach*. MIT Press, 2006.

[KTZ06] J. Knowles, L. Thiele, and E. Zitzler. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. TIK Report 214. ETH Zurich, 2006.

[MTC06] Nouredine Melab, El-Ghazali Talbi, and Sébastien Cahon. "On Parallel Evolutionary Algorithms on the Computational Grid." In: *Parallel Evolutionary Computations*. Vol. 22. Studies in Computational Intelligence. Springer, 2006, pp. 117–132.

[SSP06] Francis Sourd, Olivier Spanjaard, and Patrice Perny. "Multi-objective branch and bound. Application to the bi-objective spanning tree problem." In: *7th International Conference in Multi-Objective Programming and Goal Programming*. Tours, France, 2006.

[TYG06] K.C. Tan, Y. J. Yang, and C.K. Goh. "A distributed Cooperative coevolutionary algorithm for multiobjective optimization." In: *IEEE Transactions on Evolutionary Computation* 10.5 (2006), pp. 527–549.

[AO05] K. Aida and T. Osumi. "A case study in running a parallel branch and bound application on the grid." In: *Symposium on Applications and the Internet*. 2005, pp. 164–173.

[Alb05] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, 2005.

[Cap+05] F. Cappello, P. Fraigniaud, B. Mans, and A.L. Rosenberg. "An algorithmic model for heterogeneous hyper-clusters: rationale and experience." In: *Int. J. Found. Comput. Sci.* 16.2 (2005), pp. 195–215.

[Ehr05] M. Ehrgott. *Multicriteria optimization*. Second. Springer, 2005.

[RGK05] Prapa Rattadilok, Andy Gaw, and Raymond S. K. Kwan. "Distributed Choice Function Hyper-heuristics for Timetabling and Scheduling." In: *Practice and Theory of Automated Timetabling V: 5th International Conference, PATAT 2004, Pittsburgh, PA, USA, August 18-20, 2004, Revised Selected Papers*. Ed. by Edmund Burke and Michael Trick. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 51–67.

[SUZ05] F. Streichert, H. Ulmer, and A. Zell. "Parallelization of Multi-Objective Evolutionary Algorithms using Clustering Algorithms." In: *Int. Conf. on Evo. Multi-Criterion Optimization (EMO)*. LNCS. 2005, pp. 92–107.

[Thi05] Dirk Thierens. "An adaptive pursuit strategy for allocating operator probabilities." In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM. 2005, pp. 1539–1546.

[Tom05] Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[Xu+05] Y. Xu, T. K. Ralphs, L. Ladányi, and M. J. Saltzman. "ALPS: A framework for implementing parallel search algorithms." In: 9^{th} *INFORMS Computing Society Conference*. 2005, pp. 319–334.

[Bra+04] J. Branke, H. Schmeck, K. Deb, and M. Reddy. "Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation." In: *IEEE Congress on Evolutionary Computation (CEC)*. 2004, pp. 1952–1957.

[CS04] Carlos A. Coello Coello and Margarita Reyes Sierra. "A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm." In: *Mexican Int. Conf. on Artificial Intelligence (MICAI)*. LNCS. 2004, pp. 688–697.

[HS04] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

[LTZ04] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. "Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem." In: *Nat Comput* 3.1 (2004), pp. 37–51.

[Mer04] Peter Merz. "Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms." In: *Evolutionary Computation* 12.3 (2004), pp. 303–325.

[PCS04] L. Paquete, M. Chiarandini, and T. Stützle. "Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study." In: *Metaheuristics for Multiobjective Optimisation*. Vol. 535. Lecture Notes in Economics and Mathematical Systems. Springer, 2004. Chap. 7, pp. 177–199.

[Van+04] R. Van Nieuwpoort, J. Maassen, G. Wrzesinska, T. Kielmann, and H. E. Bal. "Adaptive Load-Balancing for Divide-and-Conquer Grid Applications." In: *Journal of Supercomputing* (2004).

[ZK04] Eckart Zitzler and Simon Künzli. "Indicator-based selection in multiobjective search." In: *Parallel Problem Solving from Nature-PPSN VIII*. Springer. 2004, pp. 832–842.

[DZJ03] Kalyanmoy Deb, Pawan Zope, and Abhishek Jain. "Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms." In: *Int. Conf. on Evo. Multi-Criterion Optimization (EMO)*. LNCS. 2003, pp. 534–549.

[Hug03] E. J. Hughes. "Multiple single objective Pareto sampling." In: *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*. Vol. 4. 2003, 2678–2684 Vol.4.

[VZL03] David A. van Veldhuizen, Jesse B. Zydallis, and Gary B. Lamont. "Considerations in engineering parallel multiobjective evolutionary algorithms." In: *IEEE Trans. Evolutionary Computation* 7.2 (2003), pp. 144–173.

[VCC03] Sébastien Verel, Philippe Collard, and Manuel Clergue. "Where are Bottlenecks in NK Fitness Landscapes?" In: *CoRR* abs/0707.0641 (2003).

[Zit+03] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. "Performance Assessment of Multiobjective Optimizers: An Analysis and Review." In: *IEEE Transactions on Evolutionary Computation* 7.2 (2003), pp. 117–132.

[AF02] K. Aida and Y. Futakata. "High-performance parallel and distributed computing for the BMI eigenvalue problem." In: *International Parallel and Distributed Processing Symposium (IPDPS)*. 2002, pp. 71–78.

[AT02] E. Alba and M. Tomassini. "Parallelism and evolutionary algorithms." In: *IEEE Transactions on Evolutionary Computation* 6.5 (2002), pp. 443–462.

[ACF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." In: *Machine learning* 47.2-3 (2002), pp. 235–256.

[Bir+02] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. "A Racing Algorithm for Configuring Metaheuristics." In: *Genetic and Evolutionary Computation Conference*. 2002, pp. 11–18.

[Deb+02] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[Jas02] A. Jaszkiewicz. "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment." In: *IEEE Transactions on Evolutionary Computation* 6.4 (2002), pp. 402–412.

[Kie+02] T. Kielmann, H.E. Bal, J. Maassen, R. Van Nieuwpoort, L. Eyraud, R. Hofman, and K. Verstoep. "Programming environments for high-performance Grid computing: the Albatross project." In: *Future Generation Computer Systems* 18.8 (2002), pp. 1113–1125.

[TC02] Shisanu Tongchim and Prabhas Chongstitvatana. "Parallel genetic algorithm with parameter adaptation." In: *Information Processing Letters* 82.1 (2002). Evolutionary Computation, pp. 47–54.

[ZL02] Zhong-Yao Zhu and Kwong-Sak Leung. "Asynchronous self-adjustable island genetic algorithm for multi-objective optimization problems." In: *IEEE World on Congress on Computational Intelligence (WCCI 2002)* (2002), pp. 837–842.

[Bre01] Leo Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32.

[CF01] Qun Chen and Michael C. Ferris. "FATCOP: A Fault Tolerant Condor-PVM Mixed Integer Programming Solver." In: *SIAM Journal on Optimization* 11.4 (2001), pp. 1019–1036.

[CKS01] P. I. Cowling, G. Kendall, and E. Soubeiga. "A Hyperheuristic Approach to Scheduling a Sales Summit." In: *3th Int. Conf. on Pract. and Th. of Auto. Timetabling*. 2001, pp. 176–190.

[Deb01] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001.

[EPH01] Jonathan Eckstein, Cynthia A. Phillips, and William E. Hart. *PICO: An Object-Oriented Framework for Parallel Branch and Bound*. Technical Report. Rutgers University, 2001.

[HO01] N. Hansen and A. Ostermeier. "Completely Derandomized Self-Adaptation in Evolution Strategies." In: *Evolutionary Computation* 9.2 (2001), pp. 159–195.

[LL01] Pedro Larraanaga and Jose A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[MIG01] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. "Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms." In: *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001 Zurich, Switzerland, March 7–9, 2001 Proceedings*. Ed. by Eckart Zitzler, Lothar Thiele, Kalyanmoy Deb, Carlos Artemio Coello Coello, and David Corne. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 82–95.

[VKB01] R. Van Nieuwpoort, T. Kielmann, and H.E. Bal. "Efficient Load Balancing for Wide-area Divide-and-conquer Applications." In: 8^{th} *Symp. on Principles and Practices of Para. Programming*. 2001, pp. 34–43.

[ABB00] U.A. Acar, G.E. Blelloch, and R.D. Blumofe. "The Data Locality of Work Stealing." In: 15^{th} *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 2000, pp. 1–12.

[BCG00] Benoît Bourbeau, Teodor Gabriel Crainic, and Bernard Gendron. "Branch-and-bound Parallelization Strategies Applied to a Depot Location and Container Fleet Management Problem." In: *Parallel Computing* 26.1 (2000), pp. 27–46.

[Deb+00] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II." In: *Conference on Parallel Problem Solving from Nature (PPSN VI)*. Ed. by M. Schoenauer et al. Vol. 1917. LNCS. Springer, 2000, pp. 849–858.

[Gou+00] J.-P. Goux, S. Kulkarni, J. Linderoth, and M. Yoder. "An enabling framework for master-worker applications on the Computational Grid." In: *The* 9^{th} *International Symposium on High-Performance Distributed Computing*. 2000, pp. 43–50.

[GLY00] J.-P. Goux, Jeff Linderoth, and Michael Yoder. "Metacomputing and the Master-Worker Paradigm." In: *Preprint MCS/ANL-P792-0200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne*. 2000.

[IF00] A Iamnitchi and I Foster. "A problem-specific fault-tolerance mechanism for asynchronous, distributed systems." In: *Parallel Processing, 2000. Proceedings. 2000 International Conference on*. 2000, pp. 4–13.

[Kal00] Ignacy Kaliszewski. "Using trade-off information in decision-making algorithms." In: *Computers & Operations Research* 27.2 (2000), pp. 161–182.

[MIG00a] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. "Cellular Genetic Local Search for Multi-Objective Optimization." In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*. 2000, pp. 307–314.

[MIG00b] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. "Cellular Genetic Local Search for Multi-objective Optimization." In: *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*. GECCO'00. Las Vegas, Nevada: Morgan Kaufmann Publishers Inc., 2000, pp. 307–314.

[Nea+00] MichaelO. Neary, Alan Phipps, Steven Richman, and Peter Cappello. "Javelin 2.0: Java-Based Parallel Computing on the Internet." English. In: *Euro-Par 2000 Parallel Processing*. Vol. 1900. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 1231–1238.

[Pel00a] David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[Peloob] David Peleg. "Proximity-preserving labeling schemes." In: *Journal of Graph Theory* 33.3 (2000), pp. 167–176.

[BL99] R. D. Blumofe and C. E. Leiserson. "Scheduling multithreaded computations by work stealing." In: *J. ACM* 46 (5 1999), pp. 720–748.

[GK99] A. Grama and V. Kumar. "State of the art in parallel search techniques for discrete optimization problems." In: *IEEE Transactions on Knowledge and Data Engineering* 11.1 (1999), pp. 28–35.

[Mie99] K. Miettinen. *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer, 1999.

[PBH99] A. Plaat, H.E. Bal, and R. F H Hofman. "Sensitivity of parallel applications to large differences in bandwidth and latency in two-layer interconnects." In: *HPCA*. 1999, pp. 244–253.

[Can98] Erick Cantu-Paz. "A Survey of Parallel Genetic Algorithms." In: *CALCULATEURS PARALLELES, RESEAUX ET SYSTEMS REPARTIS* 10 (1998).

[DD98] Indraneel Das and John E. Dennis. "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems." In: *SIAM Journal on Optimization* 8.3 (1998), pp. 631–657.

[FLR98a] Matteo F, Charles E. L, and Keith H. R. "The implementation of the Cilk-5 multithreaded language." In: *SIGPLAN Not.* 33 (5 1998), pp. 212–223.

[FLR98b] Matteo F, Charles E. L, and Keith H. R. "The implementation of the Cilk-5 multithreaded language." In: *SIGPLAN Not.* 33 (5 1998), pp. 212–223.

[IM98] H. Ishibuchi and T. Murata. "A multi-objective genetic local search algorithm and its application to flowshop scheduling." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.3 (1998), pp. 392–403.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions." In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[Vis+98] M Visée, Jacques Teghem, Marc Pirlot, and Ekunda L. Ulungu. "Two-phases Method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem." In: *Journal of Global Optimization* 12.2 (1998), pp. 139–155.

[ZT98] Eckart Zitzler and Lothar Thiele. "Multiobjective optimization using evolutionary algorithms — A comparative case study." English. In: *Parallel Problem Solving from Nature (PPSN V)*. Vol. 1498. Lecture Notes in Computer Science. Springer, 1998, pp. 292–301.

[MH97] Nenad Mladenovic and Pierre Hansen. "Variable neighborhood search." In: *Computers & OR* 24.11 (1997), pp. 1097–1100.

[WM97] David H Wolpert and William G Macready. "No free lunch theorems for optimization." In: *Evolutionary Computation, IEEE Transactions on* 1.1 (1997), pp. 67–82.

[XL97] Chengzhong Xu and Francis C. Lau. *Load Balancing in Parallel Computers: Theory and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[BBB96] J.E. Baldeschwieler, R.D. Blumofe, and E.A. Brewer. "ATLAS: an infrastructure for global computing." In: 7^{th} *ACM SIGOPS Workshop on Systems support for worldwide applications*. 1996, pp. 165–172.

[KE95] J. Kennedy and R. Eberhart. "Particle swarm optimization." In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. 1995, pp. 1942–1948.

[TLM95] S. Tschoke, R. Lubling, and B. Monien. "Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network." In: *Parallel Processing Symposium, 1995. Proceedings., 9th International*. 1995, pp. 182–189.

[FMM94] R. Feldmann, P. Mysliwiete, and B. Monien. "Studying Overheads in Massively Parallel MIN/MAX-tree Evaluation." In: 6^{th} *ACM Symposium on Parallel Algorithms and Architectures*. Cape May, New Jersey, USA, 1994, pp. 94–103.

[GC94] B. Gendron and T.G. Crainic. "Parallel Branch-and-Bound Algorithms: Survey and Synthesis." In: *Operations Research* 42 (6 1994), pp. 1042–1066.

[KGV94] Vipin Kumar, Ananth Y. Grama, and Nageshwara Rao Vempaty. "Scalable Load Balancing Techniques for Parallel Computers." In: *J. Parallel Distrib. Comput.* 22.1 (1994), pp. 60–79.

[Kum+94] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing*. Benjamin/Cummings, 1994.

[Alt+93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah A. Joseph, and José Soares. "On Sparse Spanners of Weighted Graphs." In: *Discrete & Computational Geometry* 9.1 (1993), pp. 81–100.

[JPS93] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. "Lipschitzian optimization without the Lipschitz constant." In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181.

[Kau93] S. A. Kauffman. *The Origins of Order*. New York: Oxford University Press, 1993.

[Tai93] E. Taillard. "Benchmarks for basic scheduling problems." In: *European Journal of Operational Research* 64.2 (1993), pp. 278–285.

[WR93] M. H. Willebeek-LeMair and A. P. Reeves. "Strategies for dynamic load balancing on highly parallel computers." In: *IEEE Transactions on Parallel and Distributed Systems* 4.9 (1993), pp. 979–993.

[Dor92] Marco Dorigo. "Optimization, Learning and Natural Algorithms." PhD thesis. Italy: Politecnico di Milano, 1992.

[Lin92] Nathan Linial. "Locality in Distributed Graph Algorithms." In: *SIAM J. Comput.* 21.1 (1992), pp. 193–201.

[LM92] R. Luling and B. Monien. "Load balancing for distributed branch amp; bound algorithms." In: *Parallel Processing Symposium, 1992. Proceedings., Sixth International*. 1992, pp. 543–548.

[SKS92] Niranjan G. Shivaratri, Phillip Krueger, and Mukesh Singhal. "Load Distributing for Locally Distributed Systems." In: *Computer* 25.12 (1992), pp. 33–44.

[TB92] H.W.J.M. Trienekens and A. Bruin. *Towards a Taxonomy of Parallel Branch and Bound Algorithms*. Technical Report. Rotterdam, Netherlands: Econometric Institute, Erasmus University, 1992.

[Koz90] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science, 1990.

[CJ89] Dah-Ming Chiu and Raj Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks." In: *Comput. Netw. ISDN Syst.* 17.1 (1989), pp. 1–14.

[Cyb89] G. Cybenko. "Dynamic load balancing for distributed memory multiprocessors." In: *J. Parallel Distrib. Comput.* 7 (2 1989), pp. 279–301.

[Gol89] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. 2. Addison-Wesley, Reading, MA, 1989.

[JS89] J.M. Jansen and F.W. Sijstermans. "Parallel branch-and-bound algorithms." In: *Future Generation Computer Systems* 4.4 (1989), pp. 271–279.

[PS89] D. Peleg and A.A. Schäffer. "Graph spanners." In: *Journal of Graph Theory* 13.1 (1989), pp. 99–116.

[Dij87] E. W. Dijkstra. "Derivation of a termination detection algorithm for distributed computations." In: *Control Flow and Data Flow: concepts of distributed programming* (1987), pp. 507–512.

[FM87] Raphael Finkel and Udi Manber. "DIB - a distributed implementation of backtracking." In: *ACM Trans. Program. Lang. Syst.* 9 (2 1987), pp. 235–256.

[KJ87] Michael N. Katehakis and Arthur F. Veinott Jr. "The Multi-Armed Bandit Problem: Decomposition and Computation." In: *Math. Oper. Res.* 12.2 (1987), pp. 262–268.

[ELZ86] D.L. Eager, E.D. Lazowska, and J. Zahorjan. "Adaptive load sharing in homogeneous distributed systems." In: *IEEE Transactions on Software Engineering,* SE-12.5 (1986), pp. 662–675.

[Glo86] Fred Glover. "Future paths for integer programming and links to artificial intelligence." In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[Gre86] John J Grefenstette. "Optimization of control parameters for genetic algorithms." In: *Systems, Man and Cybernetics, IEEE Transactions on* 16.1 (1986), pp. 122–128.

[Tri86] H.W.J.M. Trienekens. *Parallel Branch and Bound on an MIMD System*. Technical Report. Rotterdam, Netherlands: Econometric Institute, Erasmus University, 1986.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing." In: *Science* 220.4598 (1983), pp. 671–680.

[ES82] Paul Erdös and M Simonovits. "Compactness results in extremal graph theory." In: *Combinatorica* 2.3 (1982), pp. 275–288.

[BS81] F. Warren Burton and M. Ronan Sleep. "Executing Functional Programs on a Virtual Tree of Processors." In: *Proceedings of the 1981 Conference on Functional Programming Languages and Computer Architecture*. FPCA '81. Portsmouth, New Hampshire, USA: ACM, 1981, pp. 187–194.

[AN79] Y. P. Aneja and K. P. K. Nair. "Bicriteria Transportation Problem." In: *Management Science* 25.1 (1979).

[Ric76] John R. Rice. "The Algorithm Selection Problem." In: *Advances in Computers* 15 (1976), pp. 65–118.

[Hol75] John H Holland. "Adaptation in natural and artificial system." In: *Ann Arbor, University of Michigan Press* (1975).

[Rec73] Ingo Rechenberg. "Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologishen Evolution." PhD thesis. Technical University of Berlin, 1973.

[Erd64] Paul Erdös. "Extremal Problems in Graph Theory." In: *Publ. House Cszechoslovak Acad. Sci., Prague*. 1964, pp. 29–36.