N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

Docteur de Lille 1

Spécialité : **Informatique**

préparée au laboratoire **CRIStAL**

dans le cadre de l'École Doctorale **EDSPI**

présentée et soutenue publiquement
par

Sophie Jacquin

le 19 novembre 2015

Titre:

Hybridation des métaheuristiques et de la programmation dynamique pour les problèmes d'optimisation mono et multi-objectif : Application à la production d'énergie.

Directeur de thèse: **El-Ghazali Talbi**
Co-directeur de thèse: **Laetitia Jourdan**

Jury

M. Gilles Goncalves,	Président du jury
M. Frédéric Saubion,	Rapporteur
M. Patrice Perny,	Rapporteur
M. Lucie Galand,	Examinateur
M. Alexandre Caminada,	Examinateur
M. El-Ghazali Talbi,	Directeur de thèse
M. Laetitia Jourdan,	Co-directeur de thèse

Résumé

Cette thèse s'intéresse à l'étude de deux problèmes d'optimisation pour la production d'énergie électrique. Le premier est un problème académique très étudié : le Unit Commitment Problem (UCP). Le second est un problème de planification des débits d'eau dans un réseau hydro-électrique issu d'une application industrielle. Ces deux problèmes sont des problèmes NP-complets très difficiles car ils sont non linéaires, fortement contraints et que la taille des données est importante. Dans la première partie de cette thèse, nous proposons DYNAMOP. Il s'agit d'un algorithme génétique qui guide la recherche effectuée par la programmation dynamique en manipulant des solutions représentées sous forme de chemins du graphe d'états. Cette représentation est avantageuse car, d'une part, elle facilite la mise en place d'hybridations avec la programmation dynamique et, d'autre part, elle permet de proposer des opérateurs évolutionnaires efficaces tenant compte les dépendances entre les variables. DYNAMOP est appliqué aux deux problèmes de production d'énergie. La qualité des résultats obtenus permet d'affirmer que cette méthode est bien adaptée à la résolution de ce type de problèmes.

Dans la seconde partie, nous présentons MO-DYNAMOP une extension de DYNAMOP à l'optimisation multi-objectif. MO-DYNAMOP est évalué sur une version bi-objectif de l'UCP nécessitant l'utilisation d'une représentation indirecte. Une solution partielle sera ainsi décodée en un ensemble de solutions complètes Pareto équivalentes ce qui rend difficile l'évaluation de sa qualité. Nous proposons donc plusieurs adaptations des stratégies usuelles d'assignation de fitness et comparons les méthodes obtenues à la littérature.

Mots-clé :

métaheuristique, algorithme évolutionnaire, méthode hybride, programmation dynamique, optimisation multi-objectif, représentation indirecte

Hybridization between metaheuristic and dynamic programming for mono and multi-objective optimization problems : Application in energy production.

Abstract

In this thesis, two energy production problems are studied. The first problem is called the Unit Commitment Problem (UCP), which is extensively studied in the literature. The second one is a hydro scheduling problem with a real life application. They are both very hard (NP-complete) problems since they are non-linear and highly constrained, besides the data size is large.

In the first part of this thesis we propose a hybrid method called DYNAMOP (DYNAmic programming using Metaheuristic for Optimization Problems). DYNAMOP uses a representation based on a path in the graph of dynamic programming states. It adapted to the dynamic structure of the problem and facilitates the hybridization between evolutionary algorithms and dynamic programming. DYNAMOP is tested on the two energy production problems. The results confirm the competitiveness of the proposed method to solve energy problems.

In the second part, we present MO-DYNAMOP, which is an extension of DYNAMOP to multi-objective combinatorial optimization problems. MO-DYNAMOP is used to solve a bi-objective version of the UCP which is complicated due to indirect representation. Indeed, in this case, decoding a genotypic solution involves solving a multi-objective problem. Then, many Pareto-equivalent phenotypic solutions can be generated from one genotypic solution. We propose and compare 3 decoding strategies to overcome this difficulty. A comparative study between MO-DYNAMOP and previously proposed methods for the bi-objective UCP is performed. Experiments indicate that MO-DYNAMOP performs considerably better.

Keywords :
metaheuristic, evolutionary algorithm, hybrid mathod, dynamic programming, multi-objective optimisation, indirect representation

Remerciements

Tout d'abord, je voudrais remercier Patrice Perny et Frederic Saubion de m'avoir fait l'honneur d'être rapporteurs de ma thèse. Je souhaite également remercier, Lucie Galand, Gilles Goncalves et Alexandre Caminada d'avoir accepté d'être membres de mon jury de thèse.

J'adresse toute ma gratitude à mes directeurs de thèse, El-Ghazali Talbi et Laetitia Jourdan. Tout au long de mon doctorat, ils ont su me laisser la liberté d'exprimer et tester mes idées tout en gardant sur mon travail un oeil critique et avisé. Nos continuelles discussions m'ont permis d'aquérir méthode et rigueur scientifique et m'ont parfois évité de me perdre dans d'infinies divagations mentales.

Je tiens à remercier plus particulièrement Laetitia, qui malgrès l'importance de l'ensemble de ses charges s'est toujours montrée très disponible et m'a apporté un soutien tout aussi bien sur le plan scientifique, qu'administratif et moral.

Il me faut également remercier, toutes les personnes ayant contribué à réduire le nombre de fautes de grammaire, de syntaxe et d'orthographe de ce manuscrit à un taux acceptable. Pour ce travail laborieux, je remercie mes parents Sylvie et Dominique Jacquin, ainsi que mes soeurs Lucie et Nicole Jacquin. Je remercie plus particulièrement mon père, qui n'est pas loin d'avoir lu l'intégralité de cette thèse et qui en plus des corrections orthographiques m'a fourni des critiques très pertinentes qui m'ont permi de clarifier certains passages. De plus, je remercie ma famille, pour le soutien qu'elle m'a apporté tout au long de mon parcours.

Durant ma thèse, j'ai eu le plaisir d'encadrer plusieurs étudiants et le travail effectué avec eux m'a été très bénéfique, pour cela je les remercie. Tout d'abord, Yanis Nait Abdelaziz qui a travaillé sur l'implémentation d'un algorithme génétique classique pour le problème d'affectation d'unités mono-objectif, puis Lucien Moussin, qui a étendu cet algorithme à l'optimisation multi-objectif ainsi que Guillaume Pataut avec qui j'ai travaillé sur le problème de production d'énergie hydraulique.

Je souhaite également remercier particulièrement, mes amis doctorants, Martin Drozdik, Sezin Afsar et E-katerina Alexiva. En effet, ils ont toujours été disponibles pour relire et corriger les divers documents que j'ai rédigé en anglais. De plus, nos sympathiques discutions en langue anglaise, m'ont permises d'enrichir mon vocalubulaire et d'aquérir l'aisance orale qui me manquait.

Pour leur aide lors de ma soutenance, je remercie Phuong Dang, Emilie Allart, Martin Drozdick et Sezin Afsar. Je remercie chaleureusement ma mère pour le magnifique buffet de thèse qu'elle a préparé pour moi.

Enfin je remercie l'ensemble des personnes rencontrées dans l'équipe Dolphin. Plus largement je remercie toutes les personnes rencontrées lors de mes années de doctorat avec lesquelles j'ai eu d'agréables discussions, scientifiques ou non. Avec lesquelles j'ai partagé des repas, des cafés ou du temps hors du travail. Je suis enchantée des rencontres humaines, que j'ai faites pendant ma thèse, elles ont toutes été très enrichissantes car les personnes rencontrées ont toutes des personnalités originales qui les rendent uniques et intéressantes.

Table des matières

Table des figures

Introduction générale

Cette thèse s'intéresse à la proposition d'une méthode hybride combinant programmation dynamique et algorithme évolutionnaire, DYNAMOP (*DYNamic programming based Metaheuristic for Optimization Problem*), dans le but de résoudre des problèmes de production d'énergie électrique mono-objectif et multi-objectif. Elle a été menée au sein de l'équipe DOLPHIN (*Discrete multiobjective Optimization for Large scale Problems with Hybrid DIstributed techNiques*) du Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL, CNRS, Université Lille 1), commune à l'équipe projet du même nom du centre de recherche Inria Lille-Nord Europe.

Beaucoup de problèmes d'optimisation combinatoire rencontrés en pratique, dans le cadre d'une application industrielle, sont complexes et difficiles à résoudre. En effet, ces problèmes sont généralement NP-difficiles et de grande taille, de sorte que les méthodes de résolution exactes traditionnelles ne permettent pas de les traiter en un temps de calcul raisonnable. En particulier les problèmes de planification de la production d'énergie électrique auxquels nous nous intéresserons dans ce manuscrit sont très hardus car ce sont des problèmes à variables mixtes non-linéaires qui impliquent de nombreuses contraintes dures. La difficulté des problèmes de planification de la production d'énergie électrique est également liée à l'interdépendance forte qui existe entre les variables de décisions. Cette interdépendance réside d'une part, dans le fait que la fonction objective n'est pas séparable et, d'autre part, dans le fait que les contraintes impliquent simultanément de nombreuses variables de décisions. Ceci entraîne qu'il n'est pas possible de définir des sous problèmes indépendants et que plusieures optimisations locales n'aboutissent pas forcément à un optimum global.

Les problèmes de production d'énergie électrique ont été très étudiés dans la littérature car ils représentent un enjeu économique et écologique important. Ainsi, de nombreuses méthodologies de résolution ont été proposées. D'une part, des méthodes exactes, qui permettent de s'assurer de l'obtention de la solution optimale mais dont l'application se limite à la résolution de problèmes simplifiés ou de petites tailles qui ne correspondent pas forcément à des cas d'études réalistes. Parmi ces méthodes, la programmation dynamique est très prometteuse car elle permet de résoudre les problèmes sans qu'il soit nécessaire de linéariser la fonction objectif ou les contraintes, mais elle est particulièrement coûteuse en terme de temps de calcul ce qui la rend généralement inutilisable par les industriels.

D'autre part, les métaheuristiques qui représentent une classe de méthodes de résolution approchée peuvent permettre d'obtenir rapidement une solution réalisable de qualité. En revanche, ces méthodes ne permettent ni de garantir l'optimalité de la solution qu'elles fournissent ni ne donnent aucune garantie de performance. Ainsi, une métaheuristique peut converger prématurément vers un optimum local ou nécessiter un grand nombre d'itérations avant de converger totalement. De plus, dans le cadre des problèmes de production d'énergie électrique, la grande interdépendance entre les variables de décisions complique le parcours de l'espace des solutions réalisables par les métaheuritiques et impacte donc négativement

la convergence de ces méthodes.

Afin de dépasser les difficultés liées aux méthodes traditionnelles, des méthodes hybrides ont été proposées. Ce sont des méthodes créées en combinant une ou plusieurs méthodes classiques. En particulier, ces dernières années, les matheuristiques, qui sont des méthodes hybrides combinant méthodes exactes et métaheuristiques, ont constitué un domaine de recherche et de développement très actif. En effet ces méthodes sont conçues de façon à permettre l'obtention de solutions très précises en des temps de calcul qui restent raisonnables. C'est pourquoi nous porterons dans cette thèse un intérêt particulier à ce type de méthodes et chercherons à proposer une métaheuristique hybride entre programmation dynamique et métaheuristiques permettant de réduire les inconvénients des deux méthodes à savoir : D'une part, la vulnérabilité des métaheuristiques aux dépendances entre les variables de décisions et leur mauvaise précision et d'autre part, les temps de calculs démesurés de la programmation dynamique.

Cependant, comme la plupart des problèmes d'optimisation issus du monde réel, la planification de la production d'énergie peut nécessiter la considération simultanée de plusieurs critères, notamment afin de prendre en considération la dimension écologique du problème. La prise en compte de plusieurs objectifs à minimiser entraîne des difficultés supplémentaires. En effet, contrairement au cas mono-objectif, il n'existe pas de solution optimale unique mais un ensemble de solutions dites Pareto optimales, qui sont des solutions pour lesquelles l'amélioration à l'égard d'une des fonctions objectifs entraînerait invariablement une détérioration par rapport à une autre fonction objectif.

Dans cette thèse, la méthode de résolution proposée pour résoudre les versions mono-objectif des problèmes de production d'énergie que nous étudierons, sera donc étendue à l'optimisation multi-objectif. Cependant, nous verrons que la mise en place d'hybridation dans le contexte de l'optimisation multi-objectif est plus difficile. En effet, si en mono-objectif les méthodes concernées fonctionnent en s'échangeant des informations sur une solution unique, en multi-objectif elles doivent collaborer pour construire un ensemble de solutions Pareto optimales. Nous aborderons, en particulier, la problématique de la représentation indirecte en multi-objectif. En effet, en mono-objectif cette stratégie est très utilisée, elle consiste à utiliser une métaheuristique manipulant des solutions partielles qui sont ensuite décodées, pour obtenir la solution complète, à l'aide d'une méthode exacte ou d'une heuristique de qualité. Mais, en multi-objectif, il devient délicat d'utiliser ce genre de stratégie car à une solution partielle, un ensemble de solutions complètes Pareto optimales, peuvent être associées. Les problématiques liées à l'adaptation de la méthode hybride proposée au multi-objectif, sont très intéressantes, car les méthodes hybrides sont tout aussi prometteuses en multi-objectifs qu'en mono-objectif mais restent encore très peu étudiées dans la littérature.

Cette thèse s'organisera donc en deux parties. La première partie sera consacrée à l'optimisation combinatoire mono-objectif. Nous commencerons donc cette partie en introduisant les concepts de bases de l'optimisation combinatoire mono-objectif et en introduisant les principales méthodes de résolution existantes. Puis, dans un second chapitre, nous introduirons les deux problèmes d'applications qui nous intéresserons. Ces problèmes sont tous deux liés à la production énergétique, il s'agit du problème d'affectation d'unités, plus connu sous son nom anglais *Unit Commitment Problem* et d'un problème de planification de l'utilisation des eaux dans un réseau hydro-éléctrique. Dans ce chapitre, nous chercherons à comprendre ce qui rend ces problèmes difficiles à résoudre avec des méthodes d'optimisation combinatoire classiques. Dans la suite de cette partie, nous présenterons d'abord de manière générique DYNAMOP qui est une hybridation entre programmation dynamique et algorithme génétique. Puis nous l'appliquerons à nos deux cas d'étude. Cette

méthode constitue la principale contribution de ce mémoire, sa proposition a été motivée par les difficultés liées à la résolutions de nos deux problèmes d'application mais elle peut s'adapter à la résolution d'une large classe de problèmes.

La seconde partie sera consacrée à l'optimisation multi-objectif. Dans cette partie, nous présenterons tout d'abord l'ensemble des concepts nécessaires à la définition d'un problème d'optimisation multi-objectif et les principales méthodes de résolution existantes pour résoudre ce type de problèmes. Puis, nous montrerons comment DYNAMOP peut être généralisé à la résolution de problème d'optimisation multi-objectif, l'algorithme ainsi obtenu sera appelé MO-DYNAMOP. Par la suite nous appliquerons MO-DYNAMOP à la résolution d'une version multi-objectif du problème d'affectation d'unités. Dans cette version du problème, la minimisation des émissions en gaz à effet de serre est prise en considération en plus de la minimisation des coûts de production. Cependant, nous verrons que la résolution de ce problème pose une difficulté supplémentaire qui est la généralisation du concept de représentation indirecte pour l'optimisation multi-objectif. Ce concept de représentation indirecte semble très prometteur, car il permettrait de proposer de nombreuses hybridations entre des approches exactes en multi-objectif et des métaheuristiques, mais il semble n'avoir jamais été abordé dans la littérature scientifique. Nous proposerons donc plusieurs méthodes génériques de décodage que nous comparerons. Nous pensons que cette étude constitue une des contributions majeures de ce manuscrit. Une fois cette étude menée, nous consacrerons un chapitre à expliquer comment nous avons appliqué MO-DYNAMOP au problème d'affectation d'unités multi-objectif et à étudier les résultats de cette application.

Nous conclurons le manuscrit en synthétisant les principales contributions présentées, et en dénombrant de nouvelles voies à explorer suite à ce travail de thèse.

Plan détaillé du manuscrit

Partie I. DYNAMOP : une méthode de programmation dynamique basée sur les métaheuristiques.

Chapitre 1. Le premier chapitre défini ce qu'est un problème d'optimisation combinatoire en mono-objectif ainsi que les principales méthodes de résolution pour ce type de problèmes.

Nous donnerons une description plus détaillée des approches qui nous intéresserons dans la suite du manuscrit, à savoir la programmation dynamique, les algorithmes génétiques et les méthodes hybrides entre ces deux méthodes de résolution.

Chapitre 2. Le second chapitre introduit les deux problèmes d'application qui seront étudiés dans cette thèse. Le premier, est un problème très académique, il s'agit du problème d'affectation d'unités plus connu sous son nom anglais *Unit Commitment Problem*. Le second est un problème issu d'un cas industriel réel qui consiste à chercher une planification des débits d'eau à travers les conduites et turbines d'un réseau hydro-électrique qui permette de maximiser le profit.

Nous donnons, pour chacun de ces problèmes, une formalisation mathématique précise, avant d'étudier leurs complexités. Nous dressons également un état de l'art des méthodes de résolution ayant été proposées dans la littérature pour résoudre ces problèmes.

Chapitre 3. Le troisième chapitre est le noyau de cette thèse. Nous y présentons DYNAMOP, une méthode hybride entre programmation dynamique et métaheuristique. Cette méthode est la principale contribution de cette thèse.

Dans ce chapitre, nous commençons par présenter les problématiques liées à la résolution de certains problèmes, dont en particulier nos deux problèmes d'application, qui ont motivé la proposition de notre méthode. Nous présentons ensuite en détails le principe de fonctionnement de DYNAMOP, puis nous concluons en discutant de l'intérêt potentiel de DYNAMOP.

Chapitre 4. Dans le chapitre 4, DYNAMOP est appliqué au problème d'affectation d'unités. L'application de notre méthode à un problème très étudié de la littérature permet d'en évaluer la qualité en la comparant aux autres algorithmes existants.

Chapitre 5. Dans ce chapitre, DYNAMOP est appliqué à un problème d'application réel, qui est le problème de planification des débits d'eau dans un réseau de production d'énergie hydro-électrique. Les données que nous utilisons nous ont été fournie par une entreprise, ce qui nous permet d'évaluer le potentiel de DYNAMOP sur des jeux de données réels.

Partie II. MO-DYNAMOP : Adaptation de DYNAMOP à l'optimisation multi-objectif.

Chapitre 6. Le premier chapitre de la seconde partie, introduit les concepts de base liés à l'optimisation multi-objectif.

Les définitions essentielles à la compréhension de ce qu'est un problème d'optimisation multi-objectif sont données ainsi qu'un état de l'art sur les principales méthodes de résolution. Nous décrivons également, dans ce chapitre, le problème d'affectation d'unités multi-objectif, qui servira de cadre applicatif pour cette seconde partie.

Chapitre 7. Dans ce chapitre, nous expliquons comment DYNAMOP peut être généralisé pour permettre de résoudre des problèmes d'optimisation multi-objectif. Dans ce but, nous commençons par expliquer en détails ce que sont la programmation dynamique multi-objectif et les algorithmes évolutionnaires multi-objectif.

Chapitre 8. L'étude faite dans ce chapitre constitue une autre contribution originale. En effet, dans ce chapitre, nous nous intéressons au problème de la représentation indirecte avec un système de décodage basé sur la résolution d'un problème multi-objectif. Cette problématique est issue de l'étude du problème d'affectation d'unités multi-objectif et les solutions que nous proposons seront testées sur ce problème. Néanmoins, nous pensons que l'idée de représentation indirecte peut être réutilisée dans le cadre de la résolution d'autres problèmes. Nous avons donc pris soin, d'introduire la problématique et les solutions que nous proposons et étudions de manière aussi générique que possible.

Chapitre 9. Dans ce chapitre nous appliquons MO-DYNAMOP au problème d'affectation d'unités multi-objectif. Pour cela, nous utilisons une représentation indirecte avec un décodeur multi-objectif en nous basant sur l'étude faite dans le chapitre précédent. Sur ce problème nous testons DYNAMOP en utilisant différents paradigmes d'algorithmes évolutionnaires tels que NSGA-II, IBEA et SMS-EMOA, une étude comparative est faite entre les différentes versions de MO-DYNAMOP ainsi obtenues. Nous comparons également MO-DYNAMOP à trois algorithmes issus de la littérature.

Pour finir, nous résumons les contributions de cette thèse et proposons une ouverture sur de nouvelles directions de recherche à court terme ainsi qu'à long terme, dans un chapitre de conclusion.

Première partie

DYNAMOP : Une méthode de programmation dynamique basée sur les métaheuristiques

Chapitre 1

Méthodes de résolution de problèmes d'optimisation combinatoire

Ce premier chapitre introduit des concepts et des notations qui nous seront utiles par la suite, en particulier vous trouverez dans ce chapitre :

- *une définition d'un problème d'optimisation combinatoire mono-objectif ;*
- *un état de l'art des principales méthodes de résolution des problèmes d'optimisation combinatoire mono-objectif ;*
- *une description détaillée de la méthode de programmation dynamique ;*
- *une description détaillée des algorithmes génétiques.*

1 Introduction

Un problème d'optimisation combinatoire consiste à sélectionner la meilleure solution dans un espace discret (autrement dit énumérable) de solutions réalisables. La notion de meilleur est donnée par un critère de qualité via une fonction objectif. Formellement, un problème d'optimisation combinatoire peut être défini par, Ω l'ensemble discret des solutions réalisables du problème, on parle alors d'espace de recherche, et $f : \Omega \to \mathbb{R}$ la fonction objectif. Le but est de trouver $s^* \in \Omega$ tel que :

$$s^* = \underset{s \in \Omega}{argopt}\, f(s)$$

Où $argopt$ correspond à $argmin$ dans le cas où l'on cherche à minimiser l'objectif, on parle alors de problème de minimisation. Si au contraire l'objectif doit être maximisé, on parle de problème de maximisation et $argopt$ correspond à $argmax$.

L'optimisation combinatoire est une branche très importante de la recherche opérationnelle. En effet, les problèmes d'optimisation combinatoire représentent une catégorie de problèmes très difficiles à résoudre [139] et de nombreux problèmes pratiques peuvent être formulés sous cette forme [142].

Il existe deux classes de méthodes pour résoudre ces problèmes d'optimisation :

- Les méthodes exactes : Ce sont des méthodes qui garantissent mathématiquement l'optimalité de la solution qu'elles fournissent. On les utilise donc en priorité. Néanmoins, pour les problèmes dit NP-difficiles, il n'y a pas de méthode exacte connue permettant d'obtenir une solution en un temps de calcul raisonnable si la taille des données est importante (le temps de calcul augmente exponentiellement avec la

taille des données). De plus, ces méthodologies exigent certaines spécificités sur la nature de la fonction objectif et de l'espace de recherche. Il y a donc beaucoup de cas pratiques où il n'est pas possible d'utiliser des approches exactes.

— Les méthodes approchées : Ces méthodes ne garantissent pas que la solution obtenue soit la solution optimale. Parmi elles on distingue les métaheuristiques, qui sont des méthodes très génériques, qui peuvent être appliquées aux problème indépendamment de leurs nature. Bien que ces méthodes ne donnent aucune garantie de performance elles sont couramment utilisées pour la résolution de problèmes pratiques qui sont généralement trop difficiles pour être résolus avec des méthodes exactes.

Dans ce chapitre nous allons faire un tour d'horizon des méthodes de résolution de problèmes d'optimisation combinatoire. Nous insisterons particulièrement sur la programmation dynamique et les algorithmes génétiques qui seront réutilisés par la suite. Puis nous ferons un état de l'art sur les approches hybrides existantes. Ce sont des algorithmes combinant plusieurs méthodes de résolution.

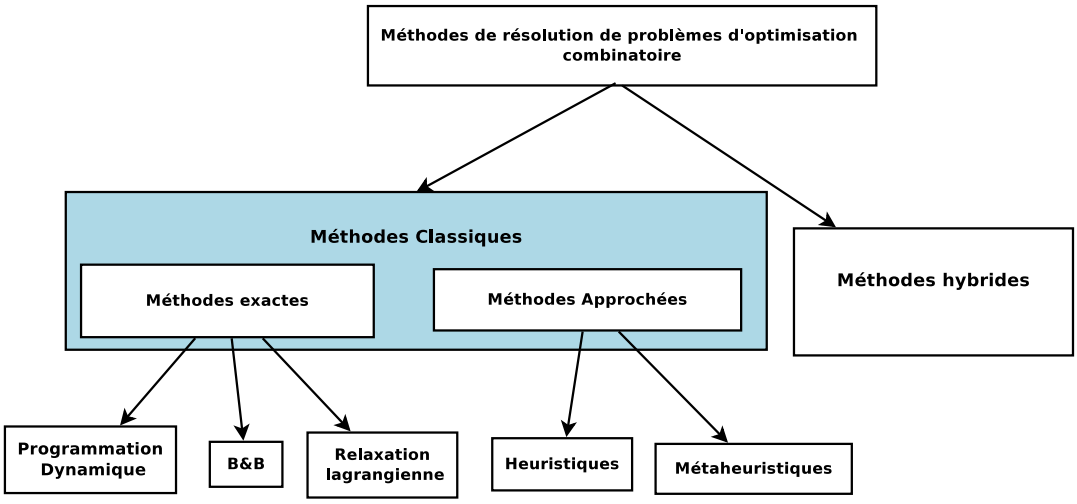

FIGURE 1.1 – Vue globale des approches de résolution des problèmes d'optimisation combinatoire.

2 La programmation dynamique

Les méthodes exactes sont les méthodes qui permettent de trouver la solution exacte d'un problème d'optimisation combinatoire et de prouver son optimalité. Parmi ces méthodes on trouve le *branch-and-bound* (B&B) [105], le *branch-and-cut* (B&C) [137], le *branch-and-price* (B&P) [17], le *branch-and-cut-and-price* (B&C&P) [52], la programmation dynamique [19], les méthodes basées sur la relaxation lagrangienne [197] et les méthodes utilisant la programmation par contraintes avec un solveur qui peut être un *branch-and-bound* [24].

Dans cette section nous allons présenter en détails la programmation dynamique. Nous nous focalisons sur la programmation dynamique car elle occupe une place centrale dans notre travail.

2.1 Principe

La programmation dynamique a été introduite en 1950 par Richard Bellman [19]. La programmation dynamique s'appuie sur un principe simple, appelé le *principe d'optimalité de Bellman* : "Une politique optimale est formée de sous-politiques optimales". Le mot politique désigne une séquence de décisions définissant une solution d'un problème d'optimisation. Concrètement, la programmation dynamique est une méthode utilisée pour résoudre des problèmes où une séquence de décisions optimale, $(d_1, d_2, ..., d_n)$, doit être trouvée. L'idée de base est que l'on peut déduire une ou la solution optimale d'un problème en combinant des solutions optimales d'une série de sous-problèmes consistant à choisir des séquences plus courtes de décisions. Les solutions des problèmes sont calculées de manière ascendante, c'est-à-dire qu'on débute par les solutions des sous-problèmes les plus petits pour ensuite déduire progressivement les solutions de l'ensemble.

2.2 Les éléments de la programmation dynamique

Un algorithme de programmation dynamique est entièrement défini par son équation fonctionnelle de programmation dynamique dont la formulation générale est [107] :

$$f(S) = opt_{d \in D(S)}\{R(S,d) \circ f(T(S,d))\} \tag{1.1}$$

Où S est un état défini dans un certain espace d'état $\mathbb{S}$, d est une décision à sélectionner dans l'espace de décision $D(S)$, la fonction *reward*, $R(S,d)$, est le coût de la décision d si elle est prise à l'état S , $T(S,d)$ est la fonction de transition et $\circ$ est un opérateur binaire. On se restreindra à la programmation dynamique discrète, où l'espace d'états et l'espace de décisions sont tous les deux discrets. Les éléments de l'équation fonctionnelle de programmation dynamique ont les caractéristiques suivantes :

- **Etat :** Un état S incorpore des informations sur l'ensemble des décisions qui ont été prises avant d'atteindre S. Il doit en fait donner le minimum d'informations sur les décisions passées permettant de définir les possibles décisions futures ainsi que leurs coûts. Si le problème consiste à trouver la meilleure séquence de décision $(d_1, d_2, .., d_n)$, un état S_i donnera les informations nécessaires sur le choix des décisions prises avant d'atteindre S_i, $(d_1, d_2, .., d_i)$, pour qu'il soit possible de définir le sous-problème : Trouver la meilleure séquence $(d_{i+1}, d_{i+2}, .., d_n)$ sachant que les décisions $(d_1, d_2, .., d_i)$ ont été prises. On dira que ce sous problème est le sous-problème associé à S_i. On notera S^* l'état initiale ou état objectif qui correspond à l'état pour lequel aucune décision n'a encore été prise.
- **Espace de décisions :** L'espace de décisions $D(S)$ est l'ensemble des décisions qui peuvent être prises à partir de l'état S. Si $D(S) = \emptyset$, c'est à dire qu'il n'y a pas de décision possible à partir de l'état S, on dit que S est un état terminal.
- **Fonction objectif :** La fonction objectif f est la fonction objectif du sous-problème associé à l'état S. C'est le profit ou coût optimal qui peut être obtenu en partant de l'état S. L'objectif de la programmation dynamique est de calculer $f(S)$ pour l'état initial S^*.
- **Fonction Coût :** La fonction coût ou fonction *reward* R dépend d'un état S et d'une décision d, c'est le coût ou le profit qui peut être attribué à la décision d si elle est prise à partir de l'état S. La fonction coût $R(S,d)$ doit être séparable par rapport aux coûts ou profits qui ont été attribués à toutes les autres décisions. Elle ne doit dépendre que de S et d. La valeur de la solution optimale (la valeur de la

fonction objectif pour l'état initiale), $f(S^*)$, est la combinaison des coûts pour la séquence optimale des décisions prises en partant de l'état initial.

— **Fonction de transformation :** La fonction de transformation (ou de transition) T est une fonction de $S * D(S)$ dans $\mathbb{S}$. $T(S, d)$ détermine l'état S' qui sera atteint si la décision d est prise à l'état S. Elle est injective par rapport à $d \in D(S)$. $T^{-1}(S, S')$ renvoie la décision qui doit être prise pour atteindre S' en partant de S ($S' \in T(S, D(S))$).

— **Opérateur :** L'opérateur $\circ$ est un opérateur binaire, habituellement l'addition, la multiplication, la minimisation ou la maximisation, qui permet de combiner le retour de décisions séparables, il doit être associatif si le retour des décisions dépend de leur ordre.

— **États terminaux :** Les états terminaux sont définis par une fonction booléenne, appelée fonction de base et notée f_b :

$$f_b(S) = 0 \Leftrightarrow D(S) \neq \emptyset$$
$$f_b(S) = 1 \Longrightarrow f(S) = 0$$

Dans certains cas, il existe des états à partir desquels il n'y a pas de décision possible mais qui ne correspondent pas à des politiques valides (solutions réalisables). Dans ce cas, on utilise une fonction de base tertiaire à valeur dans $\{0, 1, 2\}$, telle que :

$$f_b(S) = 0 \Leftrightarrow D(S) \neq \emptyset$$
$$f_b(S) = 1 \Longrightarrow f(S) = 0$$
$$f_b(S) = 2 \Longrightarrow f(S) = \pm\infty$$

La valeur infinie permet d'interdire la transition vers des états à partir desquels il n'est pas possible de construire une politique valide.

Pour résoudre un problème, par programmation dynamique, il faut tout d'abord définir tous les éléments précédemment cités puis utiliser la relation de récurrence 1.1. Nous verrons par la suite quelques exemples d'applications concrètes.

2.3 Représentation graphique

Un problème de programmation dynamique peut être modélisé comme un problème de plus court chemin dans un graphe [38]. Ce graphe est appelé graphe d'états. Chaque état S de l'espace d'états est modélisé par un sommet et chaque décision $d \in D(S)$ est modélisée par un arc reliant S à $T(S, d)$ et évalué par $R(S, d)$. L'état initiale S^* correspond à un sommet qui n'est le sommet final d'aucun arc. Les états terminaux valides correspondent à des états à partir desquels aucun arc ne part. En revanche si l'état S_T est un état final mais n'est pas un valide (si $f_b(S_T) = 2$), un sommet fictif S_f est ajouté de sorte que l'arc joignant le sommet associé à S_T à S_f soit évalué par $\pm\infty$. L'objectif du problème est alors de trouver le plus court chemin reliant le sommet initial S^* à un sommet terminal. La figure 1.2 illustre cette modélisation. Le sommet vert correspond au sommet initial, le sommet rouge est un sommet terminal correspondant à un état terminal valide, le sommet rose correspond à un état terminal non valide et le sommet orange à un état fictif.

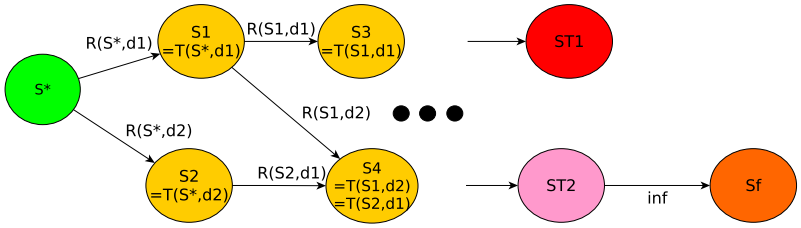

FIGURE 1.2 – Représentation graphique de la programmation dynamique.

Les graphes d'états ont la particularité d'être des graphes orientés et acycliques [38]. Ces graphes sont les graphes qui peuvent être linéarisés, c'est à dire que l'on peut disposer leurs sommets sur une ligne de sorte que tous les arcs aillent de la gauche vers la droite (Figure 2.3).

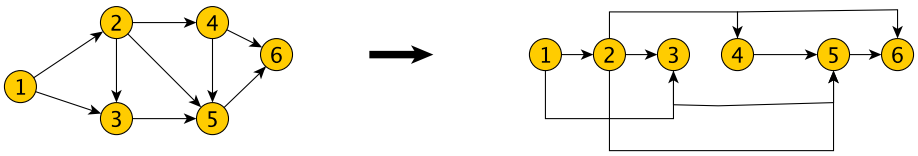

FIGURE 1.3 – Linéarisation d'un graphe orienté acyclique.

3 Les métaheuristiques

Les métaheuristiques sont des algorithmes stochastiques qui permettent d'approximer les solutions de problèmes d'optimisation difficiles pour lesquels ont ne connaît pas de méthode classique plus efficace. Elles sont souvent utilisées dans le cadre d'applications à des problèmes d'optimisation réels. En effet les problèmes réels sont souvent des problèmes NP-difficiles avec une fonction objectif non linéaire et dont la taille des données est très importante. Pour cette raison il n'est pas possible de les résoudre avec des méthodes exactes en un temps raisonnable. Lorsqu'on s'attaque à des problèmes réels, il faut donc se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.

Il existe un grand nombre de métaheuristiques différentes qui peuvent être classifiées en deux groupes : les métaheuristiques à solution unique et les métaheuristiques à population de solutions. Dans cette section, nous décrirons brièvement les méthodes associées à chacune de ces familles puis nous décrirons avec plus de détails les algorithmes génétiques qui seront utilisés dans la suite du manuscrit.

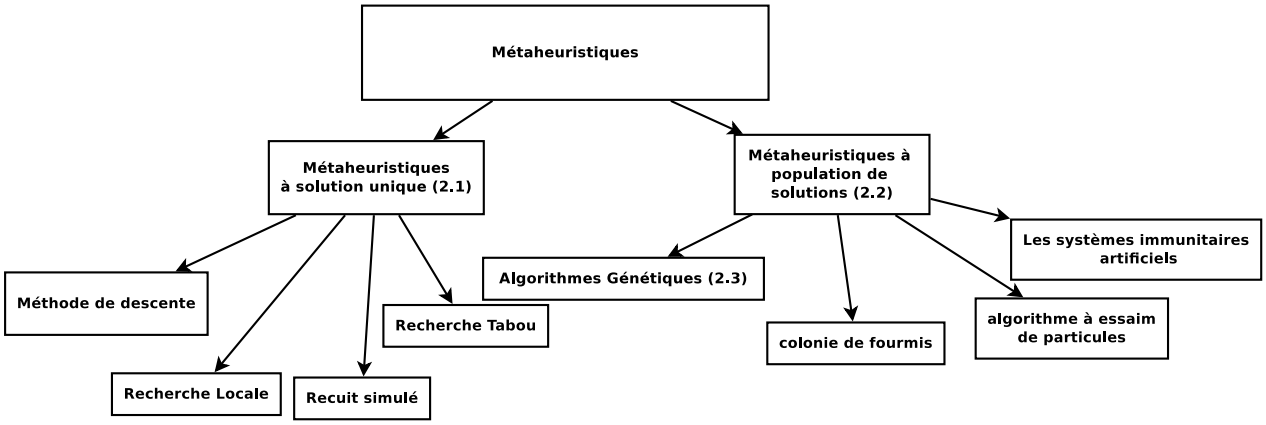

FIGURE 1.4 – Classification des métaheuristiques.

La figure 1.4 permet de visualiser la classification proposée et l'organisation de cette section.

3.1 Les métaheuristiques à solution unique

Les métaheuristiques à solution unique sont des méthodes itératives qui partant d'une solution initiale cherchent à chaque étape à l'améliorer en parcourant son voisinage [175]. Parmi ces méthodes les plus connues sont les recherches locales ou méthodes de descente [1] (LS), les recherches locales itérées [114] (ILS), le recuit simulé [95] (RS) et la recherche tabou [59] (TS).

Les méthodes de descente

Les méthodes de descente [1] sont des méthodes qui permettent de trouver très rapidement un optimum local. Elles se basent sur la définition d'un voisinage de solutions. A chaque itération la solution est remplacée par une solution de son voisinage ayant une meilleure évaluation et ceci jusqu'à ce qu'il n'y ait plus de solution améliorante dans le voisinage de la solution. Les différentes méthodes de descente se distinguent par les procédures de choix d'un meilleur voisin qu'elles utilisent. Par exemple le *Hill Climbing* consiste à choisir le voisin de la solution courante ayant la meilleure qualité (exploration exhaustive du voisinage). Le *First Improvement Hill Climbing* consiste à choisir le premier voisin rencontré qui a une meilleure qualité (exploration partielle du voisinage).

Les recherches locales itérées

Les méthodes de descente restent bloquées au premier optimal local rencontré. Pour pallier à cette difficulté et augmenter les chances de trouver l'optimum global les méthodes de recherches locales itérées [114] lancent itérativement plusieurs recherches locales démarrant de différentes solutions initiales. La solution initiale de la recherche locale d'une itération est construite à partir de la solution obtenue à l'itération précédente. Le processus se répète jusqu'à l'atteinte d'un critère d'arrêt. Le choix du critère d'arrêt et de la méthode de construction d'une nouvelle solution initiale définissent différents types de recherches locales itérées.

Le recuit simulé

La méthode du recuit simulé [95] utilise une procédure d'exploitation du voisinage qui permet de sélectionner une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre T est déterminé qui décroît tout au long de l'algorithme pour tendre vers 0. La probabilité d'acceptation des solutions détériorantes diminue avec ce paramètre. L'intérêt du recuit simulé est qu'il existe une preuve de la convergence asymptotique. Ainsi, lorsque certaines conditions sont vérifiées (schéma de décroissance particulier de T), on a la garantie d'obtenir la solution optimale. Malheureusement, le paramétrage recommandé par la théorie n'est pas réaliste et il faut beaucoup de temps pour arriver à paramétrer ces méthodes. Cette méthode peut nécessiter un critère d'arrêt, dans le cas où le paramétrage "optimal" n'a pas été trouvé.

La recherche tabou

La recherche tabou [59] consiste à remplacer, à chaque itération, la solution par la meilleure solution parmi ses solutions voisines. Si toutes les solutions voisines sont de moins bonne qualité, il est possible de détériorer la solution courante. C'est ce processus qui permet d'échapper aux optimaux locaux. Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. L'idée est d'interdire de revenir sur les dernières positions explorées qui sont stockées dans une liste (liste tabou). Les démonstrations de convergence pour la recherche tabou existent, mais supposent des conditions strictes, rarement présentes en pratique. Là aussi la définition d'un critère d'arrêt est nécessaire. La gestion de la mémoire (taille de la liste tabou) et la définition du voisinage d'une solution sont des paramètres de la recherche tabou.

3.2 Les métaheuristiques à population de solutions

Les métaheuristiques à population de solutions manipulent un ensemble de solutions qu'elles tentent d'améliorer itérativement. L'intérêt est d'utiliser la population comme un facteur de diversité. Parmi les métaheuristiques à population de solutions, les plus connues et les plus couramment utilisées sont les colonies de fourmis [42], les algorithmes à essaims de particules [90] , les systèmes immunitaires artificiels [50,91] et les algorithmes génétiques [73]. Dans cette partie, nous allons présenter très brièvement les 3 premières méthodes. Par la suite nous présenterons les algorithmes génétiques plus en détails car ils seront utilisés dans la suite du manuscrit.

Les colonies de fourmis

L'algorithme des colonies de fourmis (ACO) [42] se base sur l'observation des fourmis. Celles ci résolvent naturellement des problèmes complexes en utilisant une communication indirecte basée sur l'utilisation de phéromones. L'idée est d'utiliser une représentation du problème sous forme d'un graphe dans lequel les solutions sont des chemins entre des états initiaux et finaux et une solution optimale un plus court chemin. Après avoir atteint un état final, une fourmi dépose des phéromones sur l'ensemble des arcs de son trajet dont la quantité dépend de la qualité de la solution trouvée. Ces phéromones ont une durée de vie et s'estompent donc au fil des itérations. Par ailleurs, la probabilité qu'une fourmi emprunte un arc plutôt qu'un autre est proportionnelle à la quantité de phéromones présente sur cet arc. De ce fait, le plus court chemin a une probabilité plus importante d'être emprunté par les fourmis que les autres chemins et sera à terme emprunté par toutes les fourmis.

Les algorithmes à essaims de particules

Les algorithmes à essaims de particules [90] s'inspirent du comportement social des animaux évoluant en essaim, tels que les nuées d'oiseaux et les bancs de poissons. Dans ce genre des société des règles simples, telles que "aller dans la même direction" ou "rester proche de ses voisins", suffisent à maintenir la cohésion de l'essaim et permettent la mise en oeuvre de comportements collectifs complexes et adaptatifs. Les algorithmes à essaims de particules modélisent des agents simples, appelés particules qui représentent des solutions potentielles du problème d'optimisation. Une particule i est définie par sa position x_i dans l'espace des solutions et par un vecteur direction d_i qui est celui de son dernier déplacement. Le déplacement d'une particule est dirigé par un vecteur calculé comme la somme pondéré de trois vecteurs de direction correspondant à trois composantes :

- Une composante d'inertie : la particule tend à suivre sa direction courante de déplacement.
- Une composante cognitive : la particule tend à se fier à sa propre expérience et à suivre la direction du meilleur site par lequel elle est déjà passée.
- Une composante sociale : la particule tend à se fier à l'expérience de ses congénères et donc à suivre la direction du meilleur site déjà atteint collectivement par l'essaim.

Ainsi à chaque itération toutes les particules de l'essaim se déplacent à la recherche de l'optimum global et ceci jusqu'à l'atteinte d'un critère d'arrêt préalablement défini.

Les systèmes immunitaires artificiels (SIA)

SIA [50, 91] se base sur des principes d'immunologie pour résoudre des problèmes d'optimisation. Ces algorithmes exploitent les caractéristiques du système immunitaire pour ce qui est de l'apprentissage et de la mémorisation comme moyens de résolution de problèmes.

Les fonctionnements simulés dans les SIA comprennent notamment :

- La reconnaissance de motifs permettant de différencier le soi du non-soi qui est simulée pour repérer les similarités entre les solutions.
- L'hypermutation qui permet de générer de nouvelles solutions en transformant les solutions existantes.
- La sélection clonale pour les cellules B fabriquant les anticorps qui permet d'amplifier la présence des meilleurs solutions.
- La sélection négative pour les cellules T qui permet d'éliminer des solutions trop mauvaises.

De plus amples informations sur ces méthodes peuvent être trouvées dans [39].

3.3 Les algorithmes génétiques

La méthode proposée dans ce travail est une hybridation entre un algorithme génétique et la programmation dynamique. Pour comprendre la suite, il est important d'être familier avec le vocabulaire nécessaire à la compréhension de ces algorithmes. Cette section a donc pour objectif de présenter en détail le principe des algorithmes génétiques.

Les algorithmes génétiques sont inspirés par la théorie de l'évolution introduite par Charles Darwin en 1859 [37]. De ce fait, la terminologie utilisée pour définir un algorithme

génétique est en grande partie issue de la biologie et il convient donc de rappeler au préalable quelques termes de génétique :

- Toutes les informations nécessaires au développement d'un organisme vivant sont contenues dans ses **chromosomes** qui sont des chaînes d'ADN (acide désoxyribonucléique).

- Sur chacun de ces chromosomes, une suite de **gènes** constitue une chaîne qui code les fonctionnalités de l'organisme (la couleur des yeux par exemple).
- L'ensemble des gènes d'un individu constituent son **génotype**.
- La position d'un gène sur le chromosome est son **locus**.
- Les différentes versions d'un même gène sont appelées **allèles**.
- Le **phénotype** est l'ensemble des caractères observables d'un individu dans son environnement biologique.

Les algorithmes génétiques utilisent l'analogie avec la théorie de l'évolution qui propose qu'au fil du temps, les gènes conservés au sein d'une population donnée sont ceux qui permettent d'obtenir les caractéristiques phénotypiques permettant à l'individu de s'adapter aux besoins de son environnement.

Principe général des algorithmes génétiques

L'idée de l'algorithme génétique est de manipuler un ensemble de solutions (population) où chaque solution (appelée génotype ou individu génotypique) est modélisée comme une séquence d'informations (les gènes) en utilisant les principes de la théorie de l'évolution.

La première étape consiste donc à générer une population initiale. Ceci fait les étapes suivantes sont exécutées itérativement jusqu'à ce que le critère d'arrêt choisi soit réalisé :

- On associe à chaque solution de la population un indice de qualité (fitness) qui est une mesure abstraite permettant de classer les chromosomes. Son calcul s'effectue à partir de l'évaluation de l'objectif à optimiser du problème.
- On sélectionne des parents pour la reproduction en se basant sur leurs fitness.
- On effectue le processus de reproduction qui peut consister en :

  - Un croisement qui permet d'obtenir un ou plusieurs individus enfants héritant d'une portion des gènes de chacun de ses parents.
  - Une mutation qui permet d'obtenir un nouvel individu en modifiant aléatoirement certains gènes d'un individu connu.
- On effectue un processus de remplacement pour construire une nouvelle population en sélectionnant des individus parmi l'ancienne population et les individus enfants générés précédemment.

La figure 1.5 montre le fonctionnement itératif simplifié d'un algorithme génétique. Pour pouvoir développer un algorithme génétique il faut déterminer :

- la représentation des solutions (codage du chromosome) ;
- la fonction d'évaluation ;
- les opérateurs employés ;
- les paramètres à fixer (taille de la population, probabilité d'application des opéra-

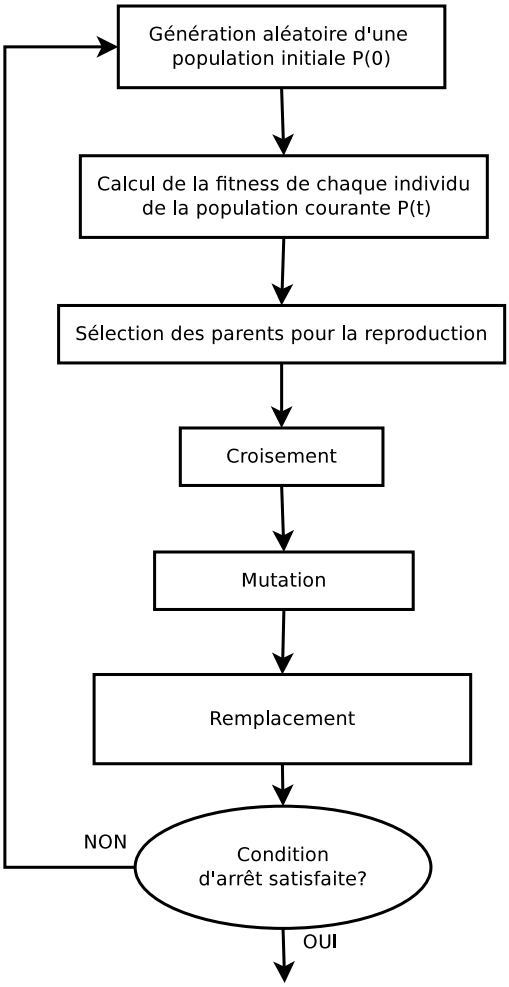

FIGURE 1.5 – Fonctionnement général d'un algorithme génétique.

teurs...).

Représentation

La représentation des solutions doit être **complète**, c'est à dire que toutes les solutions possibles du problème doivent pouvoir être codées à l'aide de cette représentation. Si la représentation peut produire plusieurs chromosomes codant la même solution (phénotypique) on dit qu'elle est **redondante**. La redondance, si elle est élevée, peut poser des problèmes pour la convergence [85].

Goldberg donne deux principes de base pour choisir la représentation d'une solution d'un algorithme génétique [60] :

— Le codage doit contenir des blocs de construction ayant un sens, c'est à dire dont on peut quantifier l'impact sur la fonction objectif.
— L'alphabet (le nombre d'allèles par gène) doit être petit.

Fonction d'évaluation

Cette fonction permet d'associer à chaque génotype une valeur de qualité ou fitness qui sera utilisée pour les classer. Elle est utilisée lors du processus de sélection. Son choix va fortement influer sur le succès de l'algorithme car elle produit la pression qui permet de faire évoluer la population vers des individus de meilleure qualité. Souvent, elle correspond simplement à l'évaluation de la fonction objectif pour l'individu encodé par le chromosome mais elle peut être différente. Par exemple, le *scaling* (mise à l'échelle) peut être utilisé. Il modifie la fonction objectif afin de réduire ou d'amplifier artificiellement les écarts entre les individus, ce qui peut faciliter le parcours de l'espace des solutions ou permettre de rejeter plus rapidement les mauvaises solutions. De plus si toutes les solutions représentables ne correspondent pas à des solutions réalisables, la fonction d'évaluation peut inclure des stratégies de pénalisation des solutions invalides.

Opérateurs génétiques

Les opérateurs génétiques agissent directement sur les individus de la population. Ils sont de différents types :

— **Initialisation :** Il sert à générer la population initiale. Un bon opérateur d'initialisation doit permettre de générer des chromosomes bien répartis dans l'espace génotypiques afin de fournir à l'algorithme du matériel génétique varié.
— **Opérateur de sélection :** La sélection tend à augmenter l'importance des bonnes solutions par rapport aux mauvaises. Les bonnes solutions étant supposées être plus prometteuses pour engendrer les individus de la nouvelle génération [188]. Il y a différentes stratégies de sélection dont les plus connues sont :
  — *La sélection par rang* [196] : Dans ce mode de sélection les individus sont déjà classés en fonction de leur fitness puis les individus sont sélectionnés avec une probabilité proportionnelle à leur rang dans cette classification.
  — *La sélection sur la fitness* [60] : La probabilité d'être sélectionné pour chaque individu est directement proportionnelle à son adaptation au problème (c'est à dire à sa fonction fitness). Afin de sélectionner un individu, on simule une roue

de la fortune biaisée. Cette roue est une roue de la fortune classique sur laquelle chaque individu est représenté par une portion proportionnelle à son adaptation. On effectue ensuite un tirage au sort homogène sur cette roue.

— *La sélection par tournoi* [13] : Un individu est sélectionné de la façon suivante : K individus sont tirés au sort aléatoirement dans la population (K est un paramètre appelé taille du tournoi) ; puis le meilleur parmi les K individus est sélectionné (cas déterministe) ou le vainqueur est choisi parmi les K individus avec une probabilité proportionnelle à sa fitness (cas probabiliste).

Il existe d'autres types de sélections moins utilisées qui sont décrites dans [13].

— **L'opérateur de croisement :** L'opérateur de croisement combine le matériel génétique de plusieurs parents pour obtenir un ou plusieurs enfants. Dans [175] E.G. Talbi souligne deux points importants à prendre en compte lors de la conception d'un opérateur de croisement :

— *L'héritabilité* : L'individu généré par l'opérateur de croisement doit hériter du matériel génétique des deux parents. On parle d'*héritabilité forte* si deux individus identiques génèrent un individu identique. Dans nos travaux nous nous référerons à un concept d'héritabilité un peu plus fort, que nous pouvons appeler *d'héritabilité phénotypique* pour signifier qu'un individu enfant hérite des caractéristiques phénotypiques des deux parents. Concrètement cela signifie que la fonction objectif de l'individu enfant peut être caractérisée par celles de ses parents.

— *La validité* : Un opérateur de croisement doit générer des individus valides.

— **L'opérateur de mutation** : La mutation est un opérateur unaire qui agit pour modifier un individu. Son rôle est d'apporter la diversité génotypique nécessaire à l'exploration de l'espace de recherche. Ceci est fait en effectuant une petite transformation dans le génotype de l'individu à muter. Certains points important doivent être pris en compte lors de la conception d'un opérateur de mutation [175] :

— *L'ergodicité :* Un opérateur de mutation est dit ergodique s'il est possible partant d'une solution quelconque d'atteindre n'importe quelle autre solution de l'espace de recherche en appliquant l'opérateur un nombre fini de fois.

— *La validité* : Un opérateur de mutation doit générer des individus valides.

— *La localité* : La mutation doit produire un changement minimal. La taille et l'impact de la mutation sont importants et doivent être contrôlables. La localité est l'effet sur la solution phénotypique (la valeur de la fonction objective) quand on applique la transformation sur le génotype. Si, lorsque la mutation produit un petit changement sur le génotype, cela entraîne un petit changement sur la solution phénotypique, on dit que la mutation est *fortement locale*. Dans le cas contraire on dit qu'elle est *faiblement locale*.

— **Opérateur de remplacement** : Le remplacement consiste en l'incorporation des nouvelles solutions dans la population courante. Le remplacement est dit total si l'ensemble des solutions nouvelles sont incorporées, sinon il est dit partiel. Lorsque la nouvelle population n'est constituée que de nouvelles solutions, on parle d'algorithme génétique générationnel. Dans le cas ou l'algorithme n'est pas générationnel, des stratégies sont mises en place pour sélectionner les individus à incorporer dans la nouvelle population. En générale ces stratégies sont élitistes, c'est à dire qu'elles privilégient l'incorporation des meilleurs solutions dans la population de la nouvelle génération. En notant N le nombre d'individus de la population, on peut distinguer deux types d'élitismes :

— Élitisme fort : Les N meilleurs individus parmi l'ensemble des individus enfants et parents sont sélectionnés.
— Élitisme faible : La nouvelle population est obtenue en sélectionnant les K individus parents ayant la meilleur fitness puis les $N - K$ individus enfants ayant la meilleur fitness. K étant un entier fixé.

4 Etat de l'art sur les méthodes hybrides entre métaheuristique et programmation dynamique

Le but de cette section est de donner une vision d'ensemble sur les stratégies d'hybridation existantes. Nous verrons donc tout d'abord comment ces méthodologies sont classifiées puis nous ferons un état de l'art plus précis sur les méthodes hybrides combinant métaheuristiques et programmation dynamique.

4.1 Les métaheuristiques hybrides

Pendant longtemps les chercheurs n'ont porté que très peu d'intérêt aux métaheuristiques hybrides [8]. Ce désintérêt était principalement une conséquence du fait que la recherche en optimisation combinatoire a longtemps été séparée en deux branches de recherche entre lesquelles il y avait peu de communication : l'une s'intéressant avant tout aux méthodes d'optimisation fondées sur les métaheuristiques et l'autre aux méthodes exactes fondées sur la programmation mathématique. Néanmoins les méthodes hybrides ont maintenant grandement gagné en popularité. En effet, les meilleurs résultats trouvés pour plusieurs problèmes d'optimisation combinatoire ont été obtenus avec des algorithmes hybrides. Il existe à présent de nombreuses stratégies pour construire des hybridations entre deux méthodologies classiques. Nous allons présenter dans la suite deux types de classification des métaheuristiques hybrides.

Classification Hiérarchique

La classification hiérarchique a été proposée par Jourdan *et al.* [86]. Elle se divise en deux classes : L'hybridation de *bas niveau* et l'hybridation de *haut niveau*. L'hybridation entre deux méthodes M_1 et M_2 est dite de bas niveau lorsque M_2 est utilisée pour remplacer l'une des fonctions de M_1. A l'inverse, on obtient une hybridation de haut niveau lorsque les deux méthodes sont hybridées sans que leur fonctionnement interne ne soit en relation.

Chacune des deux classes précédentes se subdivise en deux autres classes : relais et collaborative. Lorsque les métaheuristiques sont exécutées de façon séquentielle, l'une utilisant le résultat de la précédente comme entrée, l'hybridation est de type relais. L'hybridation collaborative se fait lorsque des agents coopèrent en parallèle pour explorer l'espace de solutions.

La combinaison des classes nommées précédemment donne quatre classes telles que illustrées figure 1.6. Ces classes sont décrites en détails ci dessous avec à chaque fois quelques exemples illustratifs de travaux réalisés.

— **Hybridation bas niveau à relais (BNR) :** Cette classe correspond aux algorithmes dans lesquels une méthode est incorporée dans une autre méthode. Les méthodes doivent ne pouvoir être exécutées que séquentiellement. C'est à dire que l'exécution de la méthode globale doit dépendre du résultat de la méthode incorporée. Dans [12], une méthode de *branch and cut* est proposée pour résoudre un

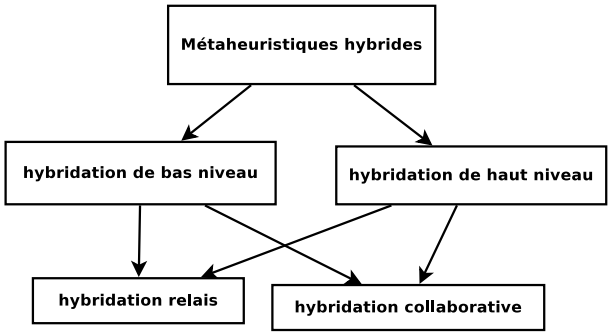

FIGURE 1.6 – Classification hiérarchique des métaheuristiques hybrides [86].

problème de routage de véhicules avec capacités. A chaque étape, une recherche tabou est utilisée pour extraire l'ensemble des contraintes de capacité qui sont violées par le problème relaxé à la base desquelles les coupes du *banch and cut* sont construites. Martin et Otto [124] ont incorporé une recherche locale dans un algorithme de recuit simulé pour résoudre le problème du voyageur de commerce et celui de la partition de graphes. Leur méthode a surpassé les algorithmes de recherche locale traditionnels. Les méthodes de recherche à voisinage large (LNS) [7] sont également des exemples typiques de méthode BNR. Ces algorithmes peuvent être vus comme des algorithmes de recherche locale dans lesquels un voisinage assez vaste est utilisé. Ce voisinage est parcouru en utilisant une méthode heuristique ou exacte. Cette méthode est notamment utilisée dans [21] et [155] et est aussi très connue sous le nom de POPMUSIC pour *Partial Optimization Metaheuristic Decomposition Search* [170].

— **Hybridation de bas niveau collaborative (BNC) :** Cette classe regroupe les hybridations dans lesquelles un élément d'une méthode donnée est remplacé par une autre méthode. La méthode incorporée doit pouvoir être exécutée en parallèle de la méthode globale. La méthode proposée par Cotta et al. [34] est un exemple d'hybridation BNC entre un algorithme génétique et la méthode de *branch and bound*. La méthode de *branch and bound* est utilisée pour construire un opérateur de croisement intelligent qui construit l'individu possédant la combinaison optimale des gènes des parents. Fleurent et Ferland [48] remplacent l'opérateur de mutation d'un algorithme génétique par une recherche locale ou une recherche tabou pour résoudre les problèmes de partition de graphes. Salcedo-Sanz et al [149] proposent une méthode qui améliore les individus d'un algorithme génétique en utilisant un réseaux de neurones avant d'appliquer les opérateurs génétiques.

— **Hybridation haut niveau à relais (HNR) :** On parle d'hybridation de haut niveau à relais lorsque des métaheuristiques complètes sont exécutées séquentiellement. Ce schéma de coopération est le plus représenté dans la littérature. Souvent une métaheuristique est utilisée avant l'exécution d'une méthode exacte afin d'avoir initialement connaissance d'une borne de qualité. Une heuristique peut également être utilisée pour générer une population de solutions de qualité qui sera utilisée comme population initiale dans une métaheuristique à population [112]. La méthode proposée dans [10] est une hybridation haut niveau à relais entre une recherche locale et une méthode exacte. La recherche locale est utilisée pour générer un ensemble de

solutions de bonne qualité. Un espace de recherche restreint est défini en considérant l'ensemble des arêtes que l'on retrouve dans toutes les solutions générées par la recherche locale. Puis la méthode exacte est utilisée pour chercher la meilleur solution existant dans cet espace restreint.

— **Hybridation haut niveau collaboratives :** Cette classe contient les algorithmes qui combinent deux méthodes qui ne s'imbriquent pas l'une dans l'autre et qui peuvent être exécutées en parallèle. Les collaborations de cette classe consistent souvent en plusieurs méthodes travaillant sur le même problème mais éventuellement dans des parties différentes de l'espace de recherche et échangeant des informations. Par exemple pour résoudre des problèmes de partitionnement de graphe, Hammami et Ghédira [66] font coopérer une recherche locale et un recuit simulé, les deux méthodes fonctionnent en parallèle et échangent des informations à intervalles réguliers concernant la meilleure solution trouvée ainsi que des pénalités infligées aux éléments de solution inintéressants. Cette méthode donne de meilleurs performances que l'exécution séparée des deux métaheuristiques. La méthode proposée en [133] fonctionne sur le même principe mais implique la coopération entre une méthode de recuit simulé et une méthode de *branch and bound*.

Classification générale

La classification générale proposée par Talbi [174] comporte 3 critères dichotomiques de classification :

— **Homogènes/Hétérogènes :** Une méthode hybride est dite homogène si les métaheuristiques combinées sont identiques, sinon elle est dite hétérogène. Toutes les méthodes hybrides citées précédemment sont des exemples de méthodes hybrides hétérogènes. Dans [178] une méthode collaborative de haut niveau homogène combine plusieurs algorithmes génétiques qui travaillent en parallèle sur de petites sous-populations. À intervalles réguliers des migrations sont effectuées entre les sous-populations permettant des échanges d'informations.

— **Globales/Partielles :** Dans une hybridation globale toutes les méthodes en jeu agissent sur l'ensemble de l'espace de recherche alors qu'une hybridation partielle découpe le problème en sous problèmes ayant des espaces de définitions différents. Toutes les méthodes citées précédemment dans la classe des hybridations de haut niveau collaboratives sont des méthodes globales. Comme exemple de méthode partielle on peut citer [169], dans laquelle le problème de routage de véhicules est divisés en sous problèmes (définis en divisant l'ensemble des villes à visiter en secteurs) résolus avec de la recherche tabou.

— **Généralistes/Spécialistes :** Les hybridations générales sont celles où toutes les méthodes résolvent le même problème d'optimisation. A l'inverse les hybridations spécialistes sont celles où chaque méthode impliquée résout un problème d'optimisation différent. Toutes les hybridations citées précédemment sont des hybridations générales excepté [12] qui est une hybridation spécialiste. Une autre approche d'hybridation spécialiste consiste à utiliser une heuristique pour optimiser la valeur des paramètres d'une autre heuristique. Par exemple, Krueger [98] optimise les paramètres d'un recuit simulé à l'aide d'un algorithme génétique, dans [2] ceux d'un

algorithme de colonie de fourmis sont optimisés grâce à un algorithme génétique et Shahookar et Mazumder [154] optimisent les paramètres d'un algorithme génétique à l'aide d'un autre algorithme génétique.

4.2 Hybridation entre métaheuristiques et programmation dynamique

Dans cette section, nous allons présenter un état de l'art des hybridations impliquant métaheuristique et programmation dynamique. Ces méthodes seront classifiées en utilisant les taxinomies précédemment présentées. Ces méthodes sont par nature toutes hétérogènes.

Les hybridations bas niveau à relais :

Les méthodes de ce type combinant métaheuristiques et programmation dynamique sont des méthodes de recherche à voisinage large [7].

La plus répandue est la Programmation Dynamique Discrète Différentielle (DDDP) qui a été proposée en 1971 par Heidari pour résoudre des problèmes de planification sur des réservoirs de grande capacité [70]. Cette méthode s'applique aux problèmes possédant la propriété de Bellman et pouvant être résolus par une approche graduelle pouvant être découpées en n étapes. A chacune de ces étapes correspond un état qui est défini par m variables d'états. L'idée de DDDP est d'utiliser une recherche locale en considérant une solution comme une séquence d'états $(S_i)_{i=1...n}$. Pour définir le voisinage d'une solution, on considère un ensemble de vecteurs m-dimensionnels :

$$\Delta s_j = (\delta s_{j,1}, \delta s_{j,2}, ..., \delta s_{j,m}) \quad j = 1, 2, ..., T^m$$

Où T est un nombre fixé de possibilités de changement pour une variable d'état. A partir des ensembles Δs_j on obtient un sous-espace de recherche défini autour de la solution initiale par :

$$\{S_j + \Delta s_i, \quad i = 1, 2, ..., T^m, j = 1, 2, ..., n\}$$

Cet espace définit un voisinage large dans lequel on utilise la programmation dynamique pour trouver le meilleur voisin.

La méthode du corridor proposée par Sniedovich et Voss [161] généralise la méthode DDDP pour l'hybridation de la recherche locale avec n'importe quelle méthode exacte. Ainsi dans [161], des règles générales sont données sur le choix de la représentation et du voisinage associé permettant de définir des sous-problèmes solvables en des temps raisonnables par la méthode exacte choisie. [108, 179] donnent deux exemples d'applications de la DDDP. La DDDP est une hybridation partielle générale entre la méthode de recherche locale et la programmation dynamique.

Une autre collaboration entre la programmation dynamique et la recherche locale a été proposée par Congram et Potts [33]. Cette méthode, appelée Dynasearch, fonctionne globalement comme un algorithme de recherche locale classique. Cependant dans la recherche locale classique le voisinage d'une solution peut être défini comme l'ensemble des solutions que l'on peut obtenir en appliquant une transformation locale sur la solution initiale (appelée mouvement). Dans Dynasearch on définit le voisinage comme l'ensemble des solutions obtenues après un nombre n de mouvements. la meilleure séquence de n mouvements (le

meilleur voisin) est trouvée en utilisant la programmation dynamique. Des exemples d'applications de Dynasearch sont donnés dans [9, 109, 162]. Dynasearch est une hybridation globale générale.

Les hybridations bas niveau collaboratives :

Une méthode hybride de bas niveau collaborative a été proposée pour résoudre les problèmes d'ordonnancement [130]. La programmation dynamique est incorporée dans un algorithme génétique pour construire un opérateur de croisement intelligent. Le principe de ce croisement est de trouver un ordre partiel commun aux deux individus à croiser, l'individu enfant est construit en cherchant la meilleur solution consistante avec cet ordre partiel en utilisant la programmation dynamique. L'ordre partiel est défini comme ceci :

$$D = \{(i,j) | \sigma_1^{-1}(i) \leq \sigma_1^{-1}(j) \ et \ \sigma_2^{-1}(i) \leq \sigma_2^{-1}(j)\}$$

Où $\sigma_1^{-1}(i)$ retourne l'emplacement de l'élément i dans la solution fournie par le premier parent (respectivement $\sigma_2^{-1}(i)$ pour le second parent). Une mutation peut être appliquée sur D avant d'appliquer la programmation dynamique dans le but d'élargir l'espace de recherche. Cette méthode est une hybridation partielle générale dans la classification générale.

Les hybridations haut niveau à relais :

L'hybridation proposée par Tosporsampan et al. en 2005 est une hybridation haut niveau à relais, globale et générale impliquant un algorithme génétique et de la DDDP [182]. L'idée est d'utiliser la solution trouvée par l'algorithme génétique comme solution de départ de la programmation dynamique discrète différentielle.

L'hybridation proposée par Dunker et al. [43] est également une hybridation haut niveau à relais mais celle ci est de type spécialiste. L'objectif est de résoudre un problème dynamique d'aménagement optimal des ateliers dans une usine. L'aménagement est dit dynamique car l'on considère T périodes de temps de sorte que l'aménagement puisse changer d'un période à l'autre de façon à être adapté aux changements de production. Il faut minimiser les temps de déplacements du personnel entre les ateliers lors de la production et les temps de ré-aménagement des ateliers. Dans un premier temps des algorithmes génétiques sont utilisés pour résoudre le problème du meilleur aménagement pour chaque période de temps t, chacun de ces algorithmes fournit une population de solutions P_t. Ensuite la programmation dynamique est utilisée pour sélectionner une solution p_t dans chaque population P_T de sorte que le planning d'aménagement/ré-aménagement obtenu soit le meilleur.

Dans [136], la programmation dynamique est utilisée avec un réseau de neurones pour résoudre le problème d'affectation d'unités (*Unit Commitment Problem*, décrit dans la suite). Tout d'abord le réseau de neurones est utilisé pour calculer un planning dans lequel la programmation de certaines unités peut être incertaine sur certaines périodes du planning. La programmation dynamique est ensuite utilisée pour fixer les incertitudes du planning.

Les hybridations haut niveau collaboratives :

L'hybridation proposée en [140] peut se classifier comme une hybridation haut niveau collaborative, globale et générale. Une hybridation entre un algorithme génétique et la programmation dynamique est utilisée pour résoudre un problème de développement optimal

de production sur du long terme. L'idée de cet algorithme est, d'appliquer la programmation dynamique pour trouver la meilleure solution dans un espace de recherche restreint défini à partir des meilleurs solutions de la population de l'algorithme génétique puis d'introduire la solution obtenue dans la population. Cette procédure a lieu à chaque fois que l'algorithme génétique arrive à convergence.

On peut noter qu'il y a relativement peu de travaux proposant des hybridations entre programmation dynamique et métaheuristiques. Cependant les travaux cités précédemment montrent que ces méthodes hybrides permettent d'obtenir de très bons résultats. En particulier DDDP et Dynasearch ont été utilisées avec succès autant sur des problèmes pratiques que théoriques.

5 Conclusion

Dans ce chapitre, nous avons tout d'abord défini ce que nous entendons par problème d'optimisation combinatoire. Nous avons vu qu'il existe deux types d'approches pour aborder ce type de problèmes. D'une part les méthodes qui garantissent l'optimalité de la solution qu'elles retournent, ces méthodes sont appelées méthodes exactes. D'autre part les heuristiques et métaheuristiques qui permettent d'obtenir de bons résultats rapidement mais qui ne garantissent pas que l'optimum ait été trouvé.

Pour chacune de ces approches nous avons dressé un état de l'art ce qui nous permet d'avoir une vue d'ensemble claire des principales méthodes de résolutions existantes.

Deux approches nous intéressent plus particulièrement, car elles sont à la base de tout le travail effectué dans cette thèse. Il s'agit de la programmation dynamique, une méthode d'optimisation exacte et de l'algorithme génétique, une métaheuristique. Dans ce chapitre nous avons donc expliqué ces deux méthodes de façon très détaillée et nous avons introduit tout le vocabulaire qui leur est associé et qui nous sera utile dans la suite de ce manuscrit.

Nous nous sommes ensuite intéressés aux techniques d'hybridations permettant de combiner plusieurs approches de résolutions avec pour objectif d'obtenir une nouvelle approche plus efficace ou plus rapide. Dans ce but deux taxinomies ont été présentées, elles permettent de comprendre quels sont les différents moyens d'hybridation possibles et comment ceux ci ont été exploités dans la littérature scientifique.

Puis, en utilisant les taxinomies présentées nous avons dressé un état de l'art des méthodes d'hybridation existantes combinant une métaheuristque avec de la programmation dynamique. Ce travail nous permettra plus tard de situer nos travaux par rapport aux méthodes déjà existantes et validées par la communauté scientifique.

Les connaissances apportées dans ce chapitre représentent le fondement théorique des contributions qui seront présentées lors des prochains chapitres. Les notions et le vocabulaire ayant été introduits dans ce chapitre seront nécessaires à la bonne compréhension de nos travaux. De plus l'état de l'art qui a été dressé permet d'appréhender l'ensemble des méthodes de résolution déjà existantes et donc fournit le bagage de connaissance nécessaire pour situer les propositions qui seront faites.

Chapitre 2

Problèmes d'optimisation combinatoires, cadres applicatifs

Ce second chapitre introduit les deux problèmes d'application qui seront étudiés dans cette thèse. Le premier, est un problème très académique, il s'agit du problème d'affectation d'unités plus connu sous son nom anglais Unit Commitment Problem. Le second est un problème issu d'un cas industriel réel qui consiste à chercher une planification des débits d'eau à travers les conduites et turbines d'un réseau hydro-électrique qui soit optimale en terme de profit.

Plus précisément, dans ce chapitre, nous donnons, pour chacun de ces problèmes :

- *sa formalisation mathématique ;*
- *une étude de sa complexité théorique ;*
- *un état de l'art des méthodes de résolution simples et composées lui ayant été appliquées dans la littérature.*

1 Introduction

Dans cette thèse nous nous focalisons sur deux problèmes de production d'énergie. L'un est un problème académique très étudié sous le nom de *Unit Commitment Problem* que nous traduirons par "Problème d'affectation d'unités". L'autre est un problème issu du monde industriel, qui concerne la planification de l'utilisation des eaux dans un réseau hydro-électrique.

Dans ce chapitre nous présentons chacun de ces problèmes en commençant par le problème d'affectation d'unités. Nous commençons à chaque fois par énoncer clairement le problème à l'aide d'une modélisation mathématique. Nous faisons ensuite une étude de sa complexité, afin d'expliquer en quoi il est difficile d'aborder ce problème avec des méthodes exactes mais également avec les métaheuristiques classiques. Puis nous dressons un état de l'art sur les méthodologies qui ont déjà été proposées pour résoudre le problème présenté ou des problèmes similaires.

2 Problème d'affectation d'unités

Dans cette partie nous allons présenter le problème d'affectation d'unités plus connu sous son nom anglais : The Unit Commitment Problem (UCP).

2.1 Énoncé du problème

Le problème de *unit commitment* ou d'affectation d'unités est un problème d'optimisation mixte de la littérature classique. Sachant qu'un système de production d'énergie

électrique doit, pour chaque période de temps d'un horizon de planification, fournir une quantité d'énergie suffisante pour répondre à une demande de consommateurs. L'objectif de ce problème consiste à faire un choix stratégique sur l'état de marche/arrêt et les quantités d'énergie produites par l'ensemble des unités de production d'électricité fonctionnant en parallèle. Cependant, cette production engendre des coûts financiers importants, il faut donc minimiser les coûts de production tout en satisfaisant la demande et en respectant un ensemble de contraintes techniques. Parmi les coûts de production, il y a le coût de carburant nécessaire pour produire l'énergie et le coût d'allumage des unités. De plus, des contraintes techniques spécifiques à chaque unité de production sont à respecter : A une heure donnée une unité doit, si elle est allumée, produire une quantité minimale d'énergie sans dépasser la quantité maximale qu'elle peut produire. En outre, lorsqu'une unité est allumée ou éteinte, celle-ci doit le rester pendant une durée minimale. A cela s'ajoute une contrainte supplémentaire, les unités allumées doivent être capables de produire une réserve d'énergie en plus de la demande pour prévenir des cas où une des unités tomberait en panne ou des cas où la demande serait supérieure à la prévision.

2.2 Modélisation mathématique

Le modèle mathématique du problème d'affectation d'unités décrit un problème d'optimisation combinatoire mixte avec un ensemble de contraintes linéaires mais une fonction objectif non linéaire. Cette fonction objectif est à minimiser, elle représente le coût de production totale de l'énergie produite tout au long de l'horizon de planification. Dans la suite, nous détaillons d'abord la fonction objectif puis les contraintes du problème.

Fonction objectif à minimiser :

Pour une planification optimale de N unités de production, sur une période de T heures, on peut décrire la fonction objectif de la façon suivante :

$$f_{obj} = \sum_{i=1}^{T}\sum_{i=1}^{N}[CF_i(p_{i,t})u_{i,t} + CS_i(T_{i,t-1}^{off})u_{i,t}(1-u_{i,t-1})] \tag{2.1}$$

Avec les notations suivantes :

- $u_{i,t} \in \{0,1\}$: variable binaire spécifiant si l'unité $i \in \{1,...,N\}$ est en état de marche ou d'arrêt à l'instant t ;
- $p_{i,t}$: variable à valeur réelle donnant la quantité d'énergie produite par l'unité i à l'instant t ;
- $T_{i,t}^{off}$: temps depuis lequel l'unité i est restée éteinte à l'instant t ;
- $CS_i(T_{i,t}^{off})$: coût d'allumage de l'unité i ;
- $CF_i(p)$: coût du carburant nécessaire à la production de la quantité p d'énergie avec l'unité i.

Le coût du carburant $CF_i(p)$ est défini par une fonction quadratique :

$$CF_i(p) = a_{0i} + a_{1i}p + a_{2i}p^2 \tag{2.2}$$

Où a_{0i}, a_{1i}, a_{2i} sont des constantes spécifiques à l'unité i.

Le coût de démarrage est une fonction constante par morceaux définie comme ceci :

$$CS_i(T^{off}_{i,t-1}) = \begin{cases} CS_{cold,i} & si \quad T^{off}_{i,min} + T_{cs,i} \leq T^{off}_{i,t-1} \\ CS_{hot,i} & sinon \end{cases} \tag{2.3}$$

Où :

— $T^{off}_{i,min} + T_{cs,i}$: définit le temps qu'il faut à l'unité de production i pour se refroidir ;
— $CS_{cold,i}$: Coût de démarrage à froid de l'unité i ;

— $CS_{hot,i}$: Coût de démarrage à chaud de l'unité i.

Contraintes :

— Les capacités de production d'une unité sont limitées ; une unité allumée produit au minimum une certaine quantité et ne peut produire au delà d'une autre quantité d'énergie :

$$u_{i,t} P^{min}_i \leq p_{i,t} \leq u_{i,t} P^{max}_i \tag{2.4}$$

P^{min}_i : quantité minimum d'énergie que produit l'unité i ;
P^{max}_i : quantité maximum d'énergie que peut produire l'unité i.

— L'ensemble des unités de production allumées doit pouvoir fournir une quantité d'énergie suffisante pour satisfaire la demande d_t et une réserve r_t dans le cas où l'une des unités tomberait en panne :

$$\sum_{i=1}^{N} u_{i,t} \times p_{i,t} = d_t \tag{2.5}$$

$$\sum_{i=1}^{N} u_{i,t} \times P^{max}_i \geq d_t + r_t \tag{2.6}$$

d_t : Quantité d'énergie correspondant à la demande à l'instant t.
r_t : Quantité d'énergie correspondant à la réserve à l'instant t.

— Lorsqu'une unité est en marche ou en arrêt, celle-ci doit rester dans le même état pendant une durée minimale :

$$\begin{aligned} T^{on}_{i,t} &\geq T^{on}_{min,i}(1 - u_{i,t}) u_{i,t-1} \\ T^{off}_{i,t} &\geq T^{off}_{min,i}(1 - u_{i,t-1}) u_{i,t} \end{aligned} \tag{2.7}$$

$T^{on}_{min,i}$: durée minimum de marche lorsque l'unité i est allumée ;
$T^{off}_{min,i}$: durée minimum d'arrêt lorsque l'unité i est éteinte ;
$T^{off}_{i,t}$: temps depuis lequel l'unité i est éteinte à l'instant t ;
$T^{on}_{i,t}$: temps depuis lequel l'unité i est allumée à l'instant t.

— $T_{i,t}^{on}$ et $T_{i,t}^{off}$ sont définis comme ceci :

$$T_{i,t}^{on} = \begin{cases} T_{i,ini}^{on} & si \quad t = 0 \\ (T_{i,t-1}^{on} + 1)u_{i,t} & sinon \end{cases} \tag{2.8}$$

$T_{i,ini}^{on}$: Temps depuis lequel l'unité i est allumée avant le début de la planification.

$$T_{i,t}^{off} = \begin{cases} T_{i,ini}^{off} & si \quad t = 0 \\ (T_{i,t-1}^{off} + 1)(1 - u_{i,t}) & sinon \end{cases} \tag{2.9}$$

$T_{i,ini}^{off}$: Temps depuis lequel l'unité i est éteinte avant le début de la planification.

2.3 Analyse et complexité

Le problème décrit précédemment est un problème d'optimisation combinatoire consistant à minimiser une fonction sur un domaine discret. Bien que l'on puisse décrire ce domaine de façon concise, celui-ci reste de très grande taille. En effet, dans le problème d'affectation d'unités, pour une planification de N unités, sur une période de T heures, le nombre de solutions potentielles au problème est de $2^{N \times T}$. La non linéarité de la fonction objective amplifie également la difficulté du problème en rendant l'application de toutes les méthodologies basées sur la programmation linéaire impossible. S'ajoute à cela la forte dépendance entre les variables de décision binaires. Elle est due d'une part aux contraintes de temps minimal d'arrêt et de marche et d'autre part à la fonction de coût de démarrage qui dépend du temps durant lequel une unité est restée éteinte. Cette dépendance a un effet négatif car elle complique le parcours de l'espace des solutions réalisables et entraîne que des solutions proches peuvent avoir des évaluations très éloignées. Il a été montré que le problème d'affectation d'unités est NP-complet [63]. Ceci veut dire qu'il n'existe pas de solution algorithmique permettant de le résoudre de façon optimale en un temps polynomial par rapport à la taille des données (à moins que la classe de problèmes P soit égale à la classe des problèmes NP). La complexité du problème d'affectation d'unités a suscité l'intérêt de beaucoup de mathématiciens et informaticiens de la fin du 20^{eme} siècle et l'on peut donc trouver de nombreux travaux traitant du problème de sa résolution dans la littérature scientifique. Dans la prochaine partie nous présentons un état de l'art de ces publications.

2.4 État de l'art des méthodes appliquées à la résolution du problème d'affectation d'unités

Le problème d'affectation d'unités est un problème qui a été très étudié. Dans cette partie nous allons passer en revue les différentes méthodes ayant été proposées. Cet état de l'art est structuré en deux parties. Dans un premier temps nous présentons les méthodes d'optimisation classiques ayant été appliquées au problème puis nous présentons les méthodes hybrides.

Méthodes classiques :

Les travaux les plus anciens sur le problème d'affectation d'unités remontent à 1966 et consistent simplement en des méthodes d'énumération [68, 92]. Cependant ces méthodes d'énumérations sont très limitées par la taille des données. En effet le nombre de solutions possibles augmente exponentiellement avec le nombre d'unités et il serait donc trop coûteux en terme de temps de calcul d'utiliser cette approche.

Une méthode qui à l'inverse permet d'obtenir des solutions réalisables très rapidement est la méthode de liste de priorité (en anglais *priority listing*). Elle consiste à établir une classification des unités basée sur leurs coûts de production (la liste de priorités) puis d'affecter ces unités, itérativement sur chaque periode de temps, en les considérant dans l'ordre de cette liste et en respectant les contraintes [25, 106, 157]. Malheureusement ces méthodes ne permettent pas d'atteindre l'optimum et il n'existe pas de garantie de performance connue.

Les méthodes basées sur le *branch and bound* [32, 75, 102], la relaxation Lagangienne [134, 171, 191] et la méthode des points intérieurs [115] permettent d'obtenir des solutions très précises. En particulier la méthode proposée par Viana [190] permet de fournir des solutions pour des instances ayant jusque 100 unités. Cependant les temps de calcul de ces algorithmes augmentent exponentiellement avec la taille des données.

La programmation dynamique est très utilisée pour le problème d'affectation d'unités [72, 138, 166]. Cette méthode pourrait théoriquement fournir une solution exacte car elle a l'avantage de pouvoir résoudre les problèmes ayant une fonction objective non-convexe comme c'est le cas pour le problème d'affectation d'unités. Néanmoins comme l'espace d'états grandit exponentiellement avec le nombre d'unités celui-ci est en général restreint et l'optimalité de la solution obtenue n'est plus garantie.

Les métaheuristiques ont été très utilisées car elles ne sont pas affectées par la *"malédiction de la dimensionnalité"* qui fait que les algorithmes basés sur des méthodes exactes voient leurs temps de calcul augmenter exponentiellement avec le nombre d'unités. De plus, leurs applications ont donné des résultats très proches de ceux obtenus par les algorithmes basés sur des méthodes de résolution exactes. Parmi elles, la recherche tabou [117], le recuit simulé [116, 159, 205], le réseau de neurones artificiels [151], les essaims de particules [53, 201, 204], la colonie de fourmis [158, 160, 167], l'algorithme gravitationnel (une métaheuristique à population de solutions similaire à l'algorithme à essaims de particules sinon que les interactions entre les solutions sont inspirées par les lois gravitationnelles de Newton) [147] et les algorithmes génétiques [36, 89, 152, 168] ont été utilisées. Plus récemment de nouvelles métaheuristiques inspirées du calcul quantique ont été développées et appliquées au problème d'affectation [83, 84, 101], elles ont donné des résultats qui sont très prometteurs sur ce problème.

Méthodes hybrides :

En réalité presque toutes les métaheuristiques citées précédemment utilisent une représentation indirecte qui peut être considérée comme un procédé d'hybridation de bas niveau. Il faut noter que le problème de distribution de la production de la demande entre les unités allumées (fixation des $p_{i,t}$ connaissant les valeurs des $u_{i,t}$) est un problème simple car c'est un problème d'optimisation continu et convexe. Les métaheuristiques proposées dans la littérature tirent en effet partie de cette propriété. Elles utilisent un codage indirect des solutions et ne fixent que les valeurs des variables binaires $u_{i,t}$ (donc ne s'intéressent qu'au problème de planification des temps de marche et d'arrêt) puis utilisent une méthode exacte pour résoudre le problème de distribution et obtenir la solution complète. La Figure 2.1 présente cette stratégie.

Dans cette partie nous nous intéresserons aux métaheuristiques qui utilisent en plus un autre procédé d'hybridation.

De nombreuses méthodes hybrides haut niveau à relais ont été proposées : dans [30] un algorithme de recuit simulé et un algorithme génétique sont appliqués successivement jusqu'à atteindre un critère d'arrêt. L'algorithme proposé dans [144] applique successive-

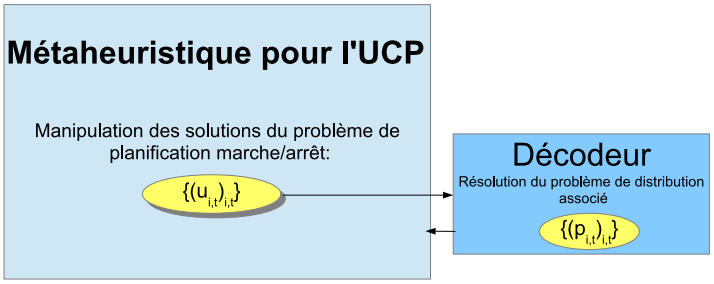

FIGURE 2.1 – Les métaheuristiques utilisant une représentation indirecte pour le problème d'affectation d'unités.

ment un algorithme de recuit simulé et une recherche locale. La méthode présentée en [146] hybride également un algorithme génétique avec un algorithme de recherche locale. L'algorithme de Ouyana et al. [136] est une hybridation de haut niveau impliquant un réseau de neurones et la programmation dynamique. L'algorithme de Trivedi et al. [183] hybride un algorithme génétique avec un algorithme évolutionnaire différentiel. Rahman et al. [145] combinent des métaheuristiques avec des méthodes de programmation linéaire entière pour créer un algorithme fournissant des solutions exactes à un ϵ prêt en un temps de calcul moindre que ceux des méthodes exactes traditionnelles.

Au niveau des hybridations bas niveau, il y a tout d'abord les algorithme mémétiques qui ont été appliqués au problème d'affectation d'unités [146, 186]. Ce sont des algorithmes génétiques dans lesquels la mutation est remplacée par une recherche locale. Chen [29] propose un algorithme appelé ES-PSO qui combine un système expert (un algorithme issu de l'intelligence artificielle tentant de reproduire les procédés cognitifs d'un expert du domaine associé au problème) et un algorithme à essaim de particules. Il fonctionne en deux étapes. Dans un premier temps le système expert est utilisé pour générer un essaim de particules initial de qualité. Puis un algorithme à essaim de particules combiné à un système expert est appliqué. Le système expert est utilisé, à chaque itération, pour guider le choix des nouvelles positions des particules. La méthode proposée par Balci et al [15] est une relaxation lagrangienne utilisant un essaim particulaire pour trouver les coefficients lagrangiens optimaux.

3 Problème de planification des eaux dans un réseau hydro-électrique

Ce problème est basé sur un cas d'application réel dont les données ont été fournies par une entreprise (GDF). Il s'agit du problème de planification des eaux dans un réseau hydro-électrique. Dans cette section nous allons présenter en détails ce problème, démontrer qu'il est NP-Complet puis faire un état de l'art sur les problèmes similaires ayant été traités dans la littérature.

3.1 Énoncé du problème

Le problème de planification des eaux dans un réseau hydro-électrique, en anglais *Hydro Scheduling Problem*, a pour objectif de maximiser les bénéfices obtenus en produisant de l'électricité avec un réseau hydro-électrique. Un réseau hydro-électrique est un système

composé de réservoirs, de turbines et de liaisons reliant des réservoirs et/ou des turbines. La Figure 2.2 illustre deux réseaux hydro-électriques.

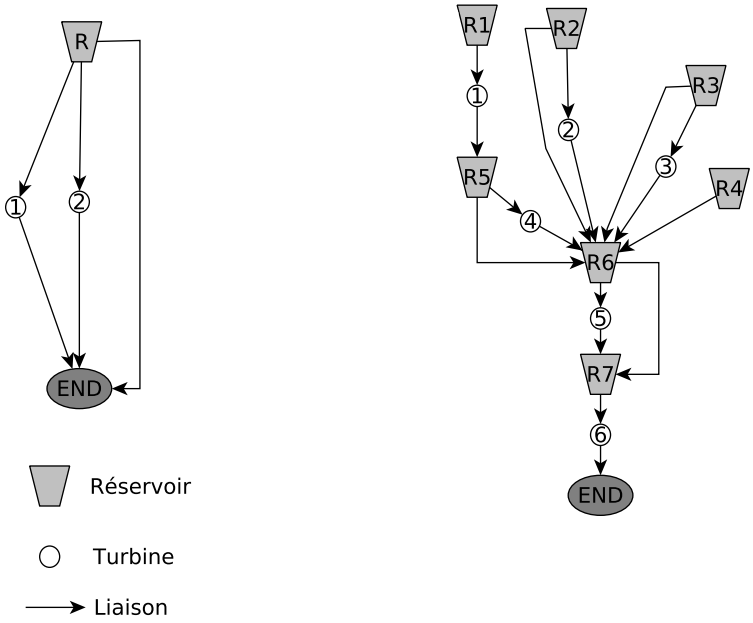

FIGURE 2.2 – Réseaux hydro-électrique 1 et 2. Les réservoirs sont représentés par des trapèzes, les turbines par des cercles et les liaisons par des arcs.

L'objectif du problème est de proposer une planification des débits d'eau parcourant chaque liaison du système pour chaque période de temps d'un horizon de planification de façon à maximiser le profit. Le profit correspond au prix de vente de l'électricité multiplié par la quantité d'électricité produite.

Le prix de vente de l'électricité varie avec le temps, il peut aussi varier en fonction de la turbine ayant produit l'électricité. En effet, l'énergie engendrée par certaines turbines se vend au prix spot, tandis que celles d'autres turbines se vend à l'obligation d'achat. Le prix spot correspond aux prix de l'électricité dans un marché au comptant, il peut donc varier très rapidement et être très différent d'une période de temps à l'autre. L'obligation d'achat correspond à un prix de vente qui a été fixé par la loi afin d'encourager la production d'énergie renouvelable. Ce prix est totalement déterminé, il est différent selon que l'on soit en hiver ou en été et dépend du moment où la vente a lieu (en heure de pointe en heure pleine ou en heure creuse).

La production d'énergie générée par une turbine est une fonction très complexe du débit d'eau traversant la turbine ainsi que de la hauteur de la chute que subit l'eau avant d'atteindre la turbine.

Le planning proposé doit respecter un certains nombre de contraintes. Tout d'abord le volume d'eau contenu dans un réservoir ne doit pas dépasser la capacité maximale du réservoir. Un volume minimal doit également être respecté pour chaque réservoir, ceci afin de garantir une certaine réserve en eau. Cette limite minimale peut varier au cours du temps, elle est par exemple plus forte en période de sécheresse. De plus, un débit minimal est à respecter sur certaines liaisons qui correspond à un écoulement naturel des eaux et il n'est pas possible de dépasser un certain débit maximal. Il faut aussi prendre en compte

le fait qu'il n'est pas possible d'utiliser une turbine si c'est pour produire moins qu'une quantité minimale d'énergie ou pour produire plus qu'une quantité maximale d'énergie. Une dernière contrainte est que l'on souhaite qu'à la fin de l'horizon de planification le système retrouve son état initial, c'est-à-dire que les contenus des réservoirs en fin de la planification soit égal à leurs contenus initiaux.

Dans la prochaine partie le problème est expliqué de façon plus formelle en utilisant une modélisation mathématique.

3.2 Modélisation mathématique

Le problème de planification de l'utilisation des eaux dans un réseau hydro-électrique est modélisé comme un problème linéaire à variables mixtes. Nous allons dans cette partie décrire ce modèle en commençant par expliquer la fonction objectif en détails.

Fonction objectif à maximiser :

L'objectif est de trouver une planification des débits dans un réseau hydro-électrique qui maximise le bénéfice annuel. Par commodité on distingue trois types de variables de décision qui vont nous permettre de définir un planning solution optimal :

- $r_{j,t}$: quantité débitée à travers une liaison j reliant deux réservoirs sans passer par une turbine au temps t ;
- $q_{j,t}$: quantité d'eau déversée pour la turbine j au temps t ;
- $V_{i,t}$: quantité d'eau présente dans le réservoir i au temps t.

Cependant les valeurs des $V_{i,t}$ sont entièrement déterminées par celles des $q_{j,t}$ et des $r_{j,t}$.

La fonction à optimiser correspond au profit :

$$profit = \sum_{t=0...T-1} \sum_{i=1...N} \sum_{j \in \delta_i^-} prix_{t,j} \times prod_j(q_{j,t}, V_{i,t}),$$

T correspond au nombre de périodes temporelles considérées pour la planification, N est le nombre de réservoirs et δ_i^- est l'ensemble des liaisons sortant du réservoir i. $prix_{t,j}$ est le prix de vente de l'énergie produite par la turbine j au temps t, ce prix peut correspondre au prix spot ou bien à une obligation d'achat en fonction des turbines. La fonction de production $prod_j$ d'une turbine j dépend non seulement de la quantité $q_{j,t}$ débitée au cours de la période t mais aussi de la quantité d'eau, $V_{i,t}$, dans le réservoir en aval. Cette fonction est très irrégulière, elle n'est ni linéaire ni convexe par rapport aux variables de décision. Cependant une modélisation de la production sous forme de fonction affine par morceaux, qui nous a été fournie par l'entreprise, a été utilisée. Elle s'exprime sous la forme suivante :

$$prod_j(q, V) = \sum_{l \in I_j^V} \sum_{k \in I_j^q} (a_{j,k,l} \times (min(B_{j,k+1}, q) - B_{j,k}) 1_{V \in l} \times 1_{q \geq B_{j,k}}),$$

Avec :

- I_j^V l'ensemble des intervalles de définition de la fonction $prod_j$ par rapport au volume d'eau V contenu dans le réservoir en aval ;
- I_j^q l'ensemble des intervalles de définition de la fonction $prod_j$ par rapport au débit q ;
- $B_{j,k}$ est la borne inférieure du $k^{\text{ième}}$ intervalle de définition de la fonction de production par rapport au débit q ;

— $a_{j,k,l}$ est un coefficient constant rattaché à l'intervalle de définition k par rapport au débit et l par rapport au volume. Il est tel que la fonction est croissante vis à vis de chaque variable de décision.

Il est possible d'exprimer, la fonction de production comme une fonction linéaire en utilisant les variables de décision suivantes :

— $b_{j,l,k,t}$: variable binaire valant 1 si la quantité d'eau dans le réservoir en aval de la liaison j appartient au $l^{\text{ième}}$ intervalle de I_j^V et si le débit dans la liaison liant le réservoir i à la turbine j est supérieur à $B_{j,k}$. Autrement dit $b_{i,j,l,k,t} = 1_{V \in l} \times 1_{q \geq B_{j,k}}$;
— $q_{j,l,k,t}$: variable réelle dont la valeur correspond à $(min(B_{j,k+1}, q) - B_{j,k}) 1_{V \in l} \times 1_{q \geq B_{j,k}}$.

Contraintes définissant les variables de décisions

Les contraintes permettant de définir les variables de décision et les liens existant entre elles sont exprimées ci dessous. Nous verrons ensuite comment les contraintes opérationnelles peuvent être modélisées.

1. Définition des variables $q_{j,t}$ (représentant la quantité d'eau déversée pour la turbine j au temps t) :

$$q_{j,t} = \sum_{l \in I_j^V} \sum_{k \in I_j^q} q_{j,l,k,t} \ \forall j, t$$

2. Définition des variables $V_{i,t}$ (représentant le volume d'eau dans le réservoir i au temps t) :

$$\begin{cases} V_{i,0} & = V_{i,init} \\ V_{i,t} & = V_{i,t-1} \quad + AN_{i,t-1} + \sum\limits_{j \in \delta_i^+} q_{j,t-1} + \sum\limits_{j \in \delta_i^+} r_{j,t-1} - \sum\limits_{j \in \delta_i^-} q_{j,t-1} - \sum\limits_{j \in \delta_i^-} r_{j,t-1} \ \forall t > 0 \end{cases}$$

Avec :

— $AN_{i,t}$: apports naturels dans le réservoir i au temps t. Ils dépendent de données météorologiques et sont, dans cette modélisation, considérés comme des constantes déterministes ;

— δ_i^+ : ensemble des liaisons entrant dans le réservoir i ;

— δ_i^- : ensemble des liaisons sortant du réservoir i ;

3. $b_{j,l,k,t} = 0$ implique $q_{j,l,k,t} = 0$:

$$q_{j,l,k,t} \leq (B_{j,k+1} - B_{j,k}) \times b_{j,l,k,t} \ \forall j, l, k, t$$

4. Définition des $b_{j,l,k,t}$:

— $b_{j,l,0,t}$ est nul si le volume d'eau dans le réservoir i en aval de la liaison j n'est pas dans l'intervalle $l = [l_1, l_2]$:

$$b_{j,l,0,t} \times l_1 \leq V_{i,t} \leq l_2 + V_{max,i} \times (1 - b_{j,l,0,t}) \ \forall j, l, t$$

où $V_{max,i}$ est la capacité maximale du réservoir i.

— $b_{j,l,k,t}$ est nul si $q_{j,l,k-1,t}$ n'a pas atteint la valeur $(B_{j,k} - B_{j,k-1})$:

$$q_{j,l,k-1,t} \geq (B_{j,k} - B_{j,k-1}) \times b_{j,l,k,t} \ \forall j, l, k, t$$

Contraintes opérationnelles :

Pour qu'un planning soit valide certaines contraintes opérationnelles doivent être respectées :

1. Contraintes de capacités minimales et maximales de stockage sur les réservoirs :

   $$V_{min,i,t} \leq V_{i,t} \leq V_{max,i} \ \forall i, t.$$

   La limite minimale dépend du temps étant donné que, par exemple, durant les périodes de sécheresse d'été, il peut être nécessaire de stocker plus d'eau.

2. Contraintes de débit minimal et maximal dans les liaisons :

   $$r_{min,i} \leq r_{i,t} \leq r_{max,i} \ \forall i, t,$$

3. Il n'est pas possible d'utiliser une turbine i pour produire moins qu'une production minimale :

   $$prodMin_j \times \sum_l b_{j,l,0,t} \leq \sum_{l \in I_j^V} \sum_{k \in I_j^q} (a_{j,k,l} \times q_{j,l,k,t}) \ \forall j, t$$

   Où j correspond au réservoir fournissant la turbine i.

4. Contraintes de débit maximal pour chaque turbine j :

   $$q_{j,t} \leq q_{max,j,l} \times b_{j,l,0,t} \forall j, l, t.$$

   Cette quantité maximale dépend de la hauteur d'eau dans le réservoir fournissant la turbine i, et donc de son contenu en eau. Elle peut correspondre à une production maximale ou à des règles de sécurité.

5. A la fin de l'horizon de planification la quantité d'eau présente dans chaque réservoir i doit être identique à la quantité qui y était présente initialement :

   $$V_{i,T} = V_{i,init}.$$

3.3 Analyse et complexité

Dans cette partie nous allons montrer que le problème de planification des eaux (HSP) tel qu'il a été formulé est un problème NP-complet. Dans la version décision du problème de planification des eaux un entier P est ajouté aux données et l'on cherche à répondre à la question suivante :

"Existe-t-il une planification réalisable des débits d'eau permettant d'obtenir un profit supérieur ou égal à P ?"

Il est facile de vérifier que ce problème appartient bien à la classe des problèmes NP. En effet, étant donné un planning, on peut aisément proposer un algorithme qui vérifie pour chaque période de temps du planning si les contraintes sont respectées et calcule le profit en un temps polynomial par rapport à la taille des données. On se concentrera ici sur la preuve de la NP-difficulté du problème. Pour cela, nous allons proposer une réduction

polynomiale entre le problème *Subset sum* qui a été montré être NP-Complet [**?**] et notre problème. *Subset sum* (SSP) peut être formulé sous la forme d'un problème de décision comme ceci :

— **Données :** un ensemble de K entiers $\{e_0, .., e_{K-1}\}$, tels que $e_0 \leq e_1 \leq ... \leq e_{K-1}$ et un entier s.

— **Question :** Existe-t-il un sous ensemble d'entiers de $\{e_0, .., e_{K-1}\}$ dont la somme fasse s ?

On propose la réduction polynomiale R suivante :

— nombre de réservoirs $(N) \leftarrow 1$

— nombre de liaisons $\leftarrow 2$

— nombre de turbines $\leftarrow 1$

— $V_{init} \leftarrow K^2 \times (e_K + 1) + s$

— $V_{max} \leftarrow K^2 \times (e_K + 1) + s$

— $V_{min} \leftarrow 0$

— $T \leftarrow K$

— $|I^V| \leftarrow K$

— $|I^q| \leftarrow 1$

— $r_{min} \leftarrow K(e_K + 1)$

— $r_{max} \leftarrow K(e_K + 1)$

— $AN_t \leftarrow 0 \ \forall t < T - 1$

— $AN_{T-1} \leftarrow K^2(e_K + 1) + s$

— $price_t \leftarrow 1 \ \forall t$

— $prodMin \leftarrow 1$

— $I_i^V \leftarrow [(K-i+1) \times K(e_K+1)+s-\sum_{k<i} e_k, (K-i+2) \times K(e_K+1)+s-\sum_{k<i-1} e_k[$

— $qmax_i \leftarrow e_i$

— $a_i \leftarrow 1/e_i$

— $P \leftarrow 0$

Montrons maintenant que la réponse au SSP est *"oui"* pour l'instance I si et seulement si la réponse au problème HSP est *"oui"* pour l'instance $R(I)$. Considérons le problème HSP sur l'instance $R(I)$. Le réseau associé est modélisé par la figure 2.3.

Étant donné que $r_{min} = r_{max} = r$ les débits sur la liaison L_2 sont complètement dé-

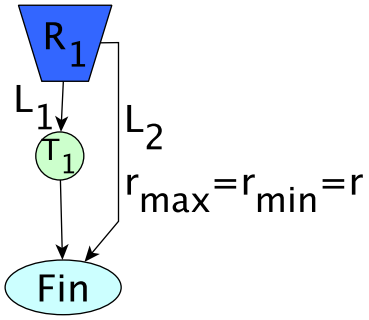

FIGURE 2.3 – Réseaux hydro-électrique obtenu par la réduction polynomiale R d'une instance I du problème SSP.

terminés. Le problème se restreint donc à planifier les débits sur L_1. Or, nous avons les propriétés suivantes :

— **au temps i les débits possibles dans L_1 sont soit 0 soit e_i :**
Montrons par récurrence qu'au temps i l'on peut choisir entre ne rien débiter pour la turbine T_1 ou débiter exactement la quantité e_i :
La propriété est vraie au temps 0 : Au temps initial 0 le réservoir contient la quantité $V_{init} = K^2 \times (e_K + 1) + s$ qui se trouve dans l'intervalle I_0^V. En effet l'intervalle I_0^V est défini par :

$$[K^2 \times (e_K + 1) + s, (K + 1) \times (e_K + 1) + s[= [V_{init}, V_{init} + K(e_K + 1)[.$$

Dans cet intervalle la quantité maximale que l'on peut débiter pour la turbine est $qmax_0 = e_0$. Comme a_0, le coefficient de production correspondant, vaut $1/e_0$ et que la production minimale est de 1, on ne peut utiliser la turbine avec un débit d'eau inférieur à e_0. On peut donc au temps 0 soit ne rien débiter pour la turbine soit débiter e_0.
Héritabilité : Supposons que la proposition soit vrai jusqu'au temps $i - 1$. Alors le volume contenu dans le réservoir au temps i appartient à l'intervalle I_i^V :

$$\begin{aligned}[(K - i + 1) \times K(e_K + 1) + s - \textstyle\sum_{k<i} e_k; (K - i + 2) \times K(e_K + 1) + s - \textstyle\sum_{k<i-1} e_k[\\ = [V_{ini} - (i - 1)r - \textstyle\sum_{k<i} e_k; V_{ini} - (i - 2)r - \textstyle\sum_{k<i-1} e_k[,\end{aligned}$$

avec $r = r_{max} = r_{min} = K(e_K + 1)$.
En effet à chaque heure le débit r est automatiquement débité par la liaison et il n'y a pas d'apport (à part durant la dernière heure) donc le volume dans le réservoir au temps i est :

$$V_{init} - (i - 1)r - \sum_{k<i} q_k,$$

où q_k est le débit dans la liaison L_1 au temps k. Or, soit $q_k = 0$ soit $q_k = e_k$. Donc le volume maximal possible est :

$$V_{init} - (i-1)r$$

et le volume minimal possible est :

$$V_{init} - (i-1)r - \sum_{k<i} e_k \ .$$

On remarque que le volume maximal au temps i est inférieur à la borne supérieure de l'intervalle car r est supérieur à $\sum_{k<i-1} e_k$.
Comme on se trouve bien dans l'intervalle I_i^V la quantité maximale que l'on peut débiter dans la liaison L_1 est $qmax_i = e_i$. De plus on ne peut pas utiliser la turbine si c'est pour produire moins de $1kW$. Comme le coefficient de production de la turbine vaut $1/e_i$ cela implique que si on utilise la turbine, on le fait avec un débit d'eau valant exactement e_i.

— **pour que le planning soit valide, il faut que la quantité totale d'eau débitée par L_1 sur l'horizon de planification soit égale à s :**
Pour que le planning soit valide, il faut que la quantité d'eau contenue dans le réservoir à la fin de l'horizon de planification soit la même que la quantité initiale. Or les apports en eaux dans le réservoir durant l'horizon de planification sont de $K^2(e_K+1)+s$. Par la liaison sans turbine on débite exactement la quantité $K^2(e_K+1)$. Il faut donc que la quantité totale d'eau débitée par la turbine soit s pour que le planing soit valide.

De ces deux points on déduit que la réponse au HSP est *"oui"* pour l'instance $R(I)$ si et seulement si la réponse au problème SSH est *"oui"* pour l'instance I.

3.4 État de l'art des méthodes appliquées à la résolution de problèmes de planification dans des réseaux hydro-électriques

Les problèmes de planification des eaux dans des réseaux hydro-électriques sont très répandus dans la littérature. Néanmoins il est rare de trouver deux papiers pour lesquels la formulation du problème soit vraiment identique, ce qui rend difficile la comparaison des résultats. Ceci est lié au fait qu'il s'agit de problèmes appliqués à des réseaux réels dont les spécificités sont différentes. De plus les choix de simplifications permettant d'aboutir à une modélisation sont également différents. Dans cette partie nous allons présenter un état de l'art des méthodes existantes en considérant tout d'abord les méthodes exactes, puis les métaheuristiques et enfin les méthodes hybrides.

Les approches exactes

La programmation dynamique ou programmation dynamique différentielle est une méthode exacte bien adaptée aux problèmes de planification des réseaux hydro-électriques. Néanmoins comme la taille du graphe d'états augmente exponentiellement avec la taille des données, il est généralement impossible de les appliquer directement sur des problèmes réels. Cependant de nombreuses adaptations de la programmation dynamique permettant de réduire l'espace de recherche ont été proposées [47, 65, 198]. Elles permettent d'obtenir des solutions très précises mais ne garantissent cependant pas l'optimalité.

Des méthodes basées sur la programmation à variables mixtes ont également été proposées [28, 57, 143, 156]. Leur utilisation nécessite de modéliser le problème comme un programme linéaire mais comme vu précédemment notre modèle est également linéaire. En revanche elles ne sont appliquées que sur des problèmes relativement petits (en terme de nombre de réservoirs et de liaisons, de capacités des réservoirs par rapport au débits minimaux, de la taille de l'horizon de planification et surtout en terme de nombre d'intervalles

de défintion utilisés pour construire une approximation de la fonction d'évaluation). Cette restriction est due au fait qu'elles sont très consommatrices en temps de calcul. L'avantage principal est qu'elles permettent d'obtenir une solution exacte du problème par rapport à la modélisation choisie.

Le problème peut également être traité avec des algorithmes de flots si l'on ne considère pas les dépendances aux volumes d'eaux et que la contrainte (3) est relaxée [51].

Les métaheuristiques

De nombreuses métaheuristiques ont été appliquées à des problèmes de planifications dans des réseaux hydro-électriques, notamment des algorithmes génétiques [100, 135, 194, 195, 213], des algorithmes de recherche tabou [118, 119], des réseaux de neurones [132] et des algorithmes d'essaims de particules [200]. Elles permettent d'obtenir des solutions de qualité en des temps raisonnables. Cependant elles tendent à converger lentement ou à rester coincées dans un optimum local. De plus elles sont affectées négativement par la grande interdépendance entre les variables de décision.

Les méthodes hybrides

Une façon d'améliorer la précision des métaheuristiques tout en gardant leur rapidité est d'utiliser des méthodes hybrides. Dans [202] une hybridation de bas niveau collaborative est proposée. L'idée est d'utiliser des éléments de l'optimisation chaotique [76] tels que les séquences chaotiques et la notion de rétropropagation d'erreurs dans un algorithme génétique pour éviter sa convergence prématurée.

La méthode proposée dans [131] combine un algorithme glouton, un algorithme génétique, de la recherche locale et de la programmation dynamique. Elle se déroule en trois étapes : Tout d'abord une population initiale de solutions est générée en utilisant une heuristique basée sur un algorithme glouton, puis un algorithme génétique combiné à de la recherche locale est appliqué et enfin un post-traitement utilisant la programmation dynamique est appliqué pour améliorer les solutions.

La programmation dynamique différentielle discrète, méthode présentée dans la section 4.2 a été appliqué pour résoudre ce genre de problème [69]. L'hybridation entre programmation dynamique différentielle discrète et algorithme génétique également présentée section 4.2 a aussi été appliquée à un problème de planification d'un réseau hydro-électrique [182].

4 Conclusion

Dans ce chapitre nous avons présenté deux problèmes d'application, le problème d'affectation d'unités et le problème de planification optimale des débits d'eau dans un réseau hydro-électrique. C'est l'étude de ces problèmes qui a motivé la proposition de la méthode DYNAMOP qui sera décrite dans le prochain chapitre et qui constitue la principale contribution de cette thèse. Par la suite, ces problèmes seront donc utilisés pour évaluer DYNAMOP.

Nous avons tout d'abord fait une description précise du problème d'affectation d'unités et du problème de planification optimale des débits d'eau. Pour cela nous avons à chaque fois proposé une modélisation mathématique. Les notations introduites dans ces modèles seront réutilisées tout au long de ce manuscrit.

Pour chacun des problèmes, une analyse de complexité a également été conduite. Nous avons pu voir que ces deux problèmes sont des problèmes NP-complets. Pour le problème

de planification optimale des débits d'eau dans un réseau hydro-électrique, nous avons proposé une démonstration originale de cette NP-complétude. Cette démonstration permet également de voir que même en se restreignant à une sous-classe de problèmes très simples, celle pour laquelle le réseaux hydro-électrique n'est constitué que d'un unique réservoir et d'une unique turbine, le seul problème qui consiste à déterminer l'existence d'une solution réalisable est déjà un problème NP-complet. Ces études de complexité sont importantes car elles permettent de justifier le fait que l'on s'intéresse à des méthodes d'approximation pour traiter des instances de grande taille. Dans ces analyses, nous avons également soulevé une autre particularité commune à ces deux problèmes, qui est la forte dépendance entre les variables de décisions. Cette particularité contribue à rendre ces problèmes difficiles à résoudre. En particulier, elle rend difficile la proposition d'heuristique efficace car la forte interdépendance des variables de décisions complique le parcours de l'espace des solutions réalisables et entraîne que des solutions proches peuvent avoir des évaluations très éloignées. Ce dernier point nous intéressera particulièrement lorsque nous détaillerons DYNAMOP, l'approche que nous proposons.

Nous avons également dressé un état de l'art des différentes méthodes ayant été utilisées pour résoudre chacun de ces problèmes ou des problèmes similaires. Cet état de l'art nous permettra de situer notre travail et servira également de fondement à certaines de nos propositions.

Chapitre 3

Présentation de DYNAMOP

Dans ce chapitre nous présentons DYNAMOP, une méthode hybride entre programmation dynamique et métaheuristique. Cette méthode est la principale contribution de cette thèse.

Nous commençons par présenter les problématiques liées à la résolution de certains problèmes, dont en particulier nos deux problèmes d'application, qui ont motivés la proposition de DYNAMOP. Nous présentons ensuite en détails le principe de fonctionnement de DYNAMOP, puis nous concluons en discutant de son intérêt potentiel.

Des articles sur cette méthode ont d'ores et déjà été publiés dans des actes de conférences internationales :

- *EVO* 2014 à Grenade, en Espagne (publication LNCS) [77] ;*
- *LION 2015 à Lille, en France (publication LNCS) [79].*

1 Introduction

Dans le chapitre précédent, deux problèmes de production d'énergie ont été introduits. Nous avons pu voir que ces problèmes sont particulièrement difficiles. Tout d'abord, il s'agit de problèmes NP-difficiles. Cela implique, sous l'hypothèse largement utilisée que $P = NP$, que le temps de résolution de ces problèmes par une méthode exacte augmente exponentiellement avec la taille des données. Les problèmes industriels étant généralement de grande taille, il en résulte, dans beaucoup de cas pratiques, une impossibilité d'appliquer ces méthodes pour obtenir des résultats en des temps raisonnables.

De plus, il est également difficile d'aborder ces problèmes avec des heuristiques. En effet, les décisions étant fortement liées, pour optimiser le processus de production dans son ensemble, il faut obligatoirement considérer simultanément plusieurs étapes de ce processus, c'est à dire que plusieurs optimisations locales n'entraînent pas une optimisation globale.

Nous proposons, dans ce chapitre, un méthode hybride entre une méthode exacte, la programmation dynamique et une métaheuristique, l'algorithme génétique, qui tente de dépasser les inconvénients de chacune de ces approches en combinant leurs forces. Cette méthode sera appelée DYNAMOP pour *Dynamic Programming based Metaheuristic for Optmization Problem*. L'idée de base de cette méthode est d'utiliser un algorithme génétique pour parcourir le graphe d'états associé à la programmation dynamique. Ainsi DYNAMOP est un algorithme génétique qui manipule une population de chemins d'un graphe d'états en utilisant des opérateurs évolutionnaires adaptés. Nous souhaitons tirer profit du concept d'état utilisé en programmation dynamique pour construire une métaheuritique moins sensible aux dépendances entre les décisions. En outre, la convergence de l'algorithme est aidée par l'utilisation d'opérateurs évolutionnaires intelligents basés sur l'application de la programmation dynamique dans des graphes d'états restreints définis autour de la solution courante. Nous espérons ainsi construire une méthode bénéficiant

de la rapidité des métaheuristiques tout en gagnant en précision via l'utilisation de la programmation dynamique.

Dans ce chapitre, nous présenterons, DYNAMOP de façon très générique, nous l'appliquerons ensuite, dans les chapitres qui suivent aux problèmes d'affectation d'unités et de gestion des eaux dans un réseau de production hydroélectrique. A cet effet, nous commencerons par présenter l'idée général de DYNAMOP en détaillant les problématiques nous ayant menés à faire cette proposition, puis nous présenterons en détails chaque module de DYNAMOP. Nous clôturerons ce chapitre en concluant sur les avantages potentiels de DYNAMOP.

2 Idée générale

Dans cette section nous allons présenter l'idée de base sous-jacente à DYNAMOP. Pour cela nous allons tout d'abord relever les éléments des problèmes de planification de production qui les rendent difficiles à résoudre avec des métaheuristiques classiques. En effet, la proposition de DYNAMOP a été motivée par la volonté de réduire l'impact néfaste de ces éléments.

2.1 Motivations

Les problèmes industriels de planification de la production d'énergie sont des problèmes où n décisions $(d_1, d_2, ..., d_n)$ ont à être prises, chacune concernant une étape du processus de production. Comme nous l'avons mentionné dans le chapitre précédent, la difficulté de ces problèmes s'explique en partie de par la forte interdépendance entre les variables de décisions. Celles ci peuvent être liées à deux niveaux soit dans le sens où l'ensemble des choix possibles pour la décision d_i dépend de l'état dans lequel le système de production se trouve suite aux décisions précédente $(d_1, ..., d_{i-1})$. On dira que les décisions sont *liées par contraintes*. Soit dans le sens où l'impact d'une décision d_i sur la fonction objectif dépend de l'état dans lequel le système de production se trouve suite aux décisions précédentes $(d_1, ..., d_{i-1})$. On dira alors que les décisions sont *liées par l'objectif*.

Dans le cadre du problème d'affectation d'unités, présenté au chapitre 2 section 2, la décision d_i correspond aux choix des unités de production qui sont éteintes ou allumées durant le i^{ieme} intervalle de temps de l'horizon de planification. Étant donné les contraintes de temps de marche et d'arrêt minimum, les décisions sont liées par contrainte. Elles sont également liées par l'objectif car le coût de démarrage dépend du temps depuis lequel une unité est restée éteinte avant d'être allumée.

Pour le problème de planification des eaux dans un réseau hydroéléctrique, présenté au chapitre 2 section 3, une décision d_i correspond aux quantités d'eaux qui sont débitées pour chaque conduite du réseau au temps i. Elles sont fortement liées par contraintes car les contraintes (1), (3) et (4) sont tributaires des hauteurs d'eau dans les réservoirs au temps i qui dépendent des débits effectués durant les périodes de temps précédentes. De plus, elles sont fortement liées par l'objectif puisque la fonction de production des turbines dépend non seulement des débits effectués mais aussi de la hauteur d'eau dans le réservoir en amont.

Pour ce genre de problème, où les décisions sont fortement liées, il est difficile de proposer des heuristiques efficaces car pour optimiser un processus de production dans son ensemble il faut considérer simultanément plusieurs étapes de production. En effet plusieurs optimisations locales n'entraînent pas nécessairement une optimisation globale.

La programmation dynamique est une méthode d'optimisation exacte qui peut être appliquée sur ce type de problème. Comme expliqué au chapitre 1, elle se base sur une modélisation du problème sous forme d'un graphe d'états dans lequel la solution optimale correspond au plus court chemin entre l'état initial et un état final. Les états synthétisent les informations sur les décisions qui ont été prises pour les atteindre permettant de définir les décisions pouvant être prises à l'étape suivante et leurs coûts. Les arcs sont évalués par le coût des décisions permettant de transiter d'un état à un autre. La programmation dynamique se base ensuite sur le fait que tout sous-chemin d'un chemin optimal est un sous-chemin optimal, pour proposer une méthode d'énumération partielle optimale. Mais, comme vu lors des états de l'art du chapitre 2, la programmation dynamique reste en général très coûteuse et ne permet, par exemple, pas de résoudre les problèmes d'affectation d'unités ou de planification des débits d'eau dans des réseaux hydro-électriques sur des instances de tailles réalistes.

D'un autre côté des métaheuristiques telles que les algorithmes génétiques peuvent être utilisées. Ils ont l'avantage de permettre d'obtenir des solutions en des temps raisonnables même pour des instances de grandes tailles. Mais ceux-ci semblent souffrir de l'interdépendance entre les variables de décisions. En effet, les algorithmes génétiques se basent sur les principes d'exploitation (via les opérateurs de sélection et de croisement) et d'exploration (via les opérateurs de sélection et de mutation). L'idée de l'exploitation est de construire une nouvelle solution en recombinant de bonnes solutions déjà trouvées. En général, le génotype d'un individu dans un algorithme génétique correspond à une séquence de décisions $(d_1, ..., d_n)$, où chaque gène est une décision. Lors d'un croisement, on construit une solution en recombinant des séquences de décisions en provenance de différents individus parents. Seulement, la forte dépendance entre les variables de décision peut entraîner que des décisions (des gènes) ayant un impact positif sur la fonction objective (le phénotype) si elles sont précédées de certaines autres décisions (si ils sont dans un certain environnement génotypique) aient un impact négatif si elles sont précédées d'autres décisions, et inversement. De ce fait, l'impact de l'interdépendance est négatif sur l'exploitation car il est néfaste à la corrélation entre la qualité des individus croisés et la qualité de l'individu obtenu. Ceci fait référence à ce que nous avons nommé la propriété d'héritabilité phénotypique. De même, l'interdépendance entre les variables a un impact négatif sur le principe d'exploration. En effet, elle entraîne que le voisinage d'une solution peut être composé de solutions de qualités très diverses. Autrement dit des solutions proches sur le plan génotypique peuvent être éloignées sur le plan phénotypique. On dit que la propriété de localité est mauvaise.

DYNAMOP a donc pour objectif de combiner la programmation dynamique et les algorithmes génétiques de façon à tirer avantage des deux approches. L'idée principale de DYNAMOP est d'utiliser une métaheuristique pour parcourir le graphe d'états de la programmation dynamique. Les solutions sont donc représentées comme des séquences d'états. Les états étant définis de manière à synthétiser les dépendances entre les variables de décision, cette représentation va nous permettre de tenir compte de ces dépendances et donc d'améliorer les propriétés d'héritabilité phénotypique et de localité de l'algorithme génétique. La programmation dynamique est également utilisée pour accélérer le processus de convergence en résolvant des sous-problèmes définis à partir d'une ou plusieurs solutions. Dans la prochaine partie, nous présentons plus en détails le principe générale de DYNAMOP.

2.2 Principe

L'idée de base de DYNAMOP est d'utiliser un algorithme génétique pour guider la recherche du plus court chemin dans le graphe d'états défini par la programmation dynamique. Ainsi DYNAMOP est un algorithme génétique qui manipule des solutions représentées sous forme de chemins du graphe d'états. C'est à dire qu'un génotype est défini comme une suite d'états du système $((S_i)_i)$ où un gène correspond à un état. Dans la figure 3.1, le graphe représenté est un graphe d'états dans lequel le chemin rouge correspond à une solution. Il relie l'état initial S^* à un état terminal qui correspond à une feuille du graphe. Chaque solution est évaluée par la sommes des coûts des arcs du chemin associé qui sont calculés avec la fonction *reward* telle que définie dans la méthode de programmation dynamique. Pour alléger les notations, par la suite $R(S_i, S_{i+1})$ sera identifié à $R(S_i, T^{-1}(S_i, S_{i+1}))$

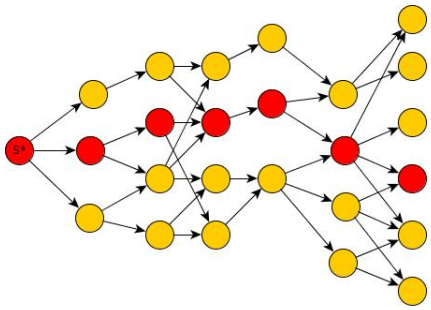

FIGURE 3.1 – Représentation DYNAMOP.

Les opérateurs évolutionnaires utilisés par DYNAMOP doivent être pensés afin de tirer profit de cette représentation. La mutation doit être construite de manière à modifier un nombre minimal d'arcs du chemin tout en possédant la propriété d'ergodicité (c'est-à-dire qu'elle doit permettre de parcourir l'ensemble de l'espace de représentation en partant d'un génotype quelconque). Quant au croisement, il doit être construit de manière à ce que les arcs composants le chemin associé à l'individu enfant proviennent essentiellement de l'un ou l'autre des parents tout en ce que l'individu obtenu corresponde à une solution valide. En effet, un grand intérêt de cette représentation est que l'évaluation associée à un individu est égale à la somme des valeurs des arcs du chemin correspondant, ce qui implique une "quasi-séparabilité" de la fonction d'évaluation à l'égard des gènes. En effet, lorsque le i^{ime} gène/état S_i d'un individu I est modifié en S'_i, la fonction d'évaluation de l'individu I' obtenu correspond à :

$$f(I') = f(I) - R(S_{i-1}, S_i) - R(S_i, S_{i+1}) + R(S_{i-1}, S'_i) + R(S'_i, S_{i+1})$$

Ceci implique une certaine corrélation entre une mutation modifiant peu d'arcs et une mutation ayant une bonne propriété de localité (impliquant un changement minime au niveau phénotypique). Ceci implique également que si le croisement a une bonne propriété d'héritabilité, c'est-à-dire si l'individu enfant possède dans son génotype principalement des arcs provenant des parents, alors il y aura une cohérence entre la qualité des parents et celle des solutions obtenues à partir de ces parents.

Afin d'améliorer la qualité de convergence de DYNAMOP, des opérateurs intelligents sont également utilisés. Leur principe est de construire un chemin (ou un sous-chemin) de

bonne qualité en explorant grâce à une méthode exacte un graphe d'états restreint défini à partir d'une ou plusieurs solutions présentes dans la population.

La figure 3.2 résume le principe général de DYNAMOP. Les modules restants inchangés vis-à-vis d'un algorithme génétique classique sont ceux en blanc et ceux qui sont spécifiques ou adaptés à DYNAMOP sont ceux en gris. Par la suite, nous allons expliciter en détail chacun des modules de DYNAMOP.

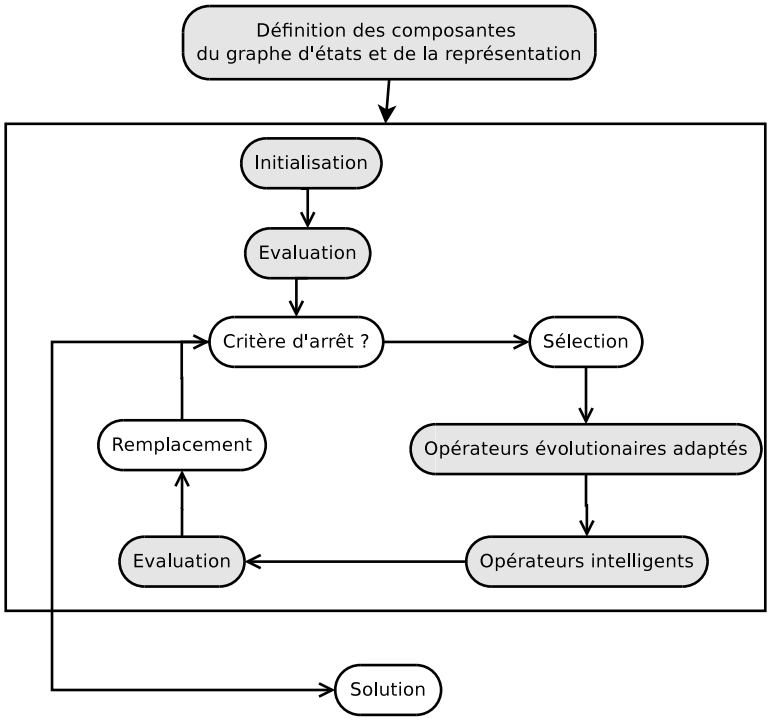

FIGURE 3.2 – Les modules spécifiques à DYNAMOP (en gris).

3 Les différents modules spécifiques à DYNAMOP

Dans cette section, les composantes spécifiques à DYNAMOP sont présentées.

3.1 Choix des composantes du graphe d'états et représentation des solutions

La première étape pour appliquer DYNAMOP est de définir les différents éléments de l'équation fonctionnelle de la programmation dynamique tels que définis dans la partie 2.2. C'est-à-dire qu'il faut définir l'espace d'états $\mathcal{S}$, l'espace de décision attaché à un état S, $D(S)$, la fonction coût ou *reward* R, la fonction de transformation T et la fonction de base f_b. Les solutions manipulées par DYNAMOP étant représentées sous forme de chemins dans le graphe d'état associé, ces choix vont totalement déterminer la représentation des individus.

Le choix de ces composantes doit se faire de façon à ce qu'il soit facile de déterminer s'il existe ou non des chemins entre deux états et s'il en existe de les générer. En effet, DYNAMOP nécessite que l'on puisse "facilement" construire les chemins liant deux états. Ici le terme "facile" réfère à l'existence d'un algorithme qui puisse s'exécuter rapidement.

C'est pourquoi il peut être intéressant de relaxer le problème, c'est-à-dire d'élargir l'espace d'états $\mathcal{S}$ ou les espace de décisions $D(S)$ en autorisant certaines décisions qui ne permettent pas de respecter toutes les contraintes de manière à ce que le problème de trouver des chemins entre deux états soit résolvable en un temps raisonnable. Dans ce cas, les décisions invalides sont gérées par des pénalités ajoutées à la fonction coût R.

Une fois toutes les composantes de l'équation fonctionnelle de la programmation dynamique fixées ; la représentation d'un individu correspond à une séquence d'états d'un chemin valide du graphe liant l'état initial S^* à un état terminal $(S_i)_i$ et à la séquence des valeurs des arcs $(e_i)_i$ correspondants.

Autrement dit le couple $(S_i, e_i)_{i=1..n}$ correspond à un individu valide si et seulement si il existe une séquence $(d_i)_i$ telle que :

$$\begin{aligned} d_i &\in D(S_{i-1}) \\ S_i &= T(S_{i-1}, d_i) \\ e_i &= R(S_{i-1}, d_i) \end{aligned}$$

$$f_b(S_i) \begin{cases} = 0 & si \quad i < n \\ > 0 & sinon \end{cases}$$

En posant $S_0 = S^*$.

3.2 Initialisation

L'initialisation peut se faire en utilisant un algorithme de parcours aléatoire de graphe. Comme le graphe est sans circuit, cet algorithme est très simple (cf algorithme 1).

Algorithme 1 Initialisation

Entrée : S^*, D, T, R, b
  $C \leftarrow$ chemin vide.
  $S \leftarrow S^*$
  tant que $card(D(S)) > 0$ **faire**
    Choisir aléatoirement d dans $D(S)$.
    ajouter $(T(S,d), R(S,d))$ à C.
    $S \leftarrow T(S,d)$
  fin tant que
  si $f_b(S) = 2$ **alors**
    ajouter $(T(S,d), \infty)$ à C.
  sinon
    ajouter $(T(S,d), 0)$ à C.
  fin si
  renvoie C

3.3 Évaluation

L'évaluation d'un individu $S = (S_i)_{i=1...N}$ correspond à la somme des valeurs de ses arcs :

$$fit(S) = \sum_{i=1}^{N} R(S_{i-1}, S_i)$$

Où S_0 correspond à l'état initial S^*. Les valeurs des arcs $R(S_{i-1}, S_i)$ sont stockées dans la représentation par le vecteur e ($e_i = R(S_{i-1}, S_i)$) de sorte qu'il est possible de mettre en place une Δ-évaluation et une évaluation itérative. Ces techniques consistent à n'évaluer que les arcs ayant été modifiés ou introduits par un opérateur génétique à la fin de chaque génération. Après une mutation on ré-évalue l'individu avec une évaluation itérative. C'est à dire que si à la génération i l'individu S^i est obtenu en modifiant l'état j, S^{i-1}_j, de l'individu S^{i-1} issu de la génération $i-1$, alors l'évaluation de l'individu S^i est effectuée via les opérations suivantes :

$$\begin{gathered} e^i_j \leftarrow R(S^i_{j-1}, S^i_j) \\ e^i_{j+1} \leftarrow R(S^i_j, S^i_{j+1}) \\ fit(S^i) = fit(S^{i-1}) - e^{i-1}_j - e^{i-1}_{j+1} + e^i_j + e^i_{j+1} \end{gathered}$$

De plus, si la solution S est obtenue en croisant les solution S^1 et S^2 et que les arcs E_i sont ceux hérités du parent S^i et les arcs E_n ceux qui sont propres à l'individu enfant S, l'évaluation de S se fait avec la Δ-évaluation :

$$fit(S) = \sum_{i \in E_1} e^1_i + \sum_{i \in E_2} e^2_i + \sum_{i \in E_n} R(S_{i-1}, S_i)$$

Ces stratégies peuvent permettre un gain important en temps de calcul. Notamment si le calcul de la fonction *reward* est consommateur en temps.

3.4 Mutation

L'idée de base pour construire l'opérateur de mutation de DYNAMOP est d'utiliser la notion d'état pour contrôler la qualité de la localité de la mutation. Pour rappel, la localité est l'effet sur la solution phénotypique lorsque l'on applique une petite perturbation sur le génotype avec la mutation. Lorsque le changement est petit sur le plan génotypique il doit en résulter un petit changement sur le plan phénotypique. La représentation des solutions sous forme de chemins d'un graphe d'états de DYNAMOP permet d'établir un lien entre les solutions proches génotypiquement et les solutions proches phénotypiquement. En effet, deux solutions sont proches génotypiquement si elles partagent des arcs en communs et étant donné que leurs fitness correspondent à la somme des valeurs de leurs arcs on peut dire qu'elles ont dès lors un bagage phénotypique commun.

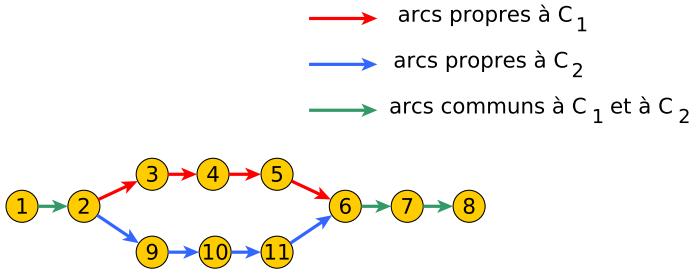

FIGURE 3.3 – La proximité de deux individus génotypiques est caractérisée par le nombre d'arcs qu'ils partagent.

La figure 3.3 présente deux solutions génotypiques qui correspondent aux chemins $\mathcal{C}_1$: $1-2-3-4-5-6-7-8$ et $\mathcal{C}_2$: $1-2-9-10-11-7-8$. Sur le plan génotypique,

la proximité de ces deux solutions est caractérisée par le nombre d'arcs communs aux chemins qui les représentent, c'est à dire aux arcs verts. La fitness du chemin $\mathcal{C}_1$ est la somme des valeurs des arcs verts plus la somme des arcs rouges et la fitness du chemin $\mathcal{C}_2$ est la somme des valeurs des arcs verts plus la somme des valeurs des arcs bleus. Les arcs en commun correspondent donc aussi à des caractéristiques phénotypiques communes aux deux solutions et on peut dire que d'un point de vue phénotypique cette notions de proximité est également viable.

La mutation doit donc permettre de modifier la solution en changeant un nombre minimal d'arcs. Pour que la mutation soit intéressante, il est préférable qu'elle soit ergodique et il faut qu'elle ne génère que des individus valides.

La construction de l'opérateur de mutation dépendra beaucoup de la structure du graphe d'états, néanmoins deux grands axes stratégiques peuvent être développés.

Le premier est de sélectionner aléatoirement deux états S_1 et S_2 séparés par au maximum L autres états et de remplacer le sous-chemin liant S_1 à S_2 par un nouveau sous-chemin aléatoirement généré. Si S_2 est un état terminal on considère les sous-chemins liant S_1 à n'importe quel état terminal. Nous appelerons simplement ce type de mutations, mutations de sous-chemin.

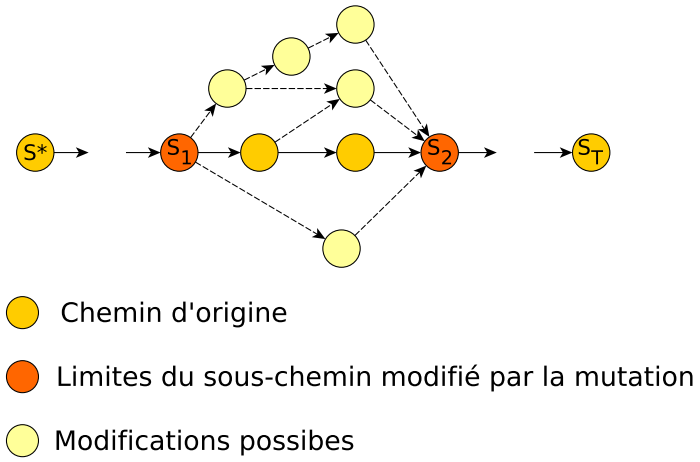

FIGURE 3.4 – Mutation de sous-chemin entre deux états non terminaux avec $L > 2$.

La figure 3.4 illustre cette stratégie dans le cas où S_2 n'est pas un état terminal. Le chemin représentant l'individu à muter est celui composé des sommets jaunes foncés. Les individus que l'on peut obtenir en appliquant la mutation entre S_1 et S_2 sont ceux suivant le chemin jaune foncé jusque S_1 puis empruntant un des sous-chemins composés d'arcs en pointillés et de sommets jaunes clairs puis suivant de nouveau le chemin d'origine jaune foncé. Similairement, la figure 3.5 illustre le cas où S_2 est un état terminal. En considérant que l'ensemble des états terminaux du graphe est $\{S_{T,1}, S_{T,2}, S_{T,3}, S_2\}$ alors l'ensemble des chemins atteignables après application de la mutation entre S_1 et S_2 est l'ensemble des chemins représentés sur le graphique.

L'avantage est qu'il est facile d'avoir un contrôle sur la taille de cette mutation en jouant sur la taille de L. Cependant il peut être difficile de garantir la parcourabilité de l'espace de recherche sans que L soit de grande taille, c'est pourquoi l'opérateur de mutation pourra être construit en utilisant une autre approche.

La seconde approche consiste à sélectionner aléatoirement un état S_i du chemin $\mathcal{C}$ à

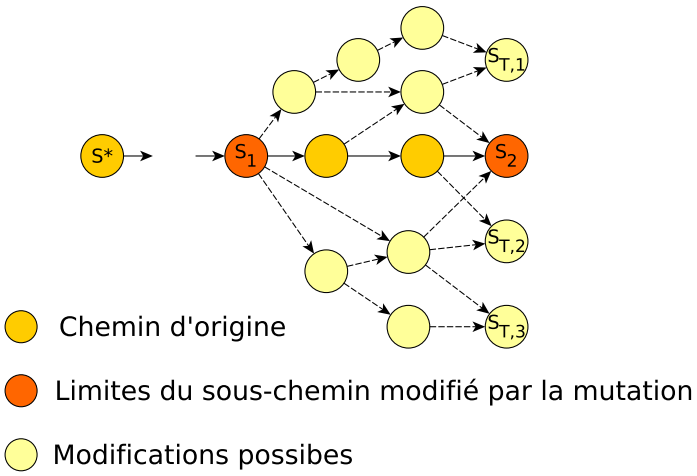

FIGURE 3.5 – Mutation de sous-chemin dans le cas où S_2 est un état terminal avec $L > 2$.

muter et de le remplacer par un nouvel état S'_i choisi aléatoirement dans son voisinage. Le chemin mutant est ensuite obtenu en rattachant S'_i à $\mathcal{C}$ en utilisant un nombre minimal d'arcs. Une telle mutation sera appelée mutation par bifurcation. Les Figures 3.6 et 3.7 illiustrent ce processus. Il sera toujours possible de lier le chemin $\mathcal{C}$ au nouveau sommet

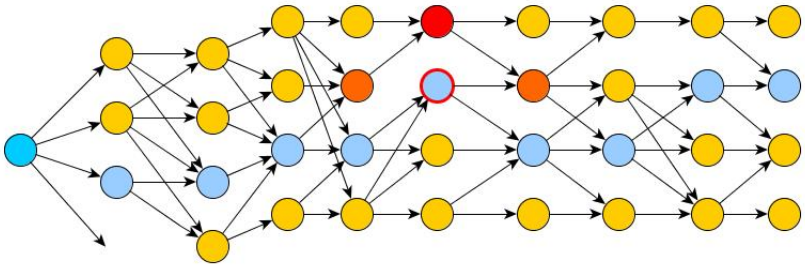

FIGURE 3.6 – Application de la mutation par bifurcation au chemin bleu . Le sommet entouré va être muté en le sommet rouge. Le plus court chemin rattachant le sommet rouge au chemin initial est celui passant par les sommets oranges.

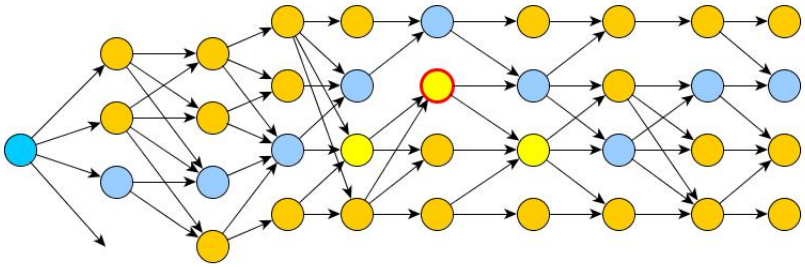

FIGURE 3.7 – Résultat de la mutation par bifurcation.

S'_i par la droite car il y a unicité de l'état initial. En revanche, il peut être impossible de trouver un chemin entre S'_i et un sommet composant le chemin $\mathcal{C}$. Dans ce cas l'idée est d'essayer de se rapprocher au maximum du chemin $\mathcal{C}$. Ici la notion de proximité entre deux états doit mesurer la similarité de leur influence sur les décisions à venir. On dira par exemple que l'état S' est proche de l'état S si l'ensemble des décisions pouvant être prises à partir de l'état S peuvent être prises à partir de l'état S', autrement dit que $D(S) \subset D(S')$ et que l'évaluation d'une décision prise partir de S est similaire à son évaluation à partir de S', c'est à dire $R(S,d) - R(S',d) < \epsilon \ \forall d \in D(S)$.

3.5 Croisement

Le principe de base de l'opérateur de croisement de DYNAMOP est de faire en sorte que l'individu enfant hérite des arcs de ses parents. Comme vu précédemment les arcs caractérisent à la fois l'individu sur le plan génotypique et sur le plan phénotypique. De ce fait, chercher à ce que la solution enfant soit principalement composée d'arcs en provenance de ses parents assure une certaine cohérence entre la qualité phénotypique des parents et la qualité phénotypique de l'individu généré par leur croisement.

Un croisement qui peut être utilisé dans tous les cas est celui plusieurs fois utilisé pour résoudre un problème de routage avec un algorithme génétique [6, 125]. L'idée est de trouver un état commun aux deux chemins qui soit différent de l'état initial et de construire deux individus enfants en utilisant cet état commun comme point de croisement. Dans la

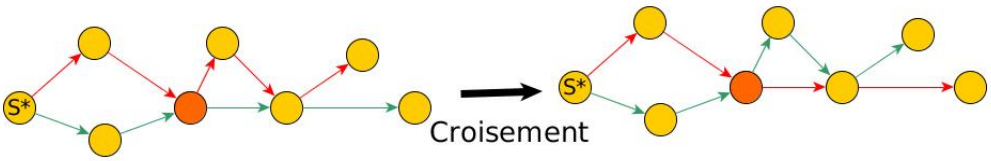

FIGURE 3.8 – Croisement : idée de base.

Figure 3.8, le chemin rouge et le chemin vert sont croisés à partir de l'état rouge. Mais ce croisement a un grand désavantage qui est que si les deux chromosomes choisis n'ont pas d'état commun en dehors de l'état initial ils ne peuvent pas être croisés. Néanmoins, cette idée peut être réutilisée et améliorée. En effet, il est toujours possible de connaître facilement les états voisins d'un état grâce aux fonctions D et T. De ce fait quand le croisement décrit précédement ne peut pas être appliqué car il n'y a pas d'état commun, on peut appliquer la stratégie suivante :

- Chercher un état S_1 appartenant au premier parent et à partir duquel il soit possible de joindre l'un des états S_2 du second parent en un arc.
- Construire l'individu enfant comme une copie du premier parent jusque S_1 concaténé à une copie du sous-chemin du second parent commençant en S_2.

Ce procédé peut-être généralisé à la recherche d'un chemin de transition liant les deux chemins parents avec moins de N arcs. On appellera un croisement de ce type un croisement 1-transition d'ordre N. Dans la même idée que pour le classique croisement 1-point qui est généralisé en croisement en K points on peut définir des croisements K-transitions. Une algorithme basé sur un parcours en largeur d'abord peut être utilisé pour trouver les chemins de transition. Mais en pratique cela peut être trop coûteux en terme de temps de calcul et d'utilisation mémoire. C'est pourquoi, il sera généralement mieux de définir des

algorithmes de recherche de chemins de transitions s'appuyant sur une connaissance du problème à traiter. Si on décide néanmoins d'utiliser un algorithme de parcours de graphe on peut jouer sur la taille de N pour réduire le nombre d'états à explorer. A contrario si le problème de déterminer l'existence de chemins entre deux états et d'en construire un s'il en existe est simple à résoudre on peut se passer de fixer N.

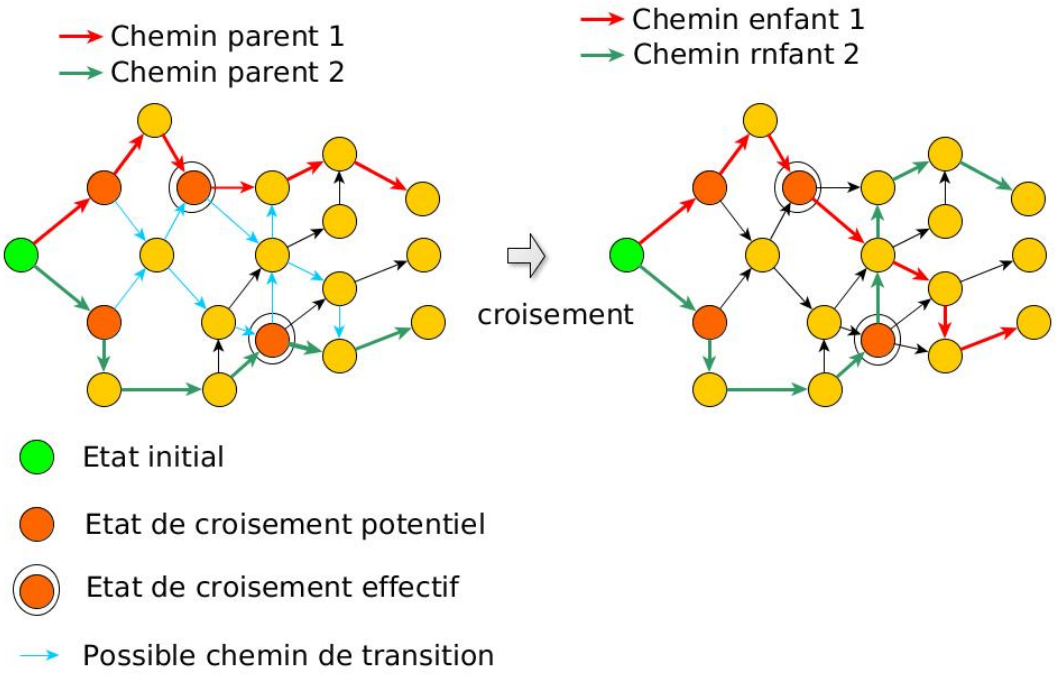

FIGURE 3.9 – Croisement 1-transition d'ordre 3.

La Figure 3.9 illustre le croisement 1-transition d'ordre 3 entre le chemin composé d'arcs vert et le chemin composé d'arcs rouge. Les états candidats pour être des points de croisement sont ceux en orange et les chemins de transition correspondant sont ceux passant par des arcs bleus.

Les états choisis pour être les points de croisement sont ceux entourés. L'individu enfant est présenté dans la seconde partie de l'image. Dans le cas où il n'y a pas de chemins de moins de N arcs connectant les deux chemins parents le croisement n'a pas lieu. S'il y a beaucoup de chemins de transitions entre les chemins parents, le choix du chemin de transition peut être stratégique (plus court chemins ou possédant le moins d'arcs possible pour maximiser l'héritabilité).

3.6 Opérateurs intelligents : hybridation

Dans DYNAMOP l'hybridation consiste en l'utilisation d'opérateurs évolutionnaires intelligents, c'est à dire d'opérateurs qui se basent sur des méthodes de résolution exactes pour améliorer la solution courante. L'objectif étant d'introduire des sous chemins optimaux qui pourront correspondre à des blocs de construction pour la solution optimale. Plusieurs stratégies peuvent être suivies pour proposer de tels opérateurs. Nous détaillerons ici les principales stratégies qui permettent de définir des opérateurs intelligents de mutation ou de croisement.

Mutation intelligente

Puisque la qualité phénotypique d'un individu dépend de la valeur de ses arcs ; une première façon de muter intelligement est d'augmenter la probabilité de modifier les arcs dont la valeur est mauvaise par rapport à ceux dont la valeur est bonne. Il est également possible d'utiliser des hybridations avec des méthodes d'optimisation exactes. Dans ce cas, l'objectif est de remplacer le chemin à muter par le meilleur chemin appartenant à son voisinage. Le voisinage d'un chemin est un graphe d'états restreint dans lequel le chemin est inclu. Pour définir un tel voisinage, deux stratégies peuvent être envisagées :

— **Restriction du graphe d'états en longueur :** Une petite portion du chemin comprise entre deux états choisis aléatoirement S_1 et S_2, est remplacée par le meilleur chemin joignant ces deux états. Ce meilleur chemin est construit en utilisant une méthode d'optimisation exacte telle que la programmation dynamique. Une distance maximale entre S_1 et S_2 est imposée de manière à limiter la taille du sous-espace d'états à considérer. Cette mutation va automatiquement permettre d'améliorer la solution complète. Il sagit en fait d'une application directe du principe de Bellman.

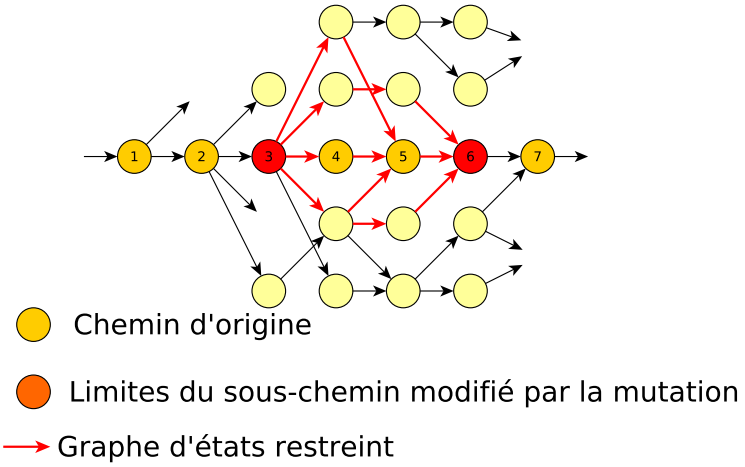

FIGURE 3.10 – Mutation intelligente avec restriction du graphe d'états en longueur.

La figure 3.10 illustre cette idée. Le chemin à muter est le chemin jaune foncé. Les deux sommets aléatoirement choisis sont les sommets en rouge. On cherche alors à remplacer le chemin intermédiaire par le meilleur chemin reliant ces deux sommets, ce qui revient à résoudre un problème de plus court chemin dans le graphe restreint composé des arêtes rouges.

— **Restriction du graphe d'états en largeur :** Le principe est de définir un graphe d'états restreint en largeur autour du chemin à muter. Pour ce faire la valeur de certaines variables d'état peut être fixée ou bien les valeurs qu'elles peuvent prendre peuvent être limitées à un petit intervalle centré autour de leurs valeurs originales dans l'individu à muter. Cette définition de voisinage est celle utilisée dans la méthode de programmation dynamique différentielle discrète [70] ou dans la méthode du corridor [161].

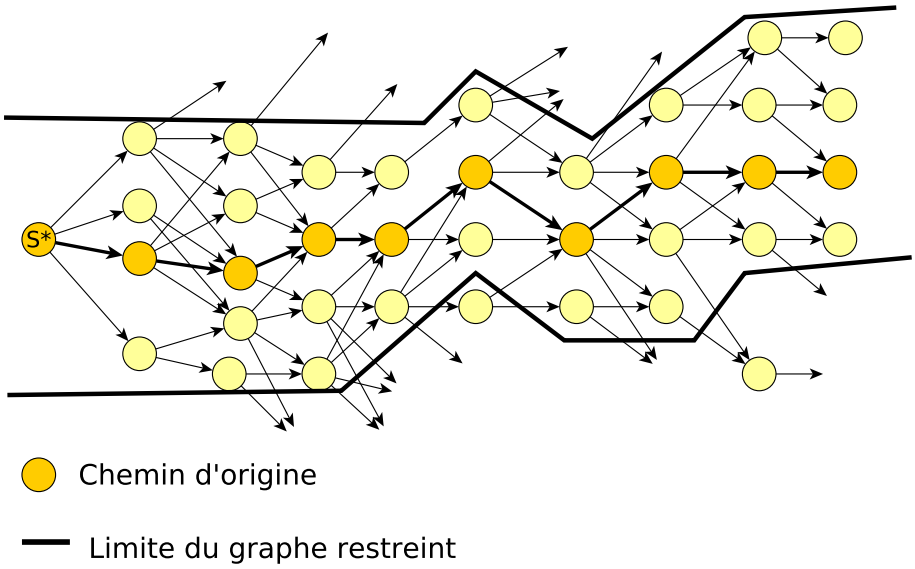

FIGURE 3.11 – Mutation intelligente avec restriction du graphe d'états en largeur.

La figure 3.11 illustre cette seconde stratégie. Le chemin à muter est le chemin jaune foncé, le graphe restreint définissant le voisinage de ce chemin est limité par les lignes noires. Après mutation le chemin actuel (jaune foncé) sera remplacé par le meilleur chemin composé de sommets compris entre les deux lignes noires.

Il est possible de combiner ces deux stratégies pour restreindre le graphe d'états à la fois en largeur et en longueur et ainsi obtenir un sous problème solvable de façon exacte en un temps raisonnable.

Croisement intelligent

Il est tout d'abord possible de tirer profit de la séparabilité de la fonction objective au regard des arcs pour croiser les individus intelligemment. On peut par exemple sélectionner une portion de chemin intéressante d'un individu parent et la rattacher à l'autre parent grâce à deux chemins de transition.

Un opérateur de croisement spécifique peut aussi être utilisé lorsque les individus sont suffisamment proches. L'idée est de sélectionner n individus et de les recombiner pour former un nouvel individu qui soit meilleur que ces n individus. Pour ce faire on cherche les états communs à tous les individus, notés $(S_i^c)_{i=1..k}$. Ensuite l'individu enfant est construit en sélectionnant les meilleurs sous-chemins parmi ceux des n individus compris entre chaque paire d'états (S_i^c, S_{i+1}^c) et (S_k^c, S_f). C'est une application directe du principe de Bellman, on appellera donc ce croisement *recombinaison de sous-politiques*.

La figure 3.12 illustre cet opérateur entre le chemin bleu, le chemin jaune et le chemin vert. Les états communs sont ceux en rouge et le chemin généré est celui passant par les arcs rouges. Dans cet exemple, on considère que le problème à traiter est un problème de minimisation et donc les sous-chemins sélectionnés sont ceux de plus petit coût.

Au lieu de sélectionner les états communs, il est possible de sélectionner les locus i pour lesquels :

$$\cup_{k=1..n}\{S_{i+1}^k\} \subset \cap_{k=1..n}\delta^+(S_i^k),$$

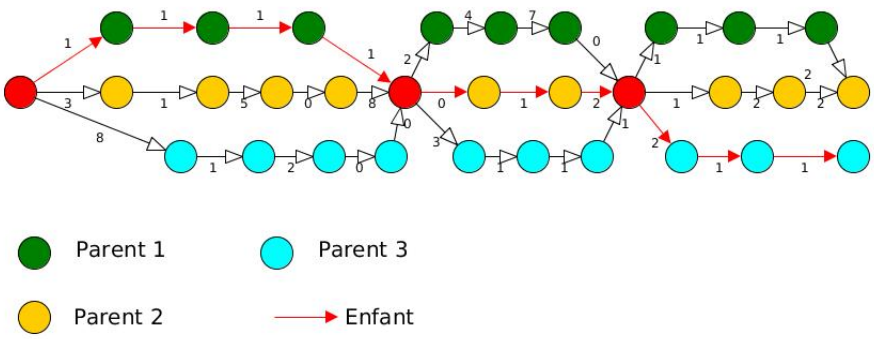

FIGURE 3.12 – Recombination de sous-politiques.

Où S_i^k est le i^{me} état du chemin du k^{me} individu et $\delta^+(S_i^k)$ est l'ensemble des états accessibles à partir de S_i^k (c'est à dire $T(S_i^k, D(S_i^k))$). C'est cette stratégie qui sera appliquée dans les cas d'application.

Il est également possible de proposer des croisements intelligents basés sur des méthodes exactes. Pour se faire, on peut distinguer deux stratégies :

— **Construction intelligente des chemins de transitions :** Le principe du croisement est de sélectionner un état S_1 du premier chemin et de trouver un chemin de transition de taille minimale (en terme du nombre d'arcs) liant cet état à l'autre chemin parent. L'individu enfant est alors constitué des arcs du premier parent jusque l'état S_1 puis des arcs du chemins de transition, puis des arcs du second parent. Ce processus peut être répété plusieurs fois pour avoir des croisements en plusieurs points (plus de précisions au paragraphe 3.5). Dans le cas où de nombreux chemins de transitions de taille minimale existent, une méthode exacte peut être utilisée pour sélectionner le meilleur chemin de transition de taille minimale.

— **Construction d'un graphe restreint à partir des chemins parents :** La seconde stratégie est de définir un graphe d'états restreint prenant en compte les caractéristiques des parents et de chercher le meilleur chemin dans ce graphe restreint pour construire l'individu enfant. De nombreuses stratégies sont envisageables pour construire le graphe d'états restreint. Le croisement proposé dans [130] est un exemple de ce type de croisement pour les problèmes de permutation. Le principe de ce croisement est de trouver un ordre partiel commun aux deux individus à croiser, l'individu enfant est construit en cherchant la meilleure solution consistante avec cet ordre partiel en utilisant la programmation dynamique. D'autres stratégies sont possibles, les valeurs des variables d'états peuvent êtres restreintes à celles des valeurs des variables d'états parentes ou être bornées par celles ci. Le choix de la méthode de construction du graphe restreint dépend du problème mais il doit prendre en compte les caractéristiques des parents et si possible permettre que leurs caractéristiques communes soient conservées.

4 Conclusion : Intérêt potentiel de DYNAMOP

Pour conclure nous allons résumer tous les aspects de DYNAMOP qui peuvent jouer en son avantage.

4.1 Améliorer la consistance de l'algorithme génétique

L'interdépendance entre les variables de décision est une caractéristique des problèmes qui est à prendre en compte quand on applique la programmation dynamique. En effet, un état doit être défini de manière à donner toutes les informations nécessaires à la définition de l'ensemble des décisions qui peuvent être prises à partir de cet état et au calcul de l'évaluation de leurs impacts sur la fonction objectif grâce à la fonction *reward*. Mais il doit donner le nombre minimal d'informations nécessaires de manière à ne pas alourdir la représentation et à minimiser la taille de l'espace d'états. Pour les algorithmes génétiques l'interdépendance entres décisions peut avoir un effet néfaste et il peut donc être intéressant d'en prendre compte pour essayer de minimiser cet effet. En effet, du fait de cette interdépendance la modification d'une unique décision peut générer de grands changements dans la fonction d'évaluation de l'individu, voire engendrer une solution invalide. Ainsi, si on utilise une représentation des solutions comme une liste de variables de décisions (un gène=une décision), ce qui est couramment fait, il est fort probable que les propriétés de localité et d'héritabilité de l'algorithme soient mauvaises. En effet, l'impact d'une décision sur la valeur d'évaluation de l'individu peut dépendre de toutes les décisions précédentes. Il est donc impossible d'évaluer une portion de génome indépendamment du reste du génome. La représentation utilisée dans DYNAMOP implique une plus grande séparabilité de la fonction d'évaluation par rapport au gène (qui correspondent à des états). En effet, la fonction d'évaluation correspond à la somme des valeurs des arcs. Il y a donc un lien direct entre la proximité des individus sur le plan génotypiques (qui partagent beaucoup de gènes) et leur proximité phénotypique (valeur de leur fonction d'évaluation). Comme vu précédemment, ceci permet d'avoir un certain contrôle sur les propriétés de localité et d'héritabilité phénotypique.

La représentation proposée, permet également de prendre mieux en compte l'aspect satisfaisabilité des contraintes. En effet, du fait que les variables de décisions sont liées par les contraintes, la modification d'une décision peut entrainer que certaines autres décisions ne sont plus valides et cette situation ne peut pas être évitée facilement avec une représentation classique. En revanche, la représentation sous forme de séquence d'états entraine que chaque décision est contextualisée, ainsi si certains états sont modifiés cela n'influe pas sur la validité des décisions associées à d'autres états.

4.2 Facilite la mise en place d'hybridations

La représentation proposée facilite la mise en place d'hybridation, en particulier avec la programmation dynamique. Tout d'abord, la représentation sous forme de chemin permet de définir facilement des sous-problèmes : Soit chercher le meilleur sous chemin liant deux états du chemin actuel afin d'en améliorer la qualité. Soit améliorer la solution actuelle en cherchant le meilleur chemin appartenant à un graphe d'états défini par une restriction autour de ce chemin. De plus, cette représentation permet de tirer un meilleur profit de ces hybridations car, d'une part, les croisements sont effectués de manières à construire des individus enfants étant composés essentiellement de sous chemins de leurs parents. Ainsi, les croisements sont faits de manière à ce que les individus enfants héritent des sous-chemins optimaux obtenus dans les hybridations. D'autre part, les mutations sont construites de

façon à minimiser le nombre d'arcs modifiés et donc permettent de conserver au maximum les bonnes portions de chemins insérées par les hybridations.

4.3 Permettre la mise en place d'une évaluation efficace

La fonction d'évaluation est séparable par rapport aux arcs. En effet elle correspond à la somme des valeurs des arcs. Ces valeurs étant stockées dans la représentation, après application d'un opérateur de mutation, il suffit de calculer les valeurs des arcs modifiés par cet opérateur et d'appliquer une méthode d'évaluation itérative pour obtenir la valeur de l'individu généré. De plus, après application d'un opérateur de croisement il suffit de calculer la valeur des arcs qui ne proviennent d'aucun des deux parents et d'appliquer une Δ-évaluation. Ceci peut permettre un gain de temps de calcul considérable car la fonction *reward* peut être coûteuse à évaluer. Ce gain peut être d'autant plus important que les opérateurs proposés cherchent à minimiser l'introduction d'arcs nouveaux.

4.4 Peut s'appliquer à de nombreux types de problèmes

DYNAMOP pourrait s'appliquer à de nombreux types de problème d'optimisation combinatoire. Pour qu'il soit possible et intéressant d'appliquer DYNAMOP il faut que :

- le problème ait la propriété de Bellman ;
- le sous-problème de déterminer l'existence d'un chemin entre deux états du graphe d'états, et d'en construire un, soit polynomial ;
- Le graphe d'états associé au problème ne soit pas un arbre mais qu'au contraire il contienne de nombreux cycles (nombre cyclomatique élevé).

Cela implique beaucoup de cas d'application, tels que des problèmes de routages dans des graphes, d'ordonnancement, de sélection, de partitionnement, de répartition, ou de gestion de production.

Il arrive que la programmation dynamique soit appliquée à des problèmes impliquant des variables de décisions dont l'espace de définition est continu, mais dans ces cas les valeurs possibles de ces décisions sont discrétisées. DYNAMOP peut être appliqué dans ces cas en utilisant la même stratégie. Cependant, dans certains cas, il sera possible de se dispenser de la discrétisation ce qui pourra permettre d'obtenir de meilleurs résultats car la discrétisation peut exclure la solution optimale de l'espace d'états résultant. Ceci sera possible si les états S peuvent être décrits sous la forme d'un couple $S = (n_k, S')$, où :

- $(n_k)_{k<N}$ est une suite donnant les différentes valeurs possibles d'une variable discrète.
- $S' \in E'$ inclut les autres variables d'état. Certaines d'entres elles pouvant être définies dans un espace continu.

La présence de n_k permet d'exclure le cas où toutes les variables sont continues

4.5 Peut se généraliser à d'autres paradigmes de métaheuristiques

Dans ce chapitre, nous avons appliqué DYNAMOP en utilisant un algorithme génétique et nous pensons que le concept de DYNAMOP est particulierement interessant pour ce type d'algorithmes. Cependant, l'idée de DYNAMOP peut se généraliser à de nombreux paradigmes de métaheuristique. Le principe central de DYNAMOP est, en effet, d'utiliser une métaheuristique pour parcourir le graphe d'état défini par une equation fonctionnelle de programmation dynamique donnée. Autrement dit, le concept de DYNAMOP est de manipuler des solutions représentées sous forme de chemins avec une métaheuristique dont les opérateurs sont adaptés à cette représentation.

En particulier, il serait très simple de généraliser DYNAMOP à toutes les métaheuristiques basées sur la recherche locale, car les mutations que nous avons proposées, peuvent facilement permettre de définir des voisinages de solutions.

Chapitre 4

Application de DYNAMOP au problème d'affectation d'unités

Dans ce chapitre, DYNAMOP est appliqué au problème d'affectation d'unités. L'application de notre méthode à un problème très étudié de la littérature permet d'en évaluer la qualité en la comparant aux autres algorithmes existants.

Nous avons présenté nos travaux concernant l'application de DYNAMOP au problème d'affectation d'unités lors de deux conférences internationales :

- *META 2014 à Marrakech, au Maroc [78] ;*
- *LION 2015 à Lille, en France (publication LNCS) [79].*

1 Introduction

Le problème d'affectation d'unités (voir chapitre 2 section 2) est, du fait de sa complexité et de son rôle stratégique pour la gestion de système de génération d'énergie, très étudié dans la littérature. Il consiste d'une part à fixer les plannings de marche et d'arrêt des unités et d'autre part à fixer la quantité exacte d'énergie produite par chaque unité allumée. L'objectif est de minimiser les coûts de production tout en répondant à la demande et en respectant les diverses contraintes opérationnelles.

L'application de DYNAMOP au problème d'affectation d'unités a plusieurs motivations.

Tout d'abord, nous pensons que la représentation sous forme de séquence d'états que nous avons proposé avec DYNAMOP peut être profitable pour un problème tel que le problème d'affectation d'unités. En effet, nous pensons que cette représentation peut être avantageuse par rapport à la représentation utilisée dans la majorité des métaheuristiques ayant été proposées pour ce problème qui consiste à utiliser un vecteur binaire de taille $T \times N$ dont chaque élément représente l'état de marche ou d'arrêt d'une unité durant une période de l'horizon de planification. Dès lors, avec la représentation usuelle, le voisinage d'une solution correspond à l'ensemble des vecteurs ne différant que d'un bit. Cependant, il y a une grande interdépendance entre les variables de décision : une unité doit rester allumée ou éteinte durant un temps minimal et le coût de production dépend du temps durant lequel l'unité est restée dans le même état. Ainsi, avec cette représentation, une solution de bonne qualité peut avoir un voisinage de solutions mauvaises et à l'inverse une solution de basse qualité peut avoir dans son voisinage des solutions très intéressantes. Ceci entraîne des problèmes de localité tel que définis précédemment. Or, nous avons vu que la représentation sous forme de chemin d'un graphe d'états, proposée pour DYNAMOP peut aider à réduire ce problème.

De plus, de nombreuses méthodes basées sur la programmation dynamique ont déjà été proposées pour résoudre le problème d'affectation d'unités ce qui facilitera la mise en place de l'hybridation.

Nous pensons également, qu'il est intéressant de tester DYNAMOP sur un problème très étudié tel que le problème d'affectation d'unités pour pouvoir en évaluer la performance en faisant des comparaison à des méthodes existantes très performantes.

La programmation dynamique étant à la base de DYNAMOP, nous montrerons dans un premier temps comment celle-ci peut être utilisée pour le problème d'affectation d'unités. Puis, nous indiquerons comment DYNAMOP peut être appliqué à ce problème et ensuite nous présenterons les résultats obtenus. Pour finir, nous conclurons sur les performances de DYNAMOP.

2 La programmation dynamique pour le problème d'affectation d'unités

Nous expliquerons, dans cette section, comment la programmation dynamique peut être utilisée pour résoudre le problème d'affectation d'unités de production. Pour cela nous commencerons par définir les différents éléments qui la caractérise (espaces d'états et de décisions, fonction de transformation...), puis nous expliquerons en détails comment le parcours de l'espace d'états ainsi défini s'effectue pour trouver la solution optimale. Nous verrons ensuite comment des méthodes approximatives basées sur la programmation dynamique peuvent être proposées pour le problème d'affectation d'unités.

2.1 Définition des éléments de la programmation dynamique

Pour mettre en place un algorithme de programmation dynamique, il est nécessaire de définir différents éléments.

Espace d'états

La définition d'un état doit permettre de déterminer les séquences de décisions qui sont réalisable à partir de cet état et leurs coûts. Ainsi, dans le cadre du problème d'affectation d'unités un état est défini par :

- Une période de temps t de l'horizon de planification ;
- l'état de marche ou d'arrêt des unités au temps t ;
- le temps depuis lequel les unités sont en état de marche ou d'arrêt au temps t. Plus précisément, si l'unité i est allumée, il est nécessaire de savoir dans combien de temps il sera possible de l'éteindre, c'est à dire le temps qu'il reste à attendre pour que l'unité ait été allumée depuis au moins $T^{on}_{min,i}$, ceci car cette information influencera les décisions futures possibles. Si l'unité est éteinte, il nous faut savoir le temps qu'il est encore nécessaire d'attendre avant de pouvoir l'allumer et si cet allumage s'effectuera à chaud ou à froid. Ainsi si l'unité est éteinte, la définition de l'état doit permettre de savoir si celle ci est éteinte depuis moins de $T^{on}_{min,i} + T_{cs,i}$ périodes et dans ce cas depuis combien de temps exactement.

Par la suite, on notera S_t un état associé à une période de temps t et on formalisera sa définition à l'aide d'un vecteur d'entiers $(S_{t,i})_{i \leq N}$. De sorte que $S_{t,i}$ est défini dans l'intervalle $[\![-T^{off}_{i,min} - T_{cs,i}, T^{up}_{i,min}]\!] - \{0\}$, et s'interprète comme ceci :

- Si $S_{t,i} = T^{up}_{i,min}$ alors au temps t l'unité i est éteinte depuis au moins $T^{up}_{i,min}$ heures.
- Si $0 < S_{t,i} < T^{up}_{i,min}$ alors au temps t l'unité i est allumée depuis $S_{t,i}$ heures.

— Si $S_{t,i} = -T^{off}_{i,min} - T_{cs,i}$ alors au temps t l'unité i est éteinte depuis au moins $T^{off}_{i,min} + T_{cs,i}$ heures.
— Si $-T^{off}_{i,min} - T_{cs,i} < S_{t,i} < 0$ alors au temps t l'unité i est éteinte depuis $-S_{t,i}$ heures.

Espace de décisions

L'espace de décisions $D(S_t)$ représente l'ensemble des décisions pouvant être prises à partir de l'état S_t. Dans ce cadre, une décision est de choisir, pour chaque unité de production i, si elle sera allumée ou éteinte durant la période $t+1$. Dans la suite, on modélisera une décision par un vecteur $(d_i)_i$, telle que :
— $d_i = -1$ si on décide que l'unité i sera éteinte.
— $d_i = 1$ si on décide que l'unité i sera allumée.
L'ensemble des décisions que l'on peut prendre à partir de l'état S_t est naturellement définit par les contraintes du problème :

$$d \in D(S_t) \Leftrightarrow \forall i :$$
— Si $0 < S_{t,i} < T^{up}_{i,min}$ alors $d_i = 1$
— Si $S_{t,i} = T^{up}_{i,min}$ alors $d_i \in \{-1, 1\}$
— Si $-T^{off}_{i,min} < S_{t,i} < 0$ alors $d_i = -1$
— Si $-T^{off}_{i,min} \leq S_{t,i}$ alors $d_i \in \{-1, 1\}$
— $\sum_{i,d_i>0} P_{min,i} \leq D_t$ et $\sum_{i,d_i>0} P_{max,i} \geq D_t + R_t$

La dernière condition permet que toute décision d pour lesquelles il n'est pas possible de distribuer la demande entre les unités allumées en respectant les contraintes de capacité et de réserve soient exclues.

Fonction de transformation

La fonction de transformation est alors définie comme ceci :

$$T(S_t, d) = S_{t+1}$$
tel que :

$$S_{t+1,i} = \begin{cases} T^{up}_{i,min} & \text{si } S_{t,i} = T^{up}_{i,min} \text{ et } d_i = 1 \\ -1 & \text{si } S_{t,i} = T^{up}_{i,min} \text{ et } d_i = -1 \\ -T^{off}_{i,min} - T_{cs,i} & \text{si } S_{t,i} = -T^{off}_{i,min} - T_{cs,i} \text{ et } d_i = -1 \\ 1 & \text{si } S_{t,i} = -T^{off}_{i,min} - T_{cs,i} \text{ et } d_i = 1 \\ S_{t,i} + d_i & \text{sinon} \end{cases} \quad (4.1)$$

États terminaux

Les sommets terminaux valides sont les états correspondant à la dernière période de planification T :

$$f_b(S_T) = 1$$

Fonction coût

La fonction coût ou fonction *reward* $R(S_t, d)$ correspond au coût optimale de production, que l'on peut obtenir au temps $t+1$, en prenant la décision d, sachant que le système était dans l'état S_t à la date t.

Ce coût correspond à la somme des coûts de démarrage CS_i et des coûts de consommation en carburant CF :

$$R(S_t, d) = \sum_{i=1}^{N} CS_i(S_t, d) + CF(d), \tag{4.2}$$

Où :

$$CS_i(S_t, d) = \begin{cases} 0 & \text{si} \quad S_{t,i} > 0 \text{ ou } d_i = -1 \\ CS_{cold,i} & \text{si} \quad -S_{t,i} = T^{off}_{i,min} + T_{cs,i} \\ CS_{hot,i} & sinon \end{cases} \tag{4.3}$$

$$CF(d) = max_p(\textstyle\sum_{i,d_i>0} a_{0i} + a_{1i}p_i + a_{2i}p_i^2)$$

tel que :

$$\begin{array}{l} P_{min,i} \leq p_i \leq P_{max,i} \quad \forall i, d_i > 0 \\ \sum_{i,d_i>0} p_i = D_{t+1} \end{array} \tag{4.4}$$

$CF(d)$ est calculée par la méthode de λ-itération telle que présentée dans [150].

Par la suite on écrira $R(S_t, S_{t+1})$ au lieu de $R(S_t, T^{-1}(S_t, S_{t+1}))$ afin de simplifier les notations.

2.2 Résolution

La méthode que nous appliquons est appelée programmation dynamique par chaînage avant, en anglais *forward dynamic programming* [189]. Il s'agit d'une stratégie de parcours du graphe d'états tirant profit de la structure multi-étapes ou *multistage* du graphe. En effet, les états de $\mathcal{S}_t$ attachés à la période t ont tous leurs successeurs parmi les états $\mathcal{S}_{t+1}$ attachés à la période $t+1$. L'idée est de construire la solution étape par étape, en cherchant à chaque étape t la valeur des meilleurs chemins entre l'état initial S_0 et les états $S_t \in \mathcal{S}_t$, en se basant sur la relation de récurrence suivante :

$$\begin{cases} f(S_1) = R(S_0, S_1) & \forall S_1 \in \mathcal{S}_1 \\ f(S_t) = \min\limits_{S_{t-1} \in \delta^-(S_t)} R(S_{t-1}, S_t) + f(S_{t-1}) & \forall S_t \in \mathcal{S}_t \end{cases} \tag{4.5}$$

Coupes

Afin de réduire les temps de calcul on introduit des coupes qui consistent à considérer comme terminaux les états à partir desquels il ne sera pas possible de trouver la solution optimale. Pour cela on utilise une relation de dominance entre les états de $\mathcal{S}_t$; telle que S_t^1 domine S_t^2 ($S_t^1 \trianglerighteq S_t^2$) si et seulement si :

$$\begin{cases} f(S_t^1) \leq f(S_t^2) & \\ S_{t,i}^1 \times S_{t,i}^2 > 0 & \forall i \\ S_{t,i}^1 \geq S_{t,i}^2 & \forall i, S_{t,i}^1 > 0 \\ S_{t,i}^1 = S_{i,2}^2 & \forall i, S_{t,i}^1 < 0 \end{cases} \tag{4.6}$$

Si $S_t^1 \trianglerighteq S_t^2$ alors la meilleure solution obtenable à partir de l'état S_t^2 sera moins bonne que celle obtenable à partir de l'état S_t^1. En effet toute séquence de décisions prise à partir de

S_t^2 peut être prise à partir de S_t^1 avec le même coût. Ainsi, à la fin de chaque étape on supprimera tous les sommets dominés. On peut donc considérer que la règle suivante est ajoutée à la définition de la fonction de base :

$$\forall S \in \mathcal{S}_t, \exists S' \in \mathcal{S}_t, S' \trianglerighteq S f_b(S) = 2$$

Pseudo-code

L'algorithme 2 résume l'application de la programmation dynamique au problème d'affectation d'unités.

Algorithme 2 Programmation Dynamique pour l'UCP

Entrée : S_0, D, T, R
  $f(S_0) \leftarrow 0.$
  $\mathcal{S}_0 \leftarrow S_0$
  $\mathcal{S}_t \leftarrow \emptyset$
  pour $t = 1$ à T **faire**
    pour tout $S_{t-1} \in \mathcal{S}_{t-1}$ **faire**
      pour tout $d \in D(S_{t-1})$ **faire**
        $S_t \leftarrow T(S_{t-1}, d).$
        $f(S_t) = f(S_{t-1}) + R(S_{t-1}, d)$
        S_t.prédecesseur $\leftarrow S_{t-1}.$
        si $\exists S \in \mathcal{S}_t$, tel que $S_t = S$ **alors**
          si $f(S_t) < f(S)$ **alors**
            Remplacer S par S_t.
          sinon
            si $\nexists S \in \mathcal{S}_t$ tel que $S \trianglerighteq S_t$ **alors**
              Ajouter S_t à $\mathcal{S}_t$.
              pour tout $S \in \mathcal{S}_t$ tel que $S_t \trianglerighteq S$ **faire**
                Retirer S de $\mathcal{S}_t$.
              fin pour
            fin si
          fin si
        fin si
      fin pour
    fin pour
  fin pour
  renvoie $\min_{S_T \in \mathcal{S}_T} f(S_T)$

2.3 Programmation dynamique approchée

En pratique la programmation dynamique est vite trop coûteuse en terme de temps de calcul et d'utilisation mémoire pour pouvoir être appliquée. En effet, le nombre d'états du graphe d'états est faiblement majoré par :

$$T \times \prod_{i=1}^{N} (T_{min,i}^{on} + T_{min,i}^{off} + T_{cs,i})$$

et peut donc être extrêmement grand. Les méthodes de programmation dynamique proposées dans la littérature tendent donc à introduire des stratégies de réduction de l'espace

d'états permettant d'obtenir une solution réalisable de bonne qualité en un temps raisonnable.

Pour ce faire, la stratégie la plus courante est d'utiliser une définition binaire des états qui ne modélise que l'état de marche ou d'arrêt des unités de production sans tenir compte du temps depuis lequel elles le sont. Ainsi, un état S_t^b est défini par une période de temps, t, et un vecteur binaire, $(S_{t,i}^b)_i$, tel que :

— $S_{t,i}^b = 0$ si l'unité i est éteinte au temps t.
— $S_{t,i}^b = 1$ si l'unité i est allumée au temps t.

L'idée est alors de ne plus associer un espace de décision avec un état S_t^b mais avec un chemin $(S_k^b)_{k\leq t}$. L'espace d'état $D((S_k^b)_{k\leq t})$ est alors l'ensemble des vecteurs de décision d vérifiant :

— Si $S_{t,i}^b = 1$ et $\sum_{k=t-T_{i,min}^{up}}^{t} S_{k,i}^b < T_{i,min}^{up}$ alors $d_i = 1$

— Si $S_{t,i}^b = 1$ et $\sum_{k=t-T_{i,min}^{up}}^{t} S_{k,i}^b = T_{i,min}^{up}$alors $d_i \in \{-1, 1\}$

— Si $S_{t,i}^b = 0$ et $\sum_{k=t-T_{i,min}^{off}}^{t} S_{k,i}^b > 0$ alors $d_i = -1$

— Si $S_{t,i}^b = 0$ et $\sum_{k=t-T_{i,min}^{off}}^{t} S_{k,i}^b = 0$ alors $d_i \in \{-1, 1\}$

— $\sum_{i,d_i>0} P_{min,i} \leq D_t$ et $\sum_{i,d_i>0} P_{max,i} \geq D_t + R_t$

Cette idée s'applique avec la stratégie de programmation dynamique par chaînage avant telle que décrite au paragrahe 2.2. Ainsi, à chaque étape les meilleurs chemins entre S_0^b et les états $S_t^b \in \mathcal{S}_t^b$ sont construits en se basant sur la connaissance des meilleurs chemins entre S_0^b et $S_{t-1}^b \in \mathcal{S}_{t-1}^b$.

Dans ce cadre la fonction de transition se re-définit très simplement comme suit :

$$T(S_t^b, d) = S_{t+1}^b, S_{t+1,i}^b = d_i \ \forall i$$

et la fonction coût se re-définit également en fonction du chemin menant à S_t^b :

$$R((S_k^b)_{k\leq t}, d) = \sum_{i=1}^{N} CS_i((S_k)_{k\leq t}, d) + CF(S_{t+1}^b)$$

Le calcul de CF reste inchangé par rapport à la programmation dynamique exacte et CS_i peut être reformulé comme ceci :

$$CS_i((S_k^b)_{k\leq t}, d) = \begin{cases} 0 & \text{si} \quad S_{t,i}^b = 1 \text{ ou } d_i = 0 \\ CS_{cold,i} & \text{si} \quad \sum_{k=t-T_{i,min}^{off}+T_{cs,i}}^{t} S_{k,i}^b > 0 \\ CS_{hot,i} & \text{sinon} \end{cases} \tag{4.7}$$

Cette stratégie permet de réduire considérablement l'espace de recherche, puisque le nombre maximal d'états devient :

$$T \times 2^N$$

Néanmoins l'optimalité de la solution n'est plus garantie car avec cette définition des états il est possible que le chemin optimal soit composé d'un sous-chemin non-optimal mais qui permette d'accéder à des décisions non accéssibles à partir du sous-chemin optimal. Ceci car les espaces de décisions sont définis par des sous-chemins et non plus des états. De plus, le nombre d'états reste exponentiel par rapport à la taille des données. Afin de réduire d'avantage le nombre d'états, la méthode de liste de priorité est généralement utilisée en complément. Cette méthode consiste à fixer un nombre maximum M d'états S_t possibles pour la période t et de sélectionner ces M états parmi les 2^N en utilisant une liste de priorité basée sur les propriétés des unités. Ainsi le nombre d'états maximal devient :

$$T \times M$$

De nombreuses méthodes de la littérature sont basées sur ces idées [72, 138, 166]. Certaines propositions ont été faites où au lieu de ne garder en mémoire que le "meilleur" chemin vers un état S_t, plusieurs chemins sont sauvegardés, ceci afin d'obtenir une solution plus proche de l'optimum global [72].

3 DYNAMOP pour le problème d'affectation d'unités

Nous allons maintenant montrer en détail comment DYNAMOP s'applique à la résolution du problème d'affectation d'unités.

3.1 Représentation et évaluation

La représentation et l'évaluation sont intimement liées aux choix qui sont faits quand à la définition du graphe d'états. La définition utilisée est celle donnée dans la section précédente. Ainsi un état est défini par un vecteur d'entier $(S_{t,i})_i$ où $S_{t,i}$ est défini dans l'intervalle $[\![-T^{off}_{i,min} - T_{cs,i}, T^{up}_{i,min}]\!] - \{0\}$ Nous avons cependant choisi de relâcher les contraintes de respect de la demande et de la réserve. Ainsi $D(S_t)$ est l'ensemble des vecteurs de décisions d vérifiant :

— Si $0 < S_{t,i} < T^{up}_{i,min}$ alors $d_i = 1$

— Si $S_{t,i} = T^{up}_{i,min}$ alors $d_i \in \{-1, 1\}$

— Si $-T^{off}_{i,min} < S_{t,i} < 0$ alors $d_i = -1$

— Si $-T^{off}_{i,min} \leq S_{t,i}$ alors $d_i \in \{-1, 1\}$

Ce choix est fait de façon à faciliter la construction de chemins entre deux états, mais il est alors nécessaire de pénaliser les décisions invalides dans la fonction coût, qui devient donc :

$$R(S_t, d) = \begin{cases} \sum_{i=1}^{N} CS_i(S_t, d) + CF(S_{t+1}) & \text{si } \sum_{i,d_i>0} P_{min,i} \leq D_t \text{ et } \sum_{i,d_i>0} P_{max,i} \geq D_t + R_t \\ M & \text{sinon} \end{cases} \tag{4.8}$$

Avec M est une constante de pénalité élevée.

3.2 Initialisation

Les chemins sont générés aléatoirement comme expliqué au chapitre 3. C'est à dire que partant de l'état S_0, on construit séquentiellement les états S_t, $t \geq 1$ en choisissant aléatoirement une décision d dans $D(S_{t-1})$ et en appliquant la relation de transformation $T(S_{t-1}, d)$. Cependant, lorsqu'il existe des décisions $d \in D(S_{t-1})$ respectant la contrainte relachée :

$$\sum_{i,d_i>0} P_{min,i} \leq D_t \text{ et } \sum_{i,d_i>0} P_{max,i} \geq D_t + R_t$$

celles ci sont choisies en priorité.

3.3 Mutation

La mutation utilisée est la mutation par bifurcation qui a été décrite au paragraphe 3.4 du chapitre 3. L'idée de cette mutation est de remplacer un état S_t par un autre état S'_t et de lier ce nouvel état au chemin en utilisant un nombre minimal d'arcs. Pour ce faire, dans le cas du problème d'affectation d'unités, les fonctions $cheminTransitoir1$ et $cheminTransitoir2$ sont utilisées. $cheminTransitoir1((S_k)_{k\leq t}, S'_t)$ renvoie le sous-chemin $(S^1_k)_{k\leq t}$ qui correspond au chemin liant liant l'état initial S^* et l'état S'_t ayant le maximum d'arcs en commun avec le chemin $(S_k)_{k\leq t}$. La fonction $cheminTransitoir2((S_k)_{k\leq t}, S'_t)$ renvoie le sous chemin $(S^2_k)_{k\leq t}$ qui lie l'état S'_t à un un état terminal et qui a le nombre maximal d'arcs en commun avec le chemin $(S_k)_{k\geq t}$. L'individu mutant est le chemin composé des chemins S^1 et S^2. Les algorithmes 3 et 4 décrivent ces fonctions en détails.

3.4 Croisements

Le croisement 2-transitions entre deux chemins (S^A et S^B) est réalisé grâce aux fonctions $cheminTransitoir1$ et $cheminTransitoir2$. Plus précisément, le croisement s'effectue en choisissant aléatoirement deux locus t_1 et t_2. Le premier individu est alors composé de $(S^{A,1}_t)_{t\leq t_1} = cheminTransitoir1((S^A_t)_{t<t_1}, S^B_{t_1})$ puis de $(S^B_t)_{t=t_1+1\ldots t_2-1}$ et enfin de $(S^{A,2}_t)_{t\geq t_2} = cheminTransitoir2(S^B_{t_2}, (S^A_t)_{t\geq t_2})$. Le deuxième individu est obtenu de la même façon mais en inversant les rôles de A et de B.

3.5 Opérateurs évolutionnaires intelligents

Nous avons utilisé trois opérateurs évolutionnaires intelligents. Tout d'abord, l'opérateur de croisement appelé *recombinaison de sous-politiques* présenté à la section 3.6 du chapitre 3. Nous avons également implémenté deux opérateurs de mutation basés sur la programmation dynamique. Ces opérateurs consistent tous les deux à chercher un meilleur chemin dans un graphe d'état restreint autour de la solution courante, mais ils diffèrent par la façon dont ce graphe d'états est défini.

Mutation hybride 1 :

Dans cette première hybridation, l'idée est de choisir aléatoirement une unité i et de calculer la planification la plus rentable pour cette unité en considérant comme fixée la planification des temps de marche et d'arrêt de toutes les autres unités.

Les éléments utilisés pour construire la méthode de programmation dynamique sont ceux utilisés pour la représentation des solutions de DYNAMOP présentés dans la section

Algorithme 3 cheminTransitoir1

Entrée : $(S_k)_{k\leq t}, S'_t$

Ensure: Le sous-chemin $(S^1_k)_{k\leq t}$ reliant l'état initial S^* et S'_t ayant le maximum d'arcs en commun avec le sous-chemin $(S_k)_{k\leq t}$

pour i=0...N **faire**
  $k = t - 1$
  trouve=faux
  $S^1 = S$
  si $S'_{t,i} > 0$ **et**$S_{t,i} \neq S'_{t,i}$ **alors**
    tant que non(trouve) **faire**
      si $S_{k,i} \leq T_{i,off}$ **et** $k \leq t - S'_{t,i}$ **alors**
        trouve = vrai
        pour $j, = k + 1...t - S'_{t,i}$ **faire**
          $S^1_j = -T_{i,off}$
        fin pour
        pour $j = 1...S'_{t,i}$ **faire**
          $S^1_{l+j,i} = j$
        fin pour
      fin si
      si $k \leq t - S'_{t,i} - T_{i,off}$ **et** $S_{k,i} = T_{i,up}$ **alors**
        trouve=vrai
        pour $j = 1...T_{i,off}$ **faire**
          $S^1_{k+j,i} = -j$
        fin pour
        pour $j = k + T_{i,off}...t - S'_{t,i}$ **faire**
          $S^1_{j,i} = -T_{i,off}$
        fin pour
        pour $j = 1...S'_{t,i}$ **faire**
          $S^1_{l+j,i} = j$
        fin pour
      fin si
      si $S'_{t,i} = T_{i,up}$ **et** $S_{k,i} = T_{i,up}$ **alors**
        trouve=vrai
        pour $j = k...t$ **faire**
          $S^1_{j,i} = T_{i,up}$
        fin pour
      fin si
      t=t-1
    fin tant que
  fin si
  si $S'_{t,i} < 0$ **et** $S_{t,i} \neq S'_{t,i}$ **alors**
    ... raisonnement symétrique
  fin si
  renvoie S^1
fin pour

Algorithme 4 cheminTransitoir2

Entrée : $S'_t, (S_k)_{k\geq t}$

Ensure: Le sous-chemin $(S^2_k)_{k\geq t}$ qui est le chemin reliant S'_t à un état terminal qui a la nombre maximal d'arcs en commun avec le sous-chemin $(S_k)_{k\geq t}$

pour i=0...N **faire**
  $k = t + T_{i,up} - S'_{t,i}$
  trouve=faux
  $S^1 = S$
  si $S'_{t,i} > 0$ **et** $S_{t,i} \neq S'_{t,i}$ **alors**
    pour $j = 1...T^{up}_{i,min} - S'_{t,i}$ **faire**
      $S^1_{t+j,i} = S'_{t,i} + j$
    fin pour
    l=$t + T^{up}_{i,min} - S'_{t,i}$
    tant que non(trouve) **faire**
      si $S_{k,i} = T^{up}_{i,min}$ **ou** $k = T$ **et** $S_{T,i} > 0$ **alors**
        trouve=vrai
        pour $j = t + T_{i,up} - S'_{t,i}...k$ **faire**
          $S_{j,i} = T_{i,up}$
        fin pour
      fin si
      si $S_{k,i} = -(T^{off}_{i,min} + T_{cs,i})$ **et** $k > t + T_{i,up} - S'_{t,i} + T^{off}_{i,min} + T_{cs,i}$ **ou** $S_{k,i} = 1$ **et** $k > t + T_{i,up} - S'_{t,i} + T^{off}_{i,min}$ **ou** $k = T$ **et** $S_{T,i} < 0$ **alors**
        trouve=vrai
        pour $j = 1...min(T^{off}_{i,min} + T_{cs,i}, k)$ **faire**
          $S^1_{l+j,i} = -j$
        fin pour
        pour $j = T^{off}_{i,min} + T_{cs,i}...k - 1$ **faire**
          $S^1_{l+j,i} = -(T^{off}_{i,min} + T_{cs,i})$
        fin pour
      fin si
    fin tant que
  fin si
  si $S'_{t,i} < 0$ **et** $S_{t,i} \neq S'_{t,i}$ **alors**
    ... (raisonnement symétrique)
  fin si
  renvoie S^1
fin pour

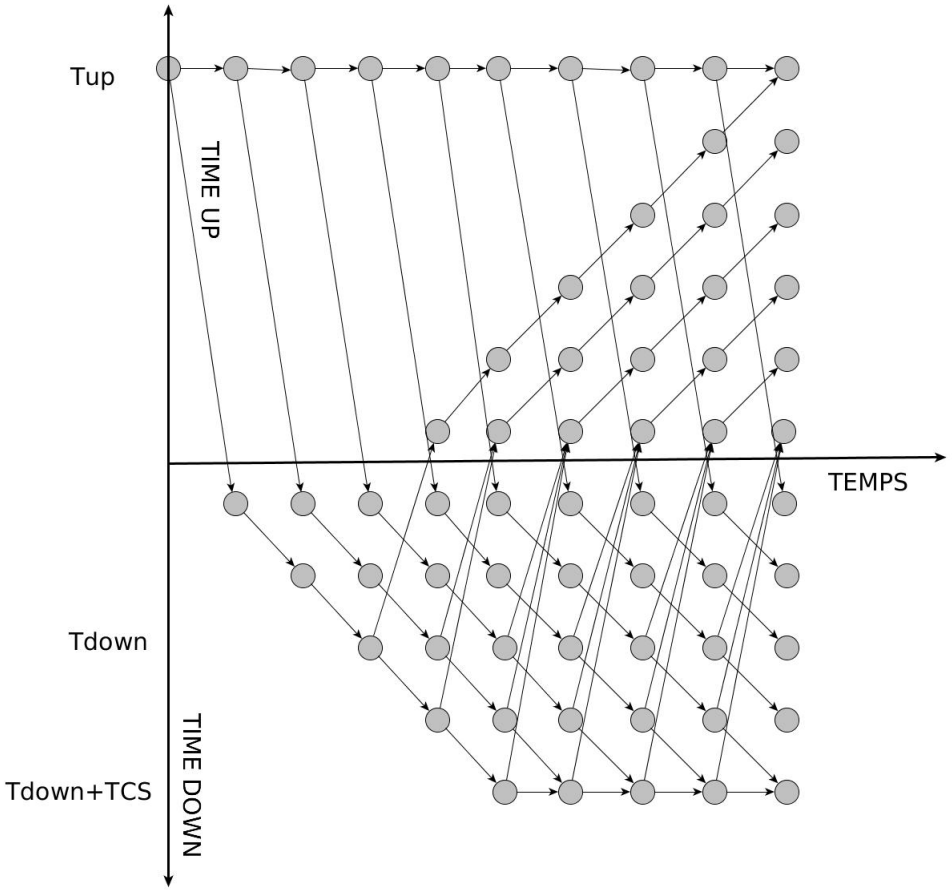

FIGURE 4.1 – Graphe d'états associé à la mutation hybride 1.

3.1 , excepté que l'on ne s'intéresse qu'aux états S_t appartenant à l'espace d'états restreint tel que si S^{or} est le chemin à muter alors :

$$S_{t,j} = S^{or}_{t,j} \quad \forall t,\ \forall j \neq i$$

Ainsi, un état associé à une période de temps t peut être représenté par un unique entier $S_{t,i}$ correspondant à l'unité i. De même, une décision n'est plus représentée par un vecteur mais par une unique valeur binaire représentant le choix d'allumer ou d'éteindre l'unité i à la prochaine étape de la planification. Le graphe d'états obtenu est très simple, la figure 3.5 représente ce graphe dans le cas où l'état initial de l'unité i est $T^{on}_{min,i}$ et où l'unité i a les propriétés suivantes :

- $T^{on}_{min,i} = 6$
- $T^{off}_{min,i} = 3$
- $T_{cs,i} = 3$

L'application de la programmation dynamique se fait similairement à l'algorithme 2, mais une coupe supplémentaire est utilisée. Celle ci a pour objectif d'écarter les solutions qui seraient forcément moins bonnes que la solution d'origine. Elle revient à supprimer à la fin de chaque étape t tous les états S_t pour lesquels :

$$f(S_t) \geq fit(S^{or})$$

Cela permet en particulier d'exclure tous les états pour lesquels il n'est pas possible de respecter les contraintes de demande et de réserve dans le cas où la solution d'origine est réalisable. Il est donc possible de récupérer le bénéfice de la contrainte relâchée dans la définition de $D(S)$ à partir du moment où l'on est assuré qu'il est possible de trouver une solution réalisable.

Mutation hybride 2 :

La seconde mutation permet de modifier un nombre N_u d'unités choisies aléatoirement. Nous noterons I_m, l'ensemble des unités choisies pour être modifiées.

L'objectif est de recalculer la planification des temps de marche et d'arrêt des N_u unités de manière optimale en considérant que les planifications de toutes les autres unités sont fixées. Cependant, comme ce calcul est très coûteux en terme d'utilisation mémoire et de temps de calcul, nous avons choisi d'utiliser un algorithme de programmation dynamique approché tel que décrit dans la section 2.3. Ainsi les états sont représentés sous forme de vecteurs binaires de taille N_u et l'espace de décisions associé à un état S_t^b est calculé en considérant le plus court chemin menant à cet état pour distinguer les décisions respectant les temps minimaux d'allumage et d'éteignage.

Cependant, afin de garantir l'obtention d'une solution de meilleure qualité que la solution d'origine, le chemin associé à celle-ci est systématiquement inclue dans le graphe d'états. C'est à dire que pour tout état S_t^b accessible depuis l'état $t-1$ du chemin d'origine, S^{or}, i.e. tel que $\forall i \in I_m$:

— Si $0 < S_{t-1,i}^{or} < T_{i,min}^{up}$ alors $S_{t,i}^b = 1$

— Si $S_{t-1,i}^{or} = T_{i,min}^{up}$ alors $S_{t,i}^b \in \{-1, 1\}$

— Si $-T_{i,min}^{off} < S_{t-1,i}^{or} < 0$ alors $S_{t,i}^b = -1$

— Si $-T_{i,min}^{off} \leq S_{t-1,i}^{or}$ alors $S_{t,i}^b \in \{-1, 1\}$

si la relation suivante est vérifiée :

$$f(S_t^b) > fit((S_k^{or})_{k \leq t-1}) + R(S_{t-1}^{or}, S_t^b)$$

alors la valeur de $f(S_t^b)$ est recalculée en considérant que S_{t-1}^{or} est l'état prédécesseur :

$$f(S_t^b) = fit((S_k^{or})_{k \leq t-1}) + R(S_{t-1}^{or}, S_t^b)$$
$$\text{et } S_k^b = S_k^{or,b} \quad \forall k < t$$

où $S^{or,b}$ est la version binaire de S^{or}.

De plus, comme dans la mutation hybride 1, on supprime à la fin de chaque étape t tous les états S_t^b pour lesquels :

$$f(S_t^b) \geq fit(S^{or})$$

4 Protocole expérimental

Dans cette section l'ensemble du protocole expérimental est présenté.

4.1 Données

DYNAMOP est testé sur des jeux de données issus de la littérature. Ces tests sont tous basés sur un système assez simple composé de 10 unités avec un horizon de planification de 24 heures. Les caractéristiques des dix unités de ce système sont données dans le tableau 4.1 et les demandes pour chaque heure sont données dans le tableau 4.2. Ces données proviennent de [89]. DYNAMOP est également testé sur des système composés de 20, 40, 60, 80 et 100 unités de production qui sont obtenus en dupliquant le cas de base. Pour ces jeux de données les demandes sont ajustées proportionnellement au nombre d'unités dans le système. Dans chaque cas une capacité de réserve correspondant à 10% de la demande est imposée.

Tableau 4.1 – Données pour le cas 10 unités.

unité	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
P_{max}**(MW)**	455	455	130	130	162	80	85	55	55	55
P_{min}**(MW)**	150	150	20	20	25	20	25	10	10	10
a	1000	970	700	680	450	370	480	660	665	670
b	16.19	17.26	16.6	16.5	19.7	22.26	27.74	25.92	27.27	27.79
c ($\times 10^{-5}$)	48	31	200	211	398	712	79	413	222	173
T_{min}^{up}	8	8	5	5	6	3	3	1	1	1
T_{min}^{down}	8	8	5	5	6	3	3	1	1	1
CS_{hot}	4500	50000	550	560	900	170	260	30	30	30
CS_{cold}	9000	10000	1100	1120	1800	340	520	60	60	60
T_{CS}	5	5	4	4	4	2	2	0	0	0
S_0	8	8	-5	-5	-6	-3	-3	-1	-1	-1

Tableau 4.2 – Demandes (en MW) sur un horizon de planification de 24 heures pour le cas 10 unités.

heure	1	2	3	4	5	6	7	8	9	10	11	12
demande	700	750	850	950	1000	1100	1150	1200	1300	1400	1450	1500
heure	13	14	15	16	17	18	19	20	21	22	23	24
demande	1400	1300	1200	1050	1000	1100	1200	1400	1300	1100	900	800

4.2 Paramétrage

Le paramétrage des algorithmes génétiques est une étape importante pour pouvoir en tirer les meilleurs performances. C'est pourquoi nous avons effectué des tests de comparaisons statistiques permettant d'étudier l'effet des taux d'application des différent opérateurs évolutionnaires ainsi que de la taille de la population et du choix de la stratégie de remplacement. Deux stratégies de remplacement sont testées. La première consiste à remplacer l'ensemble de la génération actuelle par la nouvelle génération en appliquant un élitisme faible. La seconde consiste à générer une nouvelle génération de N_o individus et d'appliquer un élitisme fort, c'est à dire de prendre les N meilleurs individus parmi les individus parents

et enfants. Il faut alors paramétrer le nombre d'individus N_o générés à chaque génération. Le nombre d'unités N_u dont le planning va être optimisé dans la seconde mutation hybride est également à paramétrer. Le critère d'arrêt utilisé durant ces tests est un temps maximal d'exécution correspondant à N secondes.

Cette analyse est effectuée grâce à Irace [113], un package de R [180]. Son principale objectif est d'automatiser la configuration d'un algorithme d'optimisation en cherchant le meilleur paramétrage étant donné un ensemble d'instances, ceci en utilisant des tests de comparaison statistique.

4.3 Comparaisons aux méthodes proposées dans la littérature scientifique

DYNAMOP est comparé à cinq autres métaheuristiques. La première est un algorithme génétique classique que nous avons implémenté en nous inspirant des articles [89, 185]. Cet algorithme est un algorithme génétique très simple qui peut être décrit par les éléments suivant :

— **Représentation :** L'algorithme manipule une population de solutions partielles associant à chaque unité un planning de temps de marche et d'arrêt. Une solution est représentée par un vecteur binaire u de taille $N \times T$, tel que les T premiers éléments donnent le planning de l'unité 1, les T suivants donnent le planning de l'unité 2 etc. La figure 4.2 illustre la représentation génotypique d'une solution.

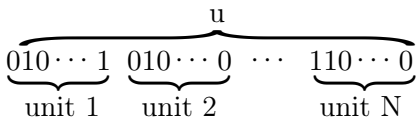

FIGURE 4.2 – Représentation génotypique d'une solution.

— **Opérateur de croisement** Deux croisements sont utilisés. Le premier est le très classique croisement 1-point et le second est un croisement 2-points un peu intelligent. Il consiste à sélectionner aléatoirement une portion du génotype (comprise entre les locus l_1 et l_2) et si le nombre de bits dans cette portion de génotype est plus petit que le nombre de bits à l'extérieur de cette portion, alors l'individu enfant est constitué des bits en provenance du parents de moins bonne valeur de fitness entre les locus l_1 et l_2 et de ceux de l'autre parent en dehors de ces locus. Le croisement s'effectue de la façon inverse si le nombre de bits dans la portion sélectionnée est plus grand que celui des bits restant en dehors de cette portion. Le croisement est illustré dans la figure 4.3.

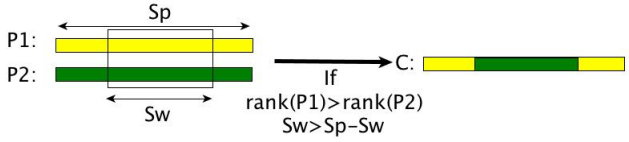

FIGURE 4.3 – Croisement deux points intelligent.

— **Opérateur de mutation** Deux opérateurs de mutation sont utilisés. Le premier est l'opérateur standard *1-bit-flip mutation*, qui inverse la valeur d'un bit choisi aléatoirement. Le deuxième choisit aléatoirement une portion du génotype et inverse

la valeur de tous les bits compris dans cette portion. Cette mutation est appelée *window-mutation*.

— **Opérateur de réparation** L'objectif de cet opérateur est de corriger une solution si elle ne respecte pas les contraintes de respect de la demande et de la réserve ou les contraintes de temps minimal de marche ou d'arrêt. La première étape effectuée par cette opérateur est de corriger les contraintes violées de temps minimal de marche ou d'arrêt en modifiant heure par heure l'état des unités les violant. Ensuite, des corrections sont faites pour diminuer le nombre des contraintes de respect de la demande et de la réserve violées. Ceci est fait heure par heure : Si la capacité maximale de production des unités allumées est plus petite que la demande plus la réserve, alors on regarde si une unité éteinte peut être allumée sans violer la contrainte de temps minimale d'arrêt et si une telle unité existe, on l'allume. Le procédé inverse est effectué si la somme des capacités minimales des unités allumées excède la demande.
Suite à cet opérateur, la solution obtenue ne sera pas forcément valide, néanmoins cet opérateur permet d'accélérer considérablement la découverte d'une solution réalisable par l'algorithme.

Par la suite cet algorithme sera dénoté *BGA*. Les quatre autres métaheuristiques auxquels nous nous comparons sont celles de la littérature qui fournissent les meilleurs résultats pour nos instances. Il s'agit de IQEA [84] ,QEA-UC [101], QBPSO [83] ,hGADE [183] et HBRKGA [146]. IQEA et QEA-UC sont des algorithmes évolutionnaires quantique, QBPSO est un algorithme d'essaim particulaire inspiré de la programmation quantique, hGADE est une méthode hybride combinant un algorithme génétique à un algorithme évolutionnaire différentiel et HBRKGA est un algorithme génétique utilisant un système d'encodage/décodage sophistiqué et hybridé avec un algorithme de recherche locale. Néanmoins ces algorithmes étant très complexes nous ne les avons pas ré-implémentés et nous nous comparerons donc seulement aux résultats fournis dans les articles correspondant.

Nous comparerons également nos résultats avec les résultats optimaux qui ont été trouvés pour ces instances par Viana *et al.* [190] et par Rahman *et al.* [145].

5 Résultats expérimentaux

Cette section présente les résultats des différents tests de comparaisons statistiques.

5.1 Influence des opérateurs évolutionnaires intelligents

Nous présentons ici, une étude de l'impact de chacun des opérateurs évolutionnaires intelligents proposé pour DYNAMOP (i.e la recombinaison de sous-politiques et les deux mutations hybrides). Cette étude se base sur la comparaison statistique de plusieurs versions de DYNAMOP qui diffèrent de par les opérateurs intelligents qu'elles utilisent. Chacune de ces versions est basée sur le même paramétrage (celui obtenu avec Irace), excepté que le taux d'application de certains opérateurs est ramené à zéro. Dans les tableaux 4.4 et 4.5, les opérateurs évolutionnaires intelligents appliqués sont signifiés par "R" pour l'opérateur de recombinaison de sous-politiques, "H1" pour la première mutation hybride et "H2" pour la seconde. Si aucun opérateur évolutionnaire intelligent n'est appliqué ceci est signifié par le signe "$\emptyset$". Pour chaque type de paramétrage et pour chaque jeu de donné, DYNAMOP est exécuté 20 fois avec les mêmes populations initiales en utilisant comme critère d'arrêt un temps d'exécution maximal qui correspond au temps de convergence moyen de DYNAMOP si tous les opérateurs évolutionnaires intelligents proposés sont utilisés.

Influence de l'opérateur de recombinaison de sous-politique

Le tableau 4.4 résume les résultats obtenus avec une version de DYNAMOP où aucun opérateur évolutionnaire intelligent n'est utilisé ainsi que les résultats obtenus avec une version de DYNAMOP où l'opérateur *recombinaison de sous-politiques* est utilisé. Les colonnes "-" donnent les plus petits coûts de production sur les 20 exécutions, les colonnes "Moy.", le coût moyen sur les 20 exécutions et les colonnes "+" donnent, pour chaque paramétrage, la valeur de la solution la plus coûteuse trouvée par la version de DYNAMOP concernée. Nous pouvons observer, que pour chaque cas d'étude, l'utilisation de l'opérateur de recombinaison de sous-politiques améliore la qualité des résultats. En effet, pour le cas 10 unités l'écart relatif entre les meilleures solutions obtenues par les deux versions est de 0.09%, pour le cas d'étude de 20 unités il est de 0.26%, de 0.02% dans le cas de 40 unités, de 0.27% pour le système de 80 unités et de 1.36% pour le système de 100 unités. Pour le cas d'étude de 60 unités l'amélioration apportée par cet opérateur de croisement intelligent est considérable puisque l'on peut observer un écart de 9.32%. Une analyse statistique basée sur le test de Friedman permet d'ailleurs d'affirmer que l'influence de l'opérateur recombinaison de sous-politiques est statistiquement significative pour tous les cas d'étude avec une valeur p de 0.01.

Influence des opérateurs hybrides

Le tableau 4.5 résume les résultats obtenus en utilisant DYNAMOP uniquement avec l'opérateur de recombinaison de sous-politiques ou bien avec l'une ou/et l'autre des mutations hybride.

La comparaison des résultats obtenus en n'utilisant que l'opérateur de recombinaison de sous-politiques à ceux obtenus en utilisant en plus la mutation hybride 1, notée H1, donne un bilan mitigé. En effet, à première vue, si ce n'est pour le cas de l'instance de 80 unités, les résultats obtenus en utilisant H1 sont moins bons que ceux obtenus en ne l'utilisant pas. L'analyse statistique montre, qu'effectivement, pour les instances de 20 et 60 unités l'utilisation de H1 avec l'opérateur de recombinaison de sous-politiques est significativement désavantageuse vis à vis d'une utilisation simple de l'opérateur de recombinaison. Ceci pourrait être du au fait de son coût en temps de calcul ou au fait qu'elle puisse pousser la convergence vers un minimal local. Par contre, pour les instances de 10, 40 et 100 unités, l'ajout de H1 n'a pas d'impact statistiquement significatif sur les résultats et pour le cas de 80 unités son impact est statistiquement significativement positif.

Lorsque nous observons les résultats obtenus en n'utilisant que l'opérateur de recombinaison de sous-politiques à ceux obtenus en utilisant en plus la mutation hybride 2, notée H2, nous pouvons constater l'influence positive de H2. Cette observation est confirmée par une analyse statistique qui montre que pour les instances de 10, 20, 40 et 80 unités la différence entre ces deux versions est significative avec une p-value inférieure à 0.01. En revanche la différence n'est pas significative pour les instances de 60 et de 100 unités.

Finalement, si l'on compare DYNAMOP muni de tous les opérateurs évolutionnaires intelligents que nous avons proposé à DYNAMOP avec uniquement l'opérateur de recombinaison de sous-politiques, on peut noter que la différence est importante, en particulier quand le nombre d'unités est grand, et ceci en faveur de la version hybride. De plus les tests statistiques effectués confirment la significativité de ces différences. De plus l'utilisation combinée des deux mutations hybrides donne, dans tous les cas d'étude, des résultats statistiquement meilleurs que ceux obtenus avec uniquement l'hybridation 1. L'utilisation combinée des deux mutations hybrides donne également des résultats statistiquement

Tableau 4.3 – Solutions optimales tirées de [190] et [145]

N	**10**	**20**	**40**	**60**	**80**	**100**
Solution ($)	563 938	1 123 297	2 242 596	3 359 955	4 480 324	5 597 770
Temps (s)	0.58	16.4	711	271	1499	2219

Tableau 4.4 – Influence de l'opérateur de recombinaison. La colonne R correspond aux résultats obtenus avec l'opérateur de recombinaison et la colonne ∅ aux résultats obtenus si cet opérateur n'est pas utilisé.

	R			∅		
N	-	Moy.	+	-	Moy.	+
10	**563 938**	568 496	578 010	564 439	**567 666**	**572 779**
20	**1 123 970**	**1 125 915**	**1 129 045**	1 126 874	1 129 364	1 131 759
40	**2 270 554**	**2 285 978**	**2 298 425**	2 271 091	**2** 771 012	7 171 217
60	**3 426 341**	**3 687 555**	**3 786 189**	3 778 525	3 784 097	3 795 330
80	**4 547 999**	**4 585 980**	**4 626 699**	4 560 116	6 242 628	14 147 650
100	**5 708 001**	**5 726 600**	**5 741 238**	5 787 172	5 812 008	5 824 192

meilleurs que ceux obtenus avec uniquement l'hybridation 2 pour les systèmes de 20 et 60 unités, pour les autres cas, en revanche, l'ajout de H1 n'améliore pas significativement les résultats. Il est à noter, que pour les deux cas d'études de 20 et 60 unités, pour lesquels la combinaison de R, l'opérateur de recombinaison de sous-politiques et de H1 est moins performante que l'utilisation unique de R, l'ajout de H1 à RH2 a en revanche une influence significativement poisitive.

Cette étude nous a donc permis de démontrer la capacité de chacun des opérateurs évolutionnaires intelligents proposés à influer positivement sur les résultats fournis par DYNAMOP. Dans la suite de ce chapitre, nous utiliserons, pour DYNAMOP, les paramétrages les plus efficaces, en incluant tous ces opérateurs évolutionnaires.

Tableau 4.5 – Influence des opérateurs hybrides. Les opérateurs intelligents utilisés sont signifiés par des lettres, "R" pour l'opérateur de recombinaison de sous-politiques, "H1" pour la première mutation hybride et "H2" pour la seconde.

	RH1H2			RH1			RH2			R		
N	-	Moy.	+	-	Moy.	+	-	Moy.	+	-	Moy.	+
10	**563 938**	**563 964.8**	**564 206**	564 614	567 190	571424	**563 938**	563 977	564 359	**563 938**	568 496	578 010
20	**1 123 297**	1 123 418	1 124 370	1 126 599	1 129 885	1 132 108	**1 123 297**	**1 123 323**	**1 123 806**	1 123 970	1 125 915	1 129 045
40	**2 242 596**	**2 243 259**	**2 244 430**	2 265 503	2 278 167	2 285 983	2 242 596	2 243 312	2 244 353	2 270 554	2 285 978	2 298 425
60	**3 360 320**	3 361 294	3 364 030	3 778 104	3 782 173	3785100	3 775 526	3 775 830	3776594	3 426 341	3 687 555	3 786 189
80	4 480 630	4 481 828	4 483 730	4 544 519	4 561 580	4 577 336	**4 480 328**	**4 481 417**	**4 483 090**	4 547 999	4 585 980	4 626 699
100	**5 599 150**	**5 601 392**	**5 603 170**	5 786 026	5 835 825	5 905 216	5 601 263	5 604 636	5 608 242	5 708 001	5 726 600	5 741 238

5.2 Comparaison à l'algorithme génétique basique

Le tableau 4.6, compare les résultats obtenus par DYNAMOP à ceux obtenus par BGA. La colonne "dif." donne le pourcentage de différence entre les meilleurs résultats fournis par les deux algorithmes en utilisant la formule :

$$\frac{meilleur_{DYNAMOP} - meilleur_{BGA}}{meilleur_{BGA}} \times 100$$

En considérant cette colonne nous pouvons immédiatement noter que leur performance sont très différentes. Une analyse statistique a été effectuée et permet de confirmer que DYNAMOP est statistiquement meilleur que BGA.

Tableau 4.6 – Comparaison avec l'algorithme génétique basique.

Methode	Meilleur coût ($)	Coût moyen ($)	Pire coût ($)	dif. (%)
10 unités				
BGA	**563 938**	566 112	568 825	
DYNAMOP	**563 938**	563 964.8	564 206	0
20 unités				
BGA	1 128 280	1 134 992	1 143 110	
DYNAMOP	**1 123 297**	1 123 418	1 124 370	**-0.44**
40 unités				
BGA	2 287 463	2 391 314	2 813 7601	
DYNAMOP	2 242 596	2 243 259	2 244 430	**-2**
60 unités				
BGA	3 430 710	3 944 848	5 099 610	
DYNAMOP	**3 360 320**	3 361 294	3 364 030	**-2.09**
80 unités				
BGA	4 891 180	5 025 740	6 040 500	
DYNAMOP	**4 480 328**	**4 481 417**	**4 483 090**	**-8.39**
100 unités				
BGA	5 848 670	6 207 561	6 956 581	
DYNAMOP	**5 599 150**	5 601 392	5 603 170	**-4.45**

5.3 Comparaison aux meilleurs résultats de la littérature

Une étude comparative est effectuée avec les algorithmes trouvés dans la littérature qui donnent les meilleurs résultats. Ces algorithmes, IQEA [84] ,QEA-UC [101], QBPSO [83], hGADE [183] et HBRKGA [146] ont été présentés dans la section 4.3.

Le Tableau 4.7 résume les résultats obtenus pour chaque jeu de données pour 20 exécutions. La colonne "Meilleur" donne la meilleure solution obtenue lors des 20 exécutions. La colonne "Moyenne" indique la moyenne de l'ensemble des résultats obtenus, la valeur de la pire solution obtenue est donnée dans la troisième colonne. Dans la colonne "dif." l'écart entre la meilleur solution trouvée par une métaheuristique de la littérature M_{connue} pour le jeu de données et la meilleure solution obtenue par la méthode de la ligne correspondante M_{trouve} est indiqué. Cet écart est calculé avec la formule suivante :

$$\frac{M_{connue} - M_{trouve}}{M_{connue}}$$

La colonne "Ecart" donne un pourcentage mesurant l'écart avec la solution optimale. Les solutions optimales sont tirées des articles [145, 190]. Le tableau 4.3 donne ces valeurs optimales ainsi que les temps de calculs nécessaire pour les obtenir les plus court parmi ceux donnés dans les articles. Enfin, la colonne temps donne les temps d'exécution fournis dans les différent articles, ces temps sont donnés à titre indicatif, les machines étant différentes. Pour DYNAMOP le temps donné correspond au temps moyen que prend l'algorithme pour converger, en considérant qu'il y a convergence après 100 générations sans évolution de la meilleure solution de la population. On peut remarquer à partir du tableau que DYNAMOP permet d'obtenir de meilleures solutions que toutes les autres métaheuristiques. On peut trouver dans la littérature certaines métaheuristiques donnant de meilleurs résultats sur certaines instance telles [29, 147], mais comme ces résultats sont meilleurs que la solution optimale nous pouvons en déduire que certaines contraintes sont violées. La dernière colonne donne le temps moyen avant convergence des différents algorithmes. Ces temps ne peuvent qu'être utilisés comme une base indicative car les résultats ont directement été récupérés des articles dans lesquels ils ont été décrits et donc ces temps sont obtenus à partir de machines différentes. Néanmoins, cela permet de voir que les temps de calculs obtenus avec DYNAMOP semblent bons par rapport à ceux des autres métaheuristiques. Si on compare DYNAMOP avec l'approche exacte, on peut constater que les résultats sont très proches puisque l'écart est toujours inférieur à 0.025%. Encore une fois il est délicat de comparer les temps d'exécution, cependant nous pouvons dire que sur les petites instances il est préférable d'utiliser l'approche exacte car elle est très rapide. Néanmoins, les temps de calcul de l'approche exacte explosent avec le nombre d'unités. DYNAMOP présente également l'avantage de s'adapter facilement en cas de changement dans la fonction évaluation. Par exemple, certaines modélisations du problème prennent en compte des coûts de démarrage évalués par une fonction faisant intervenir une composante exponentielle ou sinusoïdale [186], ce changement serait très simple à prendre en compte avec DYNAMOP mais compliquerait beaucoup la résolution exacte. Nous verrons également par la suite que DYNAMOP peut s'adapter à la résolution de la version multi-objectif de ce problème.

6 Conclusion

Dans ce chapitre nous avons appliqué DYNAMOP à un problème académique très étudié qui est le problème d'affectation d'unités. Nous l'avons testé sur des instances très répandues dans la littérature et dont certaines ont été résolues de manière exacte. Nous avons donc pu montrer que sur ce problème DYNAMOP permet d'obtenir des résultats très proches des solutions optimales avec un temps de calcul qui augmente très lentement avec le nombre d'unités considérées.

D'autre part nous avons comparé DYNAMOP aux métaheuristiques de la littérature donnant les meilleurs résultats et nous avons pu constater que DYNAMOP permet toujours d'obtenir des solutions de moindre coût ou équivalentes.

DYNAMOP est donc efficace pour résoudre le problème d'affectation d'unités.

Néanmoins, sur la majorité des instances étudiées, une méthode de résolution exacte a été proposée permettant d'obtenir des solutions en des temps qui restent raisonnables. C'est pourquoi nous pensons qu'il serait intéressant d'exploiter DYNAMOP sur des instances plus complexes et surtout plus proches de cas d'applications réels. En ce sens, plusieurs pistes peuvent être envisagées. Nous pouvons tout d'abord considérer des modélisations plus réalistes des coûts de production, en effet de nombreux travaux ne portent que sur le sous-problème de distribution économique en s'attachant à une modélisation plus réaliste des coûts de production prenant en compte une composante sinusoïdale [199]. D'autre

Tableau 4.7 – Comparaison entre DYNAMOP et les meilleurs métaheuristiques proposées dans la littérature scientifique.

Methode	Meilleur ($)	Moyenne ($)	Pire ($)	dif. (%)	Ecart (%)	Temps (s)
10 unités						
IQEA	563 977	563 977	**563 977**	-0.0069	-0.0069	15
QEA-UC	**563 938**	563 969	564 672	**0**	**0**	19
QBPSO	563 977	563 977	**563 977**	-0.0069	-0.0069	18
hGADE	**563 938**	**563 942**	**563 977**	**0**	**0**	24
HBRKGA	**563 938**	564 062	564 737	**0**	**0**	2
DYNAMOP	**563 938**	563 964.8	564 206	**0**	**0**	11
20 unités						
IQEA	1 123 890	1 124 320	1 124 504	-0.053	-0.053	42
QEA-UC	1 123 607	1 124 689	1 125 715	-0.027	-0.027	28
QBPSO	**1 123 297**	1 123 981	**1 124 294**	**0**	**0**	50
hGADE	1 123 386	1 124 262	1 124 939	-0.0079	-0.0079	56
HBRKGA	1 123 955	1 124 213	1 125 048	-0.0586	-0.0586	14
DYNAMOP	**1 123 297**	**1 123 418**	1 124 370	**0**	**0**	37
40 unités						
IQEA	2 245 151	2 246 026	2 246 701	-0.0978	-0.115	132
QEA-UC	2 245 557	2 246 728	2 248 296	-0.1453	-0.133	43
QBPSO	2 242 957	2 244 657	2 245 941	0	-0.017	158
hGADE	2 243 522	2 245 020	2 246 487	-0.0252	-0.0412	147
HBRKGA	2 244 345	2 245 350	2 245 775	-0.062	-0.078	90
DYNAMOP	**2 242 596**	**2 243 259**	**2 244 430**	**0.0162**	**0**	45
60 unités						
IQEA	3 365 003	3 365 667	3 366 223	-0.09	-0.1502	273
QEA-UC	3 366 676	3 368 220	3 372 007	-0.139	-0.2	54
QBPSO	3 361 980	3 363 763	3 365 707	0	-0.0602	328
hGADE	3 362 908	3 368 558	3 367 820	-0.027	-0.0879	326
HBRKGA	3 363 804	3 365 201	3 366 773	-0.0543	-0.1146	301
DYNAMOP	**3 360 320**	**3 361 294**	**3 364 030**	**0.04938**	**-0.0109**	88
80 unités						
IQEA	4 486 963	4 487 985	4 489 286	-0.1088	-0.1482	453
QEA-UC	4 488 470	4 490 128	4 492 839	-0.1425	-0.1818	66
QBPSO	4 482 085	4 485 410	4 487 168	0	-0.0393	554
hGADE	4 485 160	4 487 293	4 489 114	-0.0686	-0.1079	404
HBRKGA	4 485 197	4 487620	4 488 962	-0.06943	-0.1088	712
DYNAMOP	**4 480 328**	**4 481 417**	**4 483 090**	**0.039**	**-8.92×10^{-5}**	133
100 unités						
IQEA	5 606 022	5 607 561	5 607 561	-0.0631	-0.1474	710
QEA-UC	5 609 550	5 611 797	5 613 220	-0.1261	-0.21	80
QBPSO	5 602 486	5 604 275	5 606 178	0	-0.0842	833
hGADE	5 604 787	5 607 487	5 612 131	-0.0411	-0.1253	476
HBRKGA	5 605 933	5 607 024	5 608 559	-0.0615	0.1458	1259
DYNAMOP	**5 599 150**	5 601 392	5 603 170	**0.0594**	**-0.0246**	166

travaux ont été menés dans lesquels les coûts de démarrage sont modélisés de façon plus complexe pour concorder mieux aux observation faites sur des cas concrets [187]. Nous pouvons également considérer une modélisation stochastique prenant en compte le fait que la demande n'est pas forcément connue à l'avance ou que les capacités de production peuvent varier en fonction de facteurs stochastiques [172]. Enfin, il serait intéressant de considérer une modélisation multi-objectif du problème, dans laquelle le problème des émissions de gaz à effet de serre serait pris en compte. Cette dernière possibilité sera étudiée dans la seconde partie de cette thèse.

Chapitre 5

DYNAMOP appliqué à un cas réel : La planification de l'utilisation des eaux dans un réseau hydro-électrique

Dans ce chapitre, DYNAMOP est appliqué à un problème d'application réel, qui est le problème de planification des débits d'eau dans un réseau de production d'énergie hydro-électrique. Les données que nous utilisons nous ont été fournies par une entreprise, ce qui nous permet d'évaluer le potentiel de DYNAMOP sur des jeux de donnés réels.

Les travaux exposés dans ce chapitre ont été présentés dans deux conférences internationales :

- *EURO 2013 à Rome, en Italie ;*
- *EVO* 2014 à Grenade, en Espagne (publication LNCS) [77].*

1 Introduction

Le problème de planification de l'utilisation des eaux dans un réseau hydro-électrique tel qu'il a été formulé au chapitre 2 est un problème particulièrement complexe. Premièrement, comme démontré dans le chapitre 2, ce problème est un problème NP-complet. Les algorithmes de résolution exacte qui ont été proposés pour ce type de problème sont très coûteux en temps de calcul et ne s'appliquent donc que sur des instances de petite taille. De plus, ce problème est également très difficile à aborder avec des métaheuristiques. En effet, les dépendances entre les décisions sont très fortes et complexes, aussi bien du point de vue contraintes que du point de vue objectif. Il est donc difficile de produire de petits changements dans une solution sans entraîner des violations de contraintes ou en restant dans le voisinage de la solution initiale d'un point de vue objectif. Ceci rend délicat la proposition d'opérateurs de parcours de l'espace de recherche efficaces lorsque l'on met en place une métaheuristique.

Cependant comme ce problème de planification vérifie la propriété de Bellman, l'application de DYNAMOP est possible et nous pensons qu'elle peut donner de bons résultats. D'une part, DYNAMOP permet d'utiliser des méthodes de résolutions exactes sur des sous-problèmes de plus petite taille et d'améliorer ainsi la qualité de la convergence par rapport à une métaheuristique classique sans que ce ne soit trop coûteux en terme de temps de calcul. D'autre part, la représentation sous forme de séquence d'états et les opérateurs utilisés dans DYNAMOP prennent en compte les dépendances entre les variables de décisions.

Nous proposerons donc, dans ce chapitre, une application de DYNAMOP pour le problème de planification des débits d'eau dans un réseau hydro-électrique et nous comparerons

la méthode ainsi obtenue avec un algorithme génétique classique.

Dans une première section nous expliquerons comment le problème peut être traité par la méthode de programmation dynamique. Ceci nous permettra d'aborder plus aisément la mise en application de DYNAMOP sur ce problème dans la section suivante. Nous consacrerons ensuite une section à détailler le protocole expérimental et l'algorithme génétique auquel nous nous comparerons avant de présenter les résultats et de conclure en nous basant sur une analyse de ceux-ci.

Les notations utilisées dans ce chapitre sont celles introduites dans la section 3 du chapitre 2.

2 La programmation dynamique appliquée au problème de planification des eaux dans un réseau hydro-électrique

Dans cette section, nous expliquons comment le problème de planification des eaux dans un réseau hydro-éléctrique peut théoriquement être résolu de façon exacte par la méthode de programmation dynamique. Cela nous permettra de comprendre comment définir un graphe d'états pour ce problème et donc d'aborder plus facilement la question de la représentation lorsque nous expliquerons l'application de DYNAMOP à ce problème.

2.1 Définition des éléments de la programmation dynamique

La méthode de programmation dynamique est définie par son équation fonctionnelle comme expliqué dans la section 2.2 du chapitre 1. Nous en détaillons ici les différents éléments.

Espace d'états

L'état du système hydro-électrique à un instant t est totalement caractérisé par les quantités d'eaux $V_{i,t}$ contenues dans ses réservoirs. Ces informations suffisent à déterminer les décisions pouvant être prises à l'instant $t+1$ et leurs impacts sur le profit annuel, elles peuvent donc servir de base à la définition des états de la programmation dynamique. Cependant, comme notre étude se limitera à des réseaux de type arborescent, tels ceux de la figure 2.2, dans lesquels les réservoirs ont au plus un successeur, nous choisirons plutôt de décrire les états par rapport au volume total d'eau $q_{i,t}^{tot}$ ayant été déversé de chaque réservoir i au temps t. Plus précisément, pour tout réservoir i, $q_{i,t}^{tot}$ est défini comme ceci :

$$q_{i,t}^{tot} = \sum_{k<t} \sum_{j \in \delta_i^-} q_{j,k}$$

Ces informations sont équivalentes, puisqu'il est facile de connaitre les $V_{i,t}$ à partir des $q_{i,t}^{tot}$ à l'aide de la formule suivante :

$$V_{i,t} = V_{i,init} + \sum_{k<t} AN_{i,k} - q_{i,t}^{tot} + \sum_{j \in \delta_i^+} q_{j,t}^{tot}$$

En revanche, calculer $q_{i,t}^{tot}$ à partir des $V_{i,t}$ implique un calcul récursif. C'est à dire qu'avant de pouvoir calculer la valeur de $q_{i,t}^{tot}$ il faut calculer les valeurs des $q_{j,t}^{tot}$, pour tout réservoir j situé en amont de i. Ce choix présente également deux autres intérêts qui nous seront pratiques :

— Il facilite la restriction de l'espace d'états : En effet, la contrainte (5) :

$$V_{i,T} = V_{i,init} \quad \forall i$$

peut se réécrire sous la forme :

$$q^{tot}_{i,T} = \sum_{k<T} AN_{i,k} + \sum_{j\in\delta^+_i} \sum_{k<T} AN_{j,k}$$

Ainsi à l'instant t, la borne supérieure suivante doit être respectée afin qu'il soit possible d'obtenir un planing respectant à la fois la contrainte (5) de volume final et les contraintes (2) de débits minimaux :

$$q^{tot}_{i,t} \leq \sum_{k<t} AN_{i,k} + \sum_{j\in\delta^+_i} \sum_{k<t} AN_{j,k} - \sum_{j\in\delta^+_i} t \times r_{min,j,k} \tag{5.1}$$

Par la suite, on notera $q^{tot}_{max,i,t}$ cette borne supérieure. De même la borne inférieure suivante doit être respectée :

$$q^{tot}_{i,t} \geq \sum_{k<t} AN_{i,k} + \sum_{j\in\delta^+_i} \sum_{k<t} AN_{j,k} - \sum_{j\in\delta^+_i} t \times r_{max,j,t} - \sum_{j\in\delta^+_i} t \times q_{max,j,t} \tag{5.2}$$

Ceci afin qu'il soit possible d'atteindre l'état final en respectant les débits maximaux au cours des périodes de temps à venir. Par la suite, on notera $q^{tot}_{min,i,t}$ cette borne inférieure.

— Ce choix de modélisation nous permet également de facilement déterminer la fonction T^{-1}, qui est la fonction inverse de la fonction de transformation. En effet étant donnés deux états $S_t = (q^{tot}_{i,t})_i$ et $S_{t_1} = (q^{tot}_{i,t+1})_i$, il est facile de déterminer le vecteur de décision $d_t = (q_{i,t})_i$, donnant les quantités débitées de chaque réservoir au temps t, car celui ci s'obtient par la différence des deux vecteurs d'états :

$$d_t = S_{t_1} - S_t$$

Ainsi l'espace d'états est défini par l'ensemble des vecteurs $S_t = (q^{tot}_{t,i})_i$ pour lesquels les conditions 5.1 et 5.2 sont vérifiées et tels que les contraintes de capacités des réservoirs soient respectées, c'est à dire tels que :

$$V_{min,t} \leq V_{i,init} + \sum_{k<t} AN_{i,k} - q^{tot}_{i,t} + \sum_{j\in\delta^+_i} q^{tot}_{j,t} \leq V_{max,t} \tag{5.3}$$

Espace de décisions

L'espace des décision $D(S_t)$ correspond à l'ensemble des décisions qui peuvent être prises à partir de l'état $S_t = (q^{tot}_{i,t})_i$ et permettent d'atteindre un état viable, c'est à dire l'ensemble des décision $d_t = (q_{i,t})_i$ telles que :

$$q^{tot}_{i,t} + q_{i,t} \leq q^{tot}_{max,i,t} \quad \forall i,t \tag{5.4}$$

$$q^{tot}_{i,t} + q_{i,t} \geq q^{tot}_{min,i,t} \quad \forall i,t \tag{5.5}$$

$$V_{min,t+1} \leq V_{i,init} + \sum_{k<t} AN_{i,k} - q^{tot}_{i,t} - q_{i,t} + \sum_{j\in\delta^+_i} (q^{tot}_{j,t} + q_{j,t}) \leq V_{max,t+1} \quad \forall i,t \tag{5.6}$$

de plus afin de garantir le respect de la contrainte de débit minimal sur les conduites, $q_{i,t}$ doit vérifier :

$$q_{i,t} \geq \sum_{j\in\delta^+_i} r_{min,j,k} \tag{5.7}$$

En outre, afin de discrétiser l'espace d'états, on ne considère que les décisions pouvant s'écrire sous la forme :

$$q_{i,t} = \sum_{j \in \delta_i^+} r_{min,j,k} + \sum_{j \in \delta_i^+} q_{min,j} + k \times p_i \tag{5.8}$$

où p_i est un pas de discrétisation fixé associé au réservoir i et k est un entier quelconque.

Fonction de transformation

La fonction de transformation s'exprime simplement par :

$$T(S_t, d_t) = S_{t+1} = (S_{t,i} + d_{t,i})_i$$

États terminaux

Le seul état terminal valide est $S_T = (q_{T,i}^{tot})_i$, tel que :

$$q_{i,T}^{tot} = \sum_{k<T} AN_{i,k} + \sum_{j \in \Delta_i^+} \sum_{k<T} AN_{j,k} \ \forall i,$$

avec Δ_i^+ l'ensemble des réservoirs déversant dans le réservoir i.

Tout autre état terminal est invalide.

Fonction coût

La fonction coût peut s'exprimer comme ceci :

$$R(S_t, d_t) = \sum_{i=1}^{N} R_i(V_{i,t}, q_{t,i}), \tag{5.9}$$

où $V_{i,t}$ est le volume du réservoir i au temps t si le système hydro-électrique se trouve dans l'état S_t, $q_{t,i}$ est la quantité déversée du réservoir i au temps t si la décision d_t est prise et $R_i(V_{i,t}, q_{t,i})$ correspond au profit maximal qu'il est possible d'obtenir en distribuant la quantité $q_{t,i}$ entre les différentes turbines et conduites issues du réservoir i au temps t sachant que celui ci contient un volume $V_{i,t}$ d'eau.

Plus formellement, si $V_{i,t}$ est défini dans l'intervalle l, $R_i(V_{i,t}, q_{t,i})$ est la solution du problème d'optimisation combinatoire suivant :

$$\max_q \sum_{j \in \delta_i^+} prix_{j,t} \sum_{k \in Int_{q,j}} a_{j,l,k} \times q_{j,k}$$

sous contraintes :

$$q_{j,k} \leq (B_{j,k+1} - B_{j,k}) \times b_{j,k} \qquad \forall j,k \tag{5.10}$$

$$q_{j,k} \geq (B_{j,k} - B_{j,k-1}) \times b_{j,k} \qquad \forall j,k \tag{5.11}$$

$$r_{min,i} \leq r \leq r_{max,i} \tag{5.12}$$

$$q_{min,j,l} \times b_{j,0} \leq \sum_k q_{j,k} \leq q_{max,j,l} \times b_{j,0} \qquad \forall j \tag{5.13}$$

$$q_{t,i} = \sum_j \sum_k q_{j,k} + r \tag{5.14}$$

où $q_{min,j,l}$ correspond au débit minimal qui doit être utilisé par la turbine j si le volume du réservoir aval, i, est dans l'intervalle l pour respecter la contrainte de production minimale.

Pour résoudre ce problème, on utilise une méthode d'énumération implicite qui sur nos jeux de données est plus efficace qu'une résolution avec CPLEX (ce qui n'a rien de surprenant puisqu'il y a au plus deux turbines sortantes). Lorsque ce problème n'a pas de solution, c'est à dire lorsqu'il n'est pas possible de trouver une solution respectant la contrainte 5.12, $R_i(V_{i,t}, q_{t,i})$ est évalué par une constante négative de grande taille.

Nous considérerons ensuite comme état terminal, tout état S_t tel que $f(S_t) < 0$. Car dès lors pour atteindre l'état S_t, des décisions ne respectant pas les contraintes du problème, ont été prises et il ne sera donc pas possible de trouver la solution optimale du problème à partir de S_t.

2.2 Résolution

Comme dans le cas du problème d'affectation d'unités (voir section 2.2 du chapitre précédent), la méthode appliquée est la programmation dynamique par chaînage avant. Le principe de cette méthode est de construire la solution étape par étape, en cherchant à chaque étape t de l'horizon de planification la valeur des meilleurs chemins entre l'état initial S_0 et les états $S_t \in \mathcal{S}_t$ en se basant sur la relation de récurrence suivante :

$$\begin{cases} f(S_1) = R(S_0, S_1) & \forall S_1 \in \mathcal{S}_1 \\ f(S_t) = \min\limits_{S_{t-1} \in \delta^-(S_t)} R(S_{t-1}, S_t) + f(S_{t-1}) & \forall S_t \in \mathcal{S}_t \end{cases} \tag{5.15}$$

Pseudo-code

L'algorithme 2.2 résume l'application de la programmation dynamique au problème de planification de l'utilisation des eaux dans un réseau hydro-électrique.

Algorithme 5 Programmation Dynamique pour le problème de planification des eaux

Entrée : S_0, D, T, R
  $f(S_0) \leftarrow 0.$
  $\mathcal{S}_0 \leftarrow S_0$
  $\mathcal{S}_t \leftarrow \emptyset$
  pour $t = 1$ à T **faire**
    pour tout $S_{t-1} \in \mathcal{S}_{t-1}$ **faire**
      pour $i = 0$ à N **faire**
        $V_i \leftarrow$ volume d'eau dans le réservoir i à l'état S_{t-1}
        $qmin \leftarrow$ Valeur minimal de $q_{i,t-1}$ permettant de respecter les contraintes 5.4, 5.6 et 5.7.
        $qmax \leftarrow$ Valeur maximal de $q_{i,t-1}$ permettant de respecter les contraintes 5.5 et 5.6 .
        pour $d_i = qmin$ à $qmax$ avec le pas p **faire**
          calculer $R_i(V_i, d_i)$
        fin pour
      fin pour
      pour tout $d \in D(S_{t-1})$ **faire**
        $S_t \leftarrow T(S_{t-1}, d).$
        $f(S_t) = f(S_{t-1}) + \sum_{i=1}^{N} R_i(V_i, d_i)$
        S_t.prédécesseur $\leftarrow S_{t-1}.$
        si $\exists S \in \mathcal{S}_t$, tel que $S_t = S$ **alors**
          si $f(S_t) < f(S)$ **alors**
            Remplacer S par S_t.
          sinon
            si $f(S_t) \geq 0$ **alors**
              Ajouter S_t à $\mathcal{S}_t$
            fin si
          fin si
        fin si
      fin pour
    fin pour
  fin pour
  renvoie $\min_{S_T \in \mathcal{S}_T} f(S_T)$

3 DYNAMOP pour le problème de planification des eaux dans un réseau hydro-éléctrique

Dans cette section nous expliquons comment DYNAMOP peut être appliqué pour résoudre le problème de planification des eaux dans un réseau hydro-électrique.

3.1 Représentation et évaluation

La représentation est définie par les éléments du graphe d'états associé au problème. En effet, une solution est représentée par un chemin valide dans le graphe d'états. L'évaluation correspond alors simplement à la somme des valeurs des arcs du chemin et peut

s'effectuer avec la technique de Δ-évaluation ou d'évaluation itérée comme décrit au chapitre 3. Néanmoins, si on conserve les définitions telles que données dans la section 2.1, le sous-problème de déterminer si il existe un chemin entre deux états est un problème NP-complet. Il s'agit, en effet, du même problème de planification des eaux dans un réseau hydro-électrique, mais sur une période de temps plus courte et il a été montré dans la section 3.3 du chapitre 2 que le problème de déterminer s'il existe ou non un planning valide est un problème NP-complet. Or, la mise en œuvre de DYNAMOP demande de pouvoir déterminer facilement des sous-chemins liants deux états. C'est pourquoi nous avons opté pour relâcher les contraintes de débit minimal et maximal dans les conduites, c'est à dire les contraintes suivantes :

$$r_{min,i} \leq r_{i,t} \leq r_{max,i} \ \forall i, t$$

Au niveau de la définition des éléments de la programmation dynamique, ce choix implique les changements suivants :

— **Définition de l'espace des décisions :** Relâchement de la contrainte 5.7 qui entraînait le rejet des décisions ne permettant pas de respecter la contrainte $r_{min,i} \leq r_{i,t}$. De plus la contrainte 5.8, imposant que l'ensemble des décisions possibles soit discret et fini, est également relâchée, car le fait que $D(S)$ soit un ensemble non dénombrable n'est pas problématique pour DYNAMOP.
— **Définition des états terminaux :** Les états S_t qui ne sont pas joignables sans violer une contrainte du type $r_{i,t} \leq r_{max,i}$ ne sont plus considérés comme terminaux.

Avec ces changements, pour savoir s'il existe un chemin entre deux états $S_t = (q^{tot}_{t,i})_i$ et $S_k = (q^{tot}_{k,i})_i$, avec $k > t$, il suffit de vérifier que pour tout i :

$$q^{tot}_{k,i} \geq q^{tot}_{t,i}$$

la Figure 5.1 illustre la représentation dans le cas simple où le système est restreint à un unique réservoir.

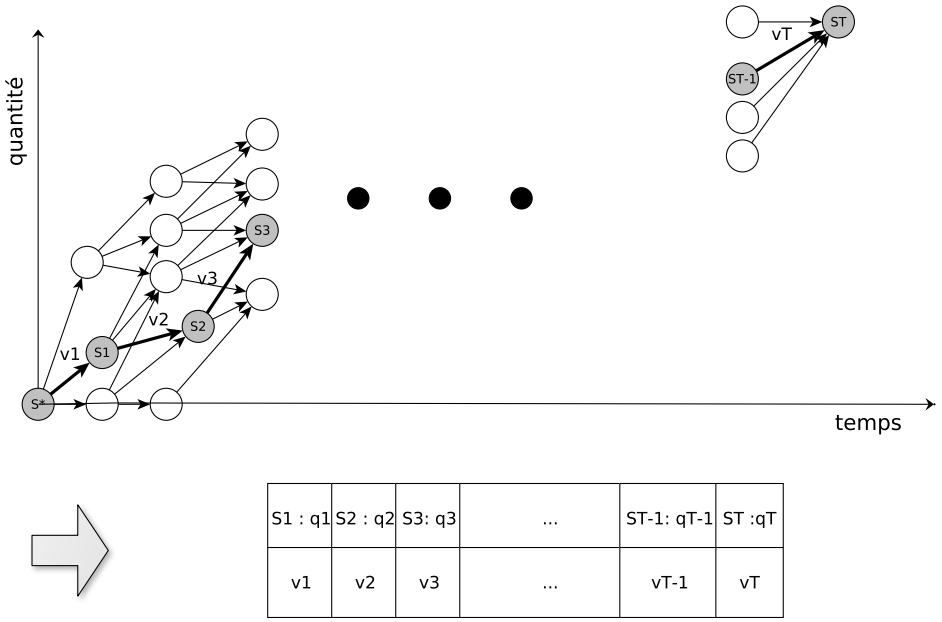

FIGURE 5.1 – Représentation.

3.2 Initialisation

Les chemins sont générés aléatoirement comme expliqué dans le chapitre 3. C'est à dire que partant de l'état S_0, on construit les états S_t, $t \geq 1$ en choisissant aléatoirement une décision d dans $D(S_{t-1})$ et en appliquant la relation de transformation $T(S_{t-1}, d)$. Cependant lorsqu'il existe des décisions $d \in D(S_{t-1})$ respectant les contrainte relâchées :

$$r_{min,i} \leq r_{i,t} \leq r_{max,i} \ \forall i, t$$

celles ci sont choisies en priorité.

3.3 Croisement

Le croisement effectué est un croisement 2-transitions tel que décrit au chapitre 3. L'idée est d'introduire une portion du premier chemin parent P_A dans le second chemin parent P_B en utilisant des chemins de transition. La portion de P_A que l'on cherche à introduire dans P_B est choisi de façon aléatoire et l'on notera $S^A_{t_1}$ et $S^A_{t_2}$ les états qui la délimitent.

On peut montrer qu'il existera toujours des chemins de transition permettant la mise en œuvre du croisement. En effet, du fait de l'unicité de l'état initial et de l'état final, il existera toujours un état de P_B à partir duquel $S^A_{t_1}$ peut être atteint et un état de P_B qui peut être atteint en partant de $S^A_{t_2}$.

Pour construire un chemin de transition liant P_B à $S^A_{t_1}$ minimal en terme du nombre de ces arcs, on cherche l'état $S^B_{t'_1}$ le plus proche, c'est-à-dire minimisant $(t_1 - t'_1)$, à partir duquel il est possible de joindre $S^A_{t_1}$, c'est-à-dire tel que :

$$S^A_{t_1,i} \geq S^B_{t_1,i} \quad \forall i$$

Puis on génère aléatoirement un chemin liant $S^B_{t'_1}$ à $S^A_{t_1}$. Un procédé similaire est utilisé pour construire le chemin de transition entre $S^A_{t_2}$ et P_B.

La figure 5.2 illustre le croisement dans le cas d'un système composé d'un unique réservoir. Le chemin noir et le chemin blanc sont croisés pour former deux nouvelles solutions. Les portions de chemins que l'on souhaite échanger sont celles comprises entre les locus t_1 et t_2. Les chemins de transition sont ceux passant par les états colorés en gris, ils sont représentés en utilisant des arcs plus épais.

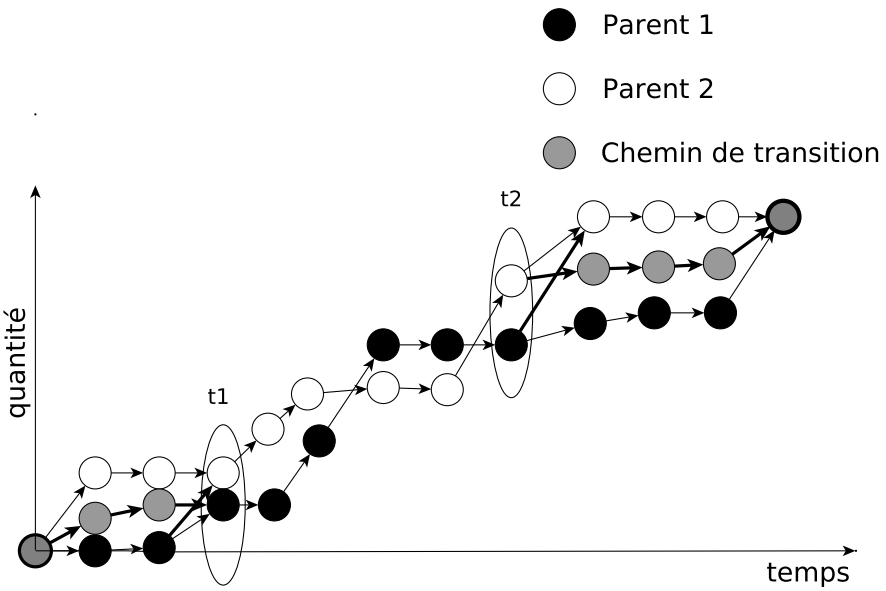

FIGURE 5.2 – Croisement.

3.4 Mutation

La mutation consiste à modifier aléatoirement un état du chemin. L'état à modifier est noté S_t, il est sélectionné aléatoirement. Il est remplacé par nouvel état S'_t qui est sélectionné aléatoirement parmi tous les états adjacents à S_{t-1} et S_{t+1}, c'est-à-dire les états vérifiant :

$$S_{t-1,i} \leq S'_{t,i} \leq S_{t+1,i} \quad \forall i$$

Si, parmi ces états, il en existe qui permettent de respecter les contraintes relâchées de bornes sur les débits, ceux-ci sont choisis préférentiellement.

3.5 Opérateurs intelligents

Mutation hybride

L'hybridation est mise en œuvre via une mutation intelligente. L'idée est de remplacer le chemin à muter par le meilleur chemin appartenant à un graphe restreint défini autour du chemin d'origine. Cette idée a déjà été détaillée dans le chapitre 3 au paragraphe 3.6.

Ici le graphe est restreint à la fois en longueur et en largeur. Plus précisément, la mutation consiste à sélectionner aléatoirement deux états S_1 et S_2 séparés par un nombre L de locus ainsi qu'un réservoir i. On cherche ensuite, à l'aide d'une méthode exacte, le meilleur chemin liant ces deux états tel que les débits de tous les réservoirs autres que i restent inchangés par rapport à ceux de la solution d'origine.

Pour résoudre le sous-problème de trouver le meilleur chemin dans le graphe restreint ainsi défini, nous avons utilisé la programmation dynamique telle qu'expliquée dans la section 2.

Croisement intelligent

Le croisement appelé recombinaison de sous-politiques qui a été introduit dans la section 3.6 du chapitre 3 est également appliqué. Il est employé entre deux individus notés A et B. Ici la seconde stratégie est utilisée pour déterminer les points de croisement, c'est à dire qu'on considère comme possible point de croisement tous les locus t tels que S^B_{t+1} soit accessible à partir de S^A_t et S^A_{t+1} soit accessible à partir de S^B_t, c'est à dire tels que :

$$\begin{cases} S^B_{t+1,i} \geq S^A_{t,i} & \forall i \\ S^A_{t+1,i} \geq S^B_{t,i} & \forall i \end{cases}$$

4 Protocole expérimental

Le protocole expérimental mis en place pour DYNAMOP sur le problème de la planification des débits d'eaux dans un réseau hydro-électrique est expliqué dans cette section.

4.1 Un algorithme génétique basique pour comparaison

Afin d'avoir un point de comparaison pour évaluer la qualité des solutions trouvées par DYNAMOP, nous avons implémenté un autre algorithme génétique utilisant une représentation et des opérateurs plus classiques. Il est dénoté BGA pour *Basic Genetic Algorithm* (Algorithme Génétique de base). Dans cette version une solution est représentée par un vecteur de $N \times T$ éléments définis dans l'intervalle [0,1]. Les T premiers éléments donnent la planification des débits pour le premier réservoirs les T éléments suivants donnent la planification du deuxième réservoir, etc.. Pour un réservoir i chaque élément du vecteur donne le pourcentage de la quantité d'eau disponible dans le réservoir qui est utilisée en plus de la quantité minimale requise. Cette quantité minimale est la quantité qui doit être déverser pour ne pas dépasser le volume maximale d'eau autorisé dans le réservoir. La quantité d'eau disponible est la quantité d'eau que l'on peut déverser en respectant la contrainte de volume minimal. Ainsi les solutions de la population respectent toujours les contraintes de bornes sur les volumes comme c'est le cas pour DYNAMOP. Cette représentation a aussi été décrite dans [126] pour le même genre de problème, sinon que les pourcentages y sont encodés en base deux afin d'utiliser des vecteurs binaires pour la représentation. Les opérateurs évolutionnaires choisis sont des opérateurs très classiques. Le croisement est un croisement 2-points et la mutation consiste à sélectionner aléatoirement un gène et de modifier sa valeur par une valeur choisie aléatoirement dans l'intervalle [0,1] en suivant une loi uniforme.

4.2 Cas d'étude

Les systèmes hydro-électriques sur lesquels DYNAMOP a été testé sont ceux présentés sur la figure 2.2. Pour le premier cas d'étude, le système est composé d'un unique réservoir, de deux turbines et d'un système d'évacuation des eaux. La capacité du réservoir est très grande et les sorties d'eau ne sont pas limitées sur les conduites. C'est pourquoi si l'on construit le graphe d'états de la programmation dynamique avec un pas de discrétisation des débits de $20m^3/s$ (ce qui n'est déjà pas très précis), celui-ci sera composé de 3×10^6 états. Ce nombre est important d'autant plus que l'évaluation des arcs est une opération coûteuse en temps de calcul. Il ne serait donc pas possible d'appliquer la programmation dynamique. Le second cas d'étude correspond au deuxième réseau hydro-électrique présenté sur la Figure 2.2. Il s'agit d'un réseau composé de 7 réservoirs de grande taille.

Comme la taille du graphe d'états augmente exponentiellement avec le nombre de réservoirs, on ne peut résoudre ce cas avec la programmation dynamique même en utilisant un pas de discrétisation grossier pour les débits (nous avons en effet un graphe de 9×10^{11} états avec un pas de discrétisation large correspondant à un découpage de l'espace des décisions par pas de $20m^3/s$).

4.3 Paramétrage

Encore une fois, nous commençons par paramétrer les différents algorithmes génétiques afin d'en tirer les meilleurs performances. Les paramètres à fixer sont le taux de croisement, le taux de mutations, la taille de la population et le choix de la stratégie de remplacement. Deux stratégies de remplacement sont en effet testées. La première consiste à remplacer toute l'ancienne génération par la nouvelle mais en appliquant un élitisme faible. C'est-à-dire que si le meilleur individu de la nouvelle génération n'est pas meilleur que le meilleur individu de l'ancienne génération alors on remplace le moins bon individu de la nouvelle génération par le meilleur de l'ancienne génération. La deuxième stratégie et de générer N_o individus et d'appliquer un élitisme fort. L'élitisme fort et de sélectionner les N meilleurs individus parmi l'ancienne génération et les N_o individus enfants générés. Dans ce cas N_o est aussi à paramétrer. Cette analyse a été effectuée avec Irace [113] qui est un package de R [180]. Son objectif principal est de configurer automatiquement les algorithmes étant donné un ensemble d'instances en trouvant les valeurs les plus appropriées pour les différents paramètres. Ici les instances correspondaient à 10 différents scénarios de prix et d'apport en eau. Pour chaque algorithme et pour chaque cas, le paramétrage a été fait pour un horizon de temps de 24 heures puis la taille de la population a été reparamétrée pour les autres horizons de planification.

5 Résultats expérimentaux

Chaque algorithme est exécuté 20 fois pour chaque type de réseau hydro-électrique et pour chaque horizon de planification, le critère d'arrêt utilisé est toujours un temps maximal d'exécution.

Des versions de DYNAMOP dans lesquels un ou plusieurs opérateurs intelligents ne sont pas utilisés sont également étudiées. On distingue ainsi trois versions modifiées de DYNAMOP :

- DYNAMOP/H : où la mutation hybride n'est pas appliquée ;
- DYNAMOP/R : où l'opérateur recombinaison de sous-politiques n'est pas utilisé ;
- DYNAMOP/RH : où ni la mutation hybride ni l'opérateur recombinaison de sous-politiques ne sont appliqués.

Ainsi, pour chaque cas d'étude, nous commencerons par faire une étude de l'impact des opérateurs intelligents proposés en comparant statisquement les différentes versions de DYNAMOP.

Nous comparerons ensuite DYNAMOP à BGA ainsi qu'à une méthode de résolution exacte. Sur le cas d'étude correspondant à un système hydro-électrique composé d'un unique réservoir, la méthode exacte utilisée est la programmation dynamique. Sur le second cas d'étude, où le système est composé de 7 réservoirs, nous utiliserons CPLEX pour résoudre le modèle de programmation linéaire mixte présenté au chapitre 2, car dans ce cas, cette approche est plus rapide.

Les tests statistiques utilisés sont toujours le test de Kruskal Wallis suivit d'un test post-hoc, car nos tests sont non appariés. Nous avons utilisé une p-value de 0,01 pour

rejeter l'hypothèse nulle.

Tous les tests seront présentés sous forme de tableau. Dans chacun de ces tableaux, la ligne "Moyenne" donne la moyenne des résultats obtenus par l'algorithme suite aux 20 exécutions, la ligne "std" l'écart type des résultats, et la ligne "Meilleur", le meilleur résultat trouvé sur les 20 exécutions. Quand à la ligne "Écart", elle donne un pourcentage qui se calcule avec la formule :

$$\frac{meilleure_{trouve} - meilleure_{algo}}{meilleure_{trouve}} \times 100\,;$$

où $meilleure_{trouve}$ est la meilleure solution réalisable trouvée pour le problème et $meilleure_{algo}$ est la meilleur solution trouvée par l'algorithme testé. La valeur de $meilleure_{trouve}$ sera notée en gras dans les tableaux. Enfin, la ligne "Temps" correspond au temps d'exécution de l'algorithme, ce temps est donné en secondes.

Dans ces tableaux, la colonne "DP/MIP" donne les résultats obtenus par l'une des deux méthodes exactes. Ces résultats ne correspondent pas forcément aux solutions optimales car la programmation dynamique utilise une discrétisation et parce qu'il peut être impossible de trouver la solution sans dépasser les capacités mémoire et dans ce cas la meilleure solution réalisable trouvée avant que le programme ne s'arrête est celle fournie.

5.1 Premier cas d'étude

Dans cette section nous présentons les résultats concernant le premier cas d'étude, c'est-à-dire le cas où le système hydro-électrique est composé d'un unique réservoir et de deux turbines.

Impact des opérateurs intelligents

Le Tableau 5.1 montre les résultats obtenus par les différentes versions de DYNAMOP. Que ce soit pour un horizon de planification d'un mois (720 h) ou bien d'un an (8760 h), l'analyse statistique a permis de montrer que DYNAMOP et DYNAMOP/R sont significativement meilleurs que les deux versions de DYNAMOP pour lesquels l'hybridation n'a pas été appliquée. On peut donc en conclure que l'hybridation a un impact positif statistiquement significatif sur la méthode. En revanche il n'y a pas de différence statistiquement significative entre DYNAMOP et DYNAMOP/R ni entre DYNAMOP/H et DYNAMOP/RH, l'impact de l'opérateur de recombinaison de sous-politiques est donc négligeable pour ce cas.

		DYNAMOP	DYNAMOP/R	DYNAMOP/H	DYNAMOP/RH
720h	Moyenne ($\times 10^6$)	2.998866	2.939158	2.125493	2.354856
	std ($\times 10^4$)	5.107712	6.512451	9.680990	1.866464
	Meilleur ($\times 10^6$)	3.061130	**3.047260**	2.312820	2.600680
	Ecart	**1.25**%	1.7%	25.39%	16.11 %
	Temps (s)	2420			
8760h	Moyenne ($\times 10^7$)	3.070198	3.061994	2.070648	2.142777
	std ($\times 10^5$)	4.567597	4.077857	3.850120	3.079199
	Meilleur ($\times 10^7$)	3.118440	**3.115630**	2.121640	2.179300
	Ecart %	3.18%	**3.27** %	34.12%	62.34 %
	Temps (s)	20000			

Tableau 5.1 – Influence des opérateurs : premier cas d'étude.

T		BGA	DYNAMOP/H	DYNAMOP	DP
720h	Moyenne ($\times 10^6$)	2,3524	2,3548	**2,9391**	**3,0999**
	std ($\times 10^6$)	0,1540	0,1866	**0,0651**	
	Meilleur ($\times 10^6$)	*2,5949*	2,6007	**3,0473**	
	Ecart	16,29 %	16,11 %	**1,7**%	0 %
	Temps (s)	2420			5030
8760h	Moyenne ($\times 10^7$)	1,4549	2,1428	**3,0620**	**3,2209**
	std ($\times 10^7$)	0,0123	0,031	**0,0410**	
	Meilleur ($\times 10^7$)	*1,4740*	2,1793	**3,1156**	
	Ecart	54,24 %	32,34 %	**3,27** %	0%
	Temps (s)	20000			79376,82

Tableau 5.2 – Résultat sur le premier cas d'étude : Système hydro-électrique à un réservoir. Les solutions obtenues par la programmation dynamique sont utilisées pour calculer les écarts.

Comparaison à BGA

Dans le Tableau 5.2, les résultats obtenus par DYNAMOP et DYNAMO/H sont comparés à BGA. On peut noter que leurs performances sont très différentes. Pour un horizon de planification de 720 heures, DYNAMOP/H et BGA donnent des résultats assez similaires et leur écart avec la solution fournie par la programmation dynamique est de plus de 16%. En revanche les résultats obtenus par DYNAMOP ne sont pas très loins de la solution de la programmation dynamique (juste 1,7% de différence en un temps de calcul deux fois moins long). Pour ce premier cas, l'hybridation permet de considérablement améliorer les résultats. Un test de Kruskal Wallis appliqué aux échantillons obtenus grâce aux différentes exécutions suivi d'un test post hoc permettent de confirmer la supériorité de DYNAMOP sur les deux autres algorithmes avec un risque de 0,01%.
Pour un horizon de planification de 8760 heures, on peut observer un écart réel entre tous les algorithmes. DYNAMOP/H permet d'obtenir un écart de 32,34% avec la solution optimale tandis que BGA a un écart à la solution optimale de 54,24%. Avec une hybridation avec la programmation dynamique, la performance de l'algorithme est réellement améliorée et DYNAMOP est proche de la solution de la programmation dynamique. L'écart est en effet seulement de 3,27% avec un temps de calcul 4 fois moindre. La signification des résultats est encore confirmée par les tests statistiques. On peut donc dire qu'on a de très bon résultats sur ce cas simple. Dans la prochaine partie nous allons étudier les résultats de DYNAMOP sur le second cas d'étude qui correspond à un système plus complexe.

5.2 Deuxième cas d'étude

Nous présentons ici les résultats liés au second cas d'étude qui correspond au système hydro-électrique composé de sept réservoirs.

Impact des opérateurs intelligents

Le tableau 5.3 permet de visualiser l'impact de l'opérateur de recombinaison de sous-politiques et de la mutation hybride sur DYNAMOP.

Pour les simulations sur une période de 24h, l'analyse statistique montre qu'il y a une différence significative entre les différentes versions de DYNAMOP. Grâce au test post-hoc

nous pouvons montrer que DYNAMOP est statistiquement meilleur que DYNAMOP/RH et que DYNAMOP/R mais qu'il n'y a pas de différence significative entre DYNAMOP et DYNAMOP/H. La différence entre DYNAMOP/RH et DYNAMOP-H est également significative. Nous pouvons donc en déduire que l'opérateur de recombinaison de sous-politique permet d'améliorer significativement les résultats. En revanche l'opérateur de mutation hybride n'a pas d'impact significatif sur la qualité de la méthode.

Sur un horizon de 720 heures, il n'y a pas différences significative entre les différentes versions si ce n'est entre DYNAMOP/R et DYNAMOP/H. En fait DYNAMOP/R est significativement meilleur que DYNAMOP/H. Cela peut être du au fait que l'opérateur de recombinaison de sous-politique accélère la convergence des algorithmes vers un optimum local mais que ce problème peut être compensé par l'hybridation.

Sur un horizon d'un an (8760 h), aucune différence statistiquement significative n'existe entre les différentes versions de DYNAMOP.

Nous avons pu voir que sur ce second cas d'étude, l'impact de l'hybridation n'est pas significatif. En fait cela n'est pas surprenant étant donné que l'hybridation n'affecte à chaque fois le planning que d'un réservoir sur les sept qui composent le système. Nous pouvons aussi remarquer que l'impact de l'opérateur de recombinaison de sous-politiques n'est pas toujours positif, cela est probablement dû au fait que cet opérateur peut réduire la diversité de la population. En effet, pour cet opérateur de croisement, un seul individu enfant est créé et celui ci est systématiquement créé en sélectionnant entre chaque couple d'états communs aux deux parents le meilleur des deux sous-chemins de ceux composant les parents. Il serait intéressant, dans un travail futur, d'étudier d'autres opérateurs intelligents pour en proposer qui soient plus efficaces sur des réseaux de grande taille.

		DYNAMOP	DYNAMOP/R	DYNAMOP/RH	DYNAMOP/H
24h	Moyenne ($\times 10^4$)	4,349057	4,304912	4,284973	4,335267
	std	307,9185	272,0350	257,9210	379,3238
	Meilleur ($\times 10^4$)	4,378780	4,352450	4,313250	4,377530
	Ecart	2,23 %	2,8%	3,69%	2,25%
	Temps (s)	1650			
720h	Moyenne ($\times 10^6$)	2,483758	2,584773	2,578471	2,441669
	std ($\times 10^5$)	3,200839	7,671210	1,066230	3,736104
	Meilleur ($\times 10^6$)	**2,742310**	2,595560	2,591170	2,732320
	Ecart	-	5,35%	5,51%	0,36%
	Temps (s)	20000			
8760h	Moyenne ($\times 10^7$)	2,572767	2,245100	2,004100	2,497051
	std ($\times 10^6$)	1,017547	0,413590	0,628700	2,741583
	Meilleur ($\times 10^7$)	2,766270	**2,781310**	2,066940	2,631820
	Ecart	0,54%	-	25 %	5,37 %
	Temps (s)	40000			

Tableau 5.3 – Influence des opérateurs intelligents : Second cas d'étude. L'écart calculé est celui avec la solution optimale obtenue avec CPLEX . Cependant lorsqu'il est trop coûteux d'obtenir une solution exacte, l'écart calculé est celui avec la meilleure solution trouvée qui est celle écrite en gras.

Comparaison avec BGA

Dans le Tableau 5.4, les résultats des simulations pour DYNAMOP et BGA sont présentés pour chaque horizon de planification (24 heures, 720 heures et 8760 heures). Les solutions données dans la dernière colonne du tableau sont celles obtenues en résolvant un programme linéaire à variable mixtes avec le logiciel CPLEX. Pour l'horizon de planification le plus court, 24 heures, cette solution est la solution optimale. En revanche pour un horizon de 720 heures il ne s'agit pas de la solution optimale. En effet, le programme est stoppé avant d'avoir trouvé la solution optimale à cause d'un manquement de mémoire. C'est donc la meilleure solution réalisable trouvée par le programme avant que le programme ne s'arrête qui est notée. Il s'agit d'une borne inférieure pour la solution optimale. Pour un horizon d'un an (8760 heures) la taille des données est trop grande pour obtenir une solution réalisable que ce soit avec CPLEX ou avec la programmation dynamique (ce cas a été noté NA dans le tableau).

Comme précédemment on laisse à tous les algorithme le même temps de calcul pour résoudre le problème. On peut tout d'abord remarquer que dans certain cas, l'algorithme génétique classique (BGA) ne permet pas d'obtenir des solutions réalisables, on obtient donc des résultats négatifs. En revanche la représentation utilisée dans DYNAMOP aide à trouver de telles solutions et donc les solutions fournies par DYNAMOP ainsi que par toutes ces variantes vérifient toujours toutes les contraintes.

Pour un horizon de planification de 24 heures, il y a une différence statistiquement significative entre les résultats obtenus par les deux algorithmes. Une analyse statistique montre DYNAMOP et toutes ces variations (DYNAMOP/H, DYNAMOP/RH et DYNAMOP/R) sont significativement meilleures que BGA. Ceci est en accord avec les résultats numériques qui montrent que DYNAMOP est meilleur que BGA avec ou sans hybridation. En fait, l'écart entre la solution de DYNAMOP et la solution optimale est de moins de 3% alors qu'aucune solution réalisable n'a été trouvée avec BGA.

De même pour un horizon de planification de 720 heures, il y a une différence statistiquement significative entre BGA et les différentes versions de DYNAMOP. En 17 fois moins de temps, DYNAMOP permet d'obtenir un résultat 47 fois meilleur que celui obtenu par CPLEX. La différence entre les solutions trouvées par DYNAMOP et les solutions trouvées par BGA est de 13%, on peut donc dire que DYNAMOP permet d'obtenir de bien meilleurs résultats qu'un algorithme classique sur ce cas.

Pour le plus long horizon de planification, c'est à dire de 8960 heures qui correspond à un horizon de planification d'un an, il n'a pas été possible de trouver une solution réalisable avec CPLEX sans dépasser les capacités mémoires. Ceci est noté NA dans le Tableau 5.4. Encore une fois DYNAMOP fournit les meilleurs résultats et la différence entre DYNAMOP et BGA est statistiquement significative. La différence entre la meilleur solution trouvée avec DYNAMOP et la meilleur solution trouvée par BGA est de 39,4%.

Tableau 5.4 – Comparaison avec BGA pour le second cas d'étude

		BGA	DYNAMOP	MILP
24h	Moyenne($\times 10^4$)	-5,634164	4,349057	**4,478600**
	std	417,2407	307,9180	
	Meilleurt($\times 10^4$)	-5,564230	4,378780	
	Ecart	NF	2,23%	
	Temps (s)	1650		3030,129
720h	Moyenne ($\times 10^6$)	2,150734	2,483758	0,055261
	std ($\times 10^5$)	0,65054	3,20084	
	Meilleur ($\times 10^6$)	2,247370	**2,742310**	
	Ecart	18%	-	97,98 %
	Tempse (s)	20000		345656,23
8760h	Moyenne ($\times 10^7$)	1,668265	2,572767	NA
	std ($\times 10^5$)	2,925216	10,17547	
	Meilleur ($\times 10^7$)	1,700970	**2,766270**	
	Ecart	38,51 %	-	
	Temps (s)	40000		-

6 Conclusion

L'efficacité de DYNAMOP à résoudre un cas d'application réel complexe a été démontrée dans ce chapitre.

En effet nous avons montré comment DYNAMOP peut s'appliquer au problème de planification de l'utilisation des eaux dans un réseau hydro-électrique. Pour cela nous avons d'abord expliqué comment les solutions de ce problème peuvent être modélisées sous forme de séquence d'états. Nous avons ensuite vu comment construire une mutation modifiant un nombre minimal d'états de manière à suivre l'idée de DYNAMOP. Nous avons également vu comment construire des chemins de transition et donc mettre en place un croisement 2-transitions tel que présenté au chapitre 3. Finalement nous avons mis en place deux des idées d'opérateurs intelligents du chapitre 3 : D'une part l'opérateur de recombinaison de sous politiques, qui partant de deux solutions cherche, en recombinant les sous-chemins, à créer une nouvelle solution de meilleure qualité et, d'autre part, une mutation hybride qui utilise une méthode exacte pour trouver le meilleur chemin dans un graphe d'états restreint défini autour de la solution courante.

Nous avons ensuite appliqué DYNAMOP sur deux instances réelles du problème. Pour chacune de ces instances nous avons tout d'abord étudié l'impact des opérateurs intelligents sur DYNAMOP grâce à des tests statistiques. Ces tests ont montré que les opérateurs intelligents proposés sont surtout intéressants sur les instances les plus petites étudiées. C'est à dire sur le cas d'étude du réseau composé d'un unique réservoir et sur le réseau arborescent pour un horizon de planification de 24 heures. Pour les instances plus grandes il serait intéressant d'étudier d'autres opérateurs intelligents et/ou hybrides. Ceci est à inclure dans nos perspectives de travail futur.

Nous avons également comparé les performances de DYNAMOP à celle d'un algorithme génétique classique mais utilisant une représentation qui a déjà montré son efficacité dans le cadre d'un problème similaire. Une étude statistique a permis de montrer que DYNAMOP donne des résultats significativement meilleurs. En fait les capacités de DYNAMOP

dépassent même grandement celle de l'algorithme génétique classique pour ce problème. Ceci prouve l'intérêt de DYNAMOP pour ce genre de problèmes.

Concernant les perspectives sur ce travail, deux directions sont à explorer. D'une part étudier d'autres opérateurs évolutionnaires, notamment d'autres opérateurs intelligents qui auraient un impact plus fort sur les instances de grandes tailles. Pour ce faire nous pensons qu'une stratégie intéressante serait de mettre en place une hybridation qui affecterait les débits de plusieurs réservoirs mais pour laquelle on limiterait les valeurs possibles des débits en fixant une déviation maximale par rapport à leurs valeurs dans la solution d'origine.

D'autre part, une perspective d'amélioration serait d'affiner la modélisation du problème pour la rendre plus réaliste et/ou plus générale en adaptant la méthode en conséquence. En ce sens, on pourrait par exemple ajouter des capacités de variations maximales des débits dans les liaisons d'une période de temps à une autre, des temps de transfert d'un réservoir à l'autre et la possibilité d'inclure des pompes dans le réseau.

Deuxième partie

MO-DYNAMOP : Adaptation de DYNAMOP à l'optimisation multi-objectif

Chapitre 6

Optimisation multi-objectif, concepts de bases, approches de résolution et cadre applicatif

Le premier chapitre de cette seconde partie, introduit les concepts de base liés à l'optimisation multi-objectif, à savoir :

- *définition formelle d'un problème d'optimisation multi-objectif ;*
- *notions de dominance Pareto et de Pareto optimalité ;*
- *définition de ce que nous entendons par solution d'un problème d'optimisation multi-objectif ;*
- *notions de critère de performance et méthodes d'évaluation de la qualité d'une solution d'un problème d'optimisation combinatoire multi-objectif ;*
- *état de l'art des méthodes de résolution existantes.*

Nous décrivons également, dans ce chapitre, le problème d'affection d'unités multi-objectif, qui servira de cadre applicatif pour cette seconde partie.

1 Introduction

Ce chapitre a pour objectif de présenter l'ensemble des concepts nécessaires à la compréhension de ce qu'est un problème d'optimisation multi-objectif.

Dans une première partie, nous introduirons un certain nombre de définitions. Tout d'abord nous formaliserons le concept de problème d'optimisation multi-objectif ; puis nous expliquerons ce que signifie, dans cette thèse, résoudre un problème d'optimisation multi-objectif. Pour cela, nous introduirons les concept de Pareto dominance, de Pareto optimalité et de front Pareto.

Puis, dans une seconde partie, nous aborderons le concept d'approximation en optimisation multi-objectif. Nous définirons tout d'abord des critères de qualités permettant de définir ce que l'on entend par *"bonne approximation"* de la solution d'un problème multi-objectif. Puis nous montrerons comment, à partir de ces critères, il est possible d'évaluer une approximation en multi-objectif.

Nous présenterons ensuite les différentes méthodologies de résolution de problèmes multi-objectifs existantes. Pour cela, nous commencerons par situer les approches nous intéressant dans le cadre des problèmes de décision multi-critère en considérant le rôle donné au décideur dans le processus de décision.

Enfin, nous présenterons le problème d'affectation d'unités multi-objectif dont la résolution sera étudiée dans cette partie.

2 Définitions

Cette section pose les différentes définitions et notations liées à l'optimisation combinatoire multi-objectif qui nous seront utiles par la suite. Elle a pour but d'introduire un certain nombre de concepts de base tels que la définition d'un problème d'optimisation multi-objectif et les notions de dominance Pareto et d'optimalité. De nombreuses définitions sont tirées du livre de Ehrgott [45].

2.1 Formulation générale d'un problème d'optimisation multi-objectif

Définition : Un problème d'optimisation multi-objectif peut se définir formellement comme ceci :

$$\min_x \{f_1(x), f_2(x), ..., f_m(x)\} \tag{6.1}$$

tel que :

$$x \in X \tag{6.2}$$

$$m \geq 2 \tag{6.3}$$

où m est le nombre de fonctions objectif f_k que l'on cherche à minimiser (ou maximiser). x est le vecteur des variables de décisions et X est l'espace des solutions réalisables. Il faut bien remarquer que dans le cas de l'optimisation multi-objectif **l'espace objectif** :

$$Z = f(X) = \{(f_1(x), f_2(x), ..., f_m(x)), x \in X\} \ ,$$

est un espace à m dimension. Sans perte de généralité nous supposerons par la suite que $Z \subset \mathcal{R}^m$ et que toutes les fonctions objectifs sont à minimiser. Généralement les objectifs à minimiser sont conflictuels, c'est à dire que si x_k^* correspond à la solution optimale de la fonction f_k elle n'est pas optimale pour les autres objectifs. Il n'est donc pas possible de chercher une solution optimisant tous les objectifs simultanément, c'est pourquoi on cherche un ensemble de solutions "efficaces".

Dans notre étude la notion d'efficacité choisie est celle de la Pareto optimalité. Elle sera définie dans la section 2.2. La définition suivante introduit une notation que nous utiliserons couramment par la suite :

Définition : L'image d'une solution x dans l'espace objectif est le point :

$$z^x = (f_1(x), f_2(x), ..., f_m(x))^T.$$

2.2 Notion de dominance Pareto et d'optimalité

Comme nous l'avons dit précédemment, l'optimisation multi-objectif nécessite de définir la notion d'ensemble de solutions "efficaces". En effet, étant donné que les objectifs sont généralement conflictuels il n'existe pas forcément de solution qui serait simultanément optimale pour chaque objectif. De ce fait, il n'est pas évident de définir un ordre total entre les solutions comme c'est le cas en optimisation mono-objectif. Ainsi, une relation d'ordre partielle est généralement définie, il s'agit de la relation de Pareto dominance.

Définition : On dit que le vecteur z^x **domine au sens de Pareto** le vecteur z^y ($z^x \succ z^y$) si et seulement si $f_k(x) \leq f_k(y)\ \forall k$ et $x \neq y$.

Définition : On dit que le vecteur z^x **domine strictement ou fortement au sens de Pareto** le vecteur z^y ($z^x \succ z^y$) si et seulement si $f_k(x) < f_k(y)\ \forall k$ et $x \neq y$. Si il existe k tel que $f_k(x) = f_k(y)$, alors on parle de **dominance faible**.

Définition : Une solution x domine (faiblement/fortement) au sens de Pareto une solution y ($x \succ y$) si et seulement si z^x domine (faiblement/fortement) z^y.

Définition : Un ensemble A domine un ensemble B si et seulement si pour tout $b \in B$, il existe $a \in A$ tel que a domine b. Cette définition peut être étendue aux concepts de dominance forte et faible.

Définition : Une solution x est **efficace** ou **Pareto optimale** si il n'existe pas de solution réalisable qui domine X. L'image d'une solution efficace est un **point non dominé**.

Définition : Deux solutions x et y sont dites **Pareto équivalentes** si l'une ne domine pas l'autre (au sens de Pareto).

Définition : L'image de l'ensemble des solutions Pareto optimales ou efficaces est le **front Pareto optimal**, noté Z^P. Ces solutions sont Pareto équivalentes.

Ainsi la résolution d'un problème d'optimisation multi-objectif basée sur le concept de Pareto dominance revient à fournir l'ensemble des solutions efficaces, c'est à dire Pareto optimales et le front Pareto associé. En outre par la suite lorsque l'on parlera de dominance ce sera toujours au sens de Pareto.

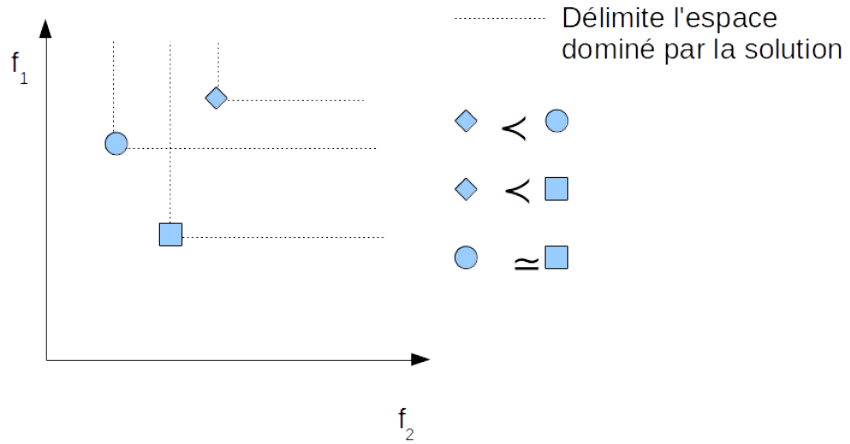

FIGURE 6.1 – Illustration du principe de Pareto dominance dans l'espace objectif.

La figure 6.1 aide à visualiser les notions de dominance et d'équivalence dans le cadre d'un problème d'optimisation multi-objectif où tous les objectifs sont à minimiser. Les points rond et carré sont équivalents et ils dominent tous les deux le point en forme de losange.

2.3 Classification des ensembles Pareto optimaux

Dans le cadre de notre étude, le but de l'optimisation multi-objectif est donc de fournir l'ensemble Pareto optimal au décideur afin que celui-ci sélectionne la ou les solutions qui correspondent à ses préférences. Néanmoins la notion d'ensemble optimal peut être affinée pour prendre en compte la notion de solutions équivalentes.

Définition : Deux solutions $x, y \in X$ sont équivalentes si et seulement si $f(x) = f(y)$.

Définition : Si Z^P est le front Pareto optimal d'un problème d'optimisation multi-objectif et X^P l'ensemble des solutions Pareto optimales, un **ensemble complet** est un ensemble $X^C \subset X^P$, tel que pour chaque vecteur z de Z^P il existe au moins une solution $x \in X^C$ pour laquelle $f(x) = z$.

Définition : Un **ensemble complet minimal** est un ensemble complet sans solutions équivalentes.

Définition : Un **ensemble complet maximal** est un ensemble complet contenant toutes les solutions équivalentes.

De même qu'en mono-objectif on ne cherche à fournir qu'une seule solution optimale, le but de l'optimisation multi-objectif est de fournir un ensemble complet minimal au décideur. Dans certaines applications le but est uniquement de fournir une image de l'ensemble Pareto optimal dans l'espace objectif, mais là encore, il s'avère inutile de conserver les solutions équivalentes.

2.4 Bornes du front Pareto

En supposant que l'optimum de chaque fonction objectif soit connu, on peut définir les notions de point idéal, de point utopique et de point Nadir.

Définition : Le point Zeleny ou point idéal , $z^I = (z_1^I, z_2^I, ...z_m^I)$, est le point obtenu en minimisant chaque objectif séparément. Il représente une solution idéale mais non réalisable. $z^I = (f_1(x_1^*), f_2(x_2^*), ..., f_m(x_m^*))^T$, où x_k^* correspond à la solution minimisant la fonction f_k.

Définition : Le point utopique, $z^U = (z_1^U, z_2^U, ...z_m^U)$, est le vecteur utopique irréalisable dont les composantes sont formées par $z_i^U = z_i^I - \epsilon_i$ pour $i \in \{1, 2, ..., m\}$, où z^I est le point idéal et ϵ est un vecteur de valeurs scalaires strictement positives relativement petites mais significatives.

Définition : Le point Nadir, $z^N = (z_1^N, z_2^N, ..., z_m^N)^T$, est construit à partir des pires valeurs qui peuvent être prises par chaque objectif dans l'ensemble des points du front de Pareto. C'est à dire que $z_k^N = \max_{\{X, z^X \in P^*\}} (f_k(X))$.

Ce point Nadir peut être très difficile à calculer, notamment lorsque le nombre de fonctions objectifs est strictement supérieur à deux.

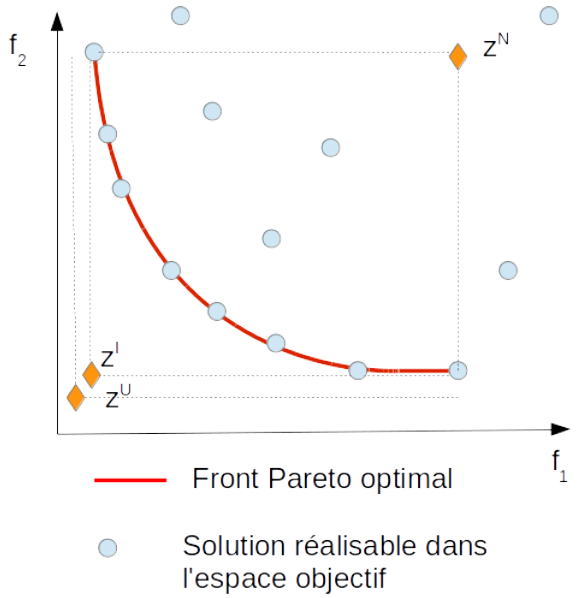

FIGURE 6.2 – Représentation du point Nadir z^N, du point utopique z^U et du point idéal z^I dans l'espace objectif.

La Figure 6.2 illustre les concepts de point idéal, de point utopique et de point Nadir.

3 Solution approchée pour l'optimisation multi-objectif : critères de qualité et méthode de comparaison

Dans cette section nous allons aborder la notion de solution approchée pour les problèmes multi-objectifs et voir quelles méthodes peuvent être appliquées pour mesurer la qualité d'un ensemble de solutions sous-optimales ou pour comparer deux approximations du front de Pareto. En effet, si la comparaison entre deux solutions approchées d'un problème mono-objectif est très instinctive, il en est tout autrement pour les problèmes multi-objectif où une solution approchée du problème correspond à un ensemble de solutions Pareto équivalentes. Cependant, la plupart des méthodes de résolution de problème multi-objectif ne permettent pas d'assurer que la solution qu'elles fournissent est un ensemble Pareto optimal complet. Il est donc nécessaire d'avoir des indicateurs de qualité permettant de mesurer la qualité d'une approximation d'un front de Pareto. Ces mesures de qualité ont pour but de déterminer si un algorithme fournit une meilleure solution qu'un autre et, si oui, quelle est l'importance de l'amélioration.

3.1 Critères de qualité

Deux critères doivent être considérés pour évaluer la qualité d'un ensemble de solutions Pareto équivalentes : un critère de convergence ou de qualité et un critère de diversification. L'idéal étant de fournir un front composé de solutions Pareto optimales bien réparties. La figure 6.3 illustre bien cette situation.

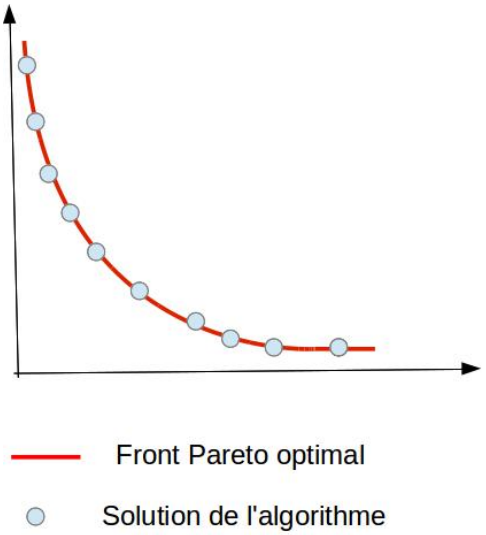

FIGURE 6.3 – Solution idéale : ensemble de solutions Pareto optimales bien réparties.

Le critère de convergence correspond à la distance de l'ensemble obtenu au front Pareto optimal. La Figure 6.4 illustre un cas de mauvaise convergence.

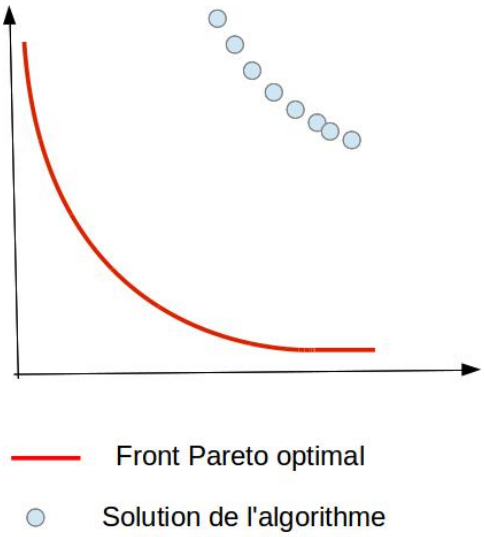

FIGURE 6.4 – Mauvaise convergence : Le front Pareto obtenu est loin du front optimal.

Le critère de diversification regroupe deux concepts. D'une part, la dispersion des solutions qui sera considérée bonne si la distance entre les différentes solutions est régulière et qu'elles ne sont pas trop proches les unes des autres. D'autre part, l'expansion qui correspond au fait que l'ensemble des solutions doit s'étendre vers les solutions extrémales (c'est-à-dire optimales pour l'un des objectifs). La figure 6.5 montre un cas où l'ensemble solution a une mauvaise distribution mais une bonne expansion. A l'inverse, la figure 6.6 montre un cas où l'ensemble obtenu a une mauvaise expansion mais une bonne distribution.

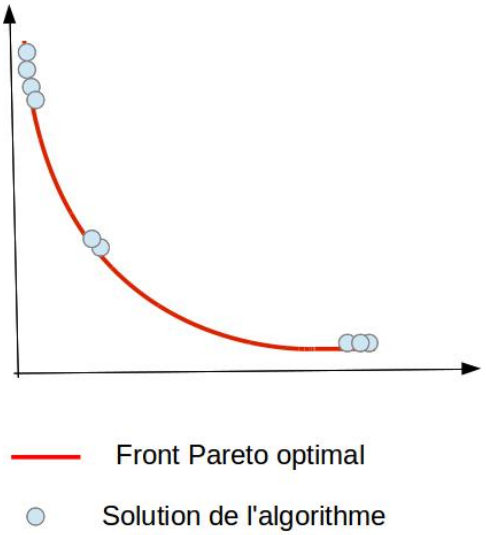

FIGURE 6.5 – Mauvaise distribution mais bonne expansion.

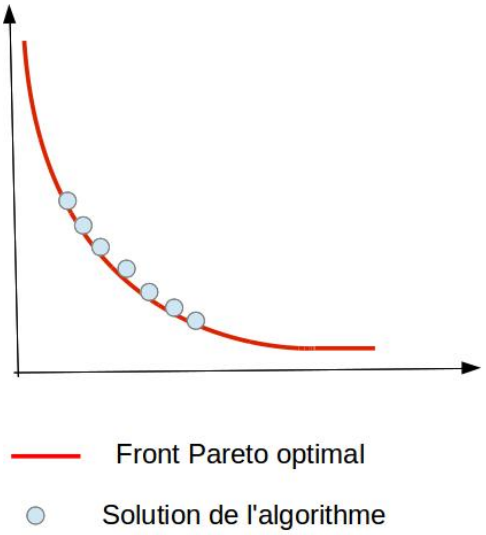

FIGURE 6.6 – Mauvaise expansion mais bonne distribution.

3.2 Indicateur de qualité

Les indicateurs de qualité peuvent être binaires ou unaires. Un indicateur binaire est un indicateur qui permet de comparer deux solutions ou deux ensembles de solutions. Un indicateur unaire mesure la qualité d'une solution ou d'un ensemble de solutions. Ces indicateurs peuvent aussi être classés selon qu'ils permettent de mesurer la convergence (e.g. la mesure de contribution, la distance générationelle, l'ϵ-indicateur ou la mesure de cardinalité), la diversification (e.g. la mesure de dispersion (en anglais *spread*), la mesure d'extension ou l'entropie), ou bien les deux simultanément (e.g. l'hypervolume, les fonctions R ou les fonctions de Tchebycheff). Le choix d'un indicateur doit donc être fait en déterminant préalablement quel est le critère de qualité que l'on souhaite étudier. Nous

nous contenterons de présenter ici, un indicateur de chaque type, en choisissant préférentiellement les indicateurs que nous utiliserons dans la suite de ce manuscrit, une description détaillée des autres indicateurs peut être trouvée dans [175].

— **Hypervolume :** L'hypervolume a été introduit en 1999 par Ziztler et Thiele [211]. C'est un indicateur hybride qui combine la mesure de convergence et de diversité. Il peut être utilisé comme un opérateur unaire ou binaire. Dans le cas unaire, l'indicateur d'hypervolume est utilisé pour mesurer la qualité d'un ensemble A correspondant à une solution approchée et est noté $I_H(A)$. Il s'agit du volume de l'espace objectif qui est dominé par l'ensemble A. Dans le cas binaire, l'hypervolume est utilisé pour comparer deux ensembles A et B et est noté $I_H(A, B)$. $I_H(A, B)$ est le volume de l'espace dominé par A moins le volume de l'espace dominé par B. On appelle également la version binaire de l'hypervolume, l'hypervolume différence. Le calcul de l'hypervolume nécessite l'utilisation d'un point de référence dans l'espace objectif. Par exemple le point Nadir peut être utilisé. Le point de référence peut également être défini comme le point composé des pires valeurs de chaque objectif parmi les solutions à comparer. La figure 6.7 permet de visualiser à quoi correspond l'hypervolume. Dans cette figure, l'hypervolume (en tant que mesure unaire) de l'ensemble des points triangulaires est l'aire de l'espace coloré en bleu foncé. Celui de l'ensemble des points en forme de losanges est l'aire de l'espace coloré en bleu clair plus l'aire de l'espace coloré en bleu foncé. L'hypervolume différence entre l'ensemble des losanges et des triangles est l'aire de la portion de l'espace coloré en bleu clair. Quant à l'hypervolume différence entre l'ensemble des triangles et des losanges, c'est l'aire de la portion de l'espace coloré en bleu clair multiplié par -1. C'est un indicateur très utilisé dans la littérature car il permet de mesurer les deux critères simultanément et il vérifie une propriété importante, il est strictement conforme au sens de Pareto. Ceci signifie que si un front A domine un front B, alors l'hypervolume de A sera strictement supérieur à l'hypervolume de B. En revanche, il est à noter que cet indicateur est sensible à l'échelle des fonctions objectif et au choix du point de référence. Par conséquent, les magnitudes de toutes les fonctions objectifs se doivent d'être normalisées, ceci afin de ne pas privilégier un objectif au détriment d'un autre.

— **ϵ-indicateur :** l'ϵ-indicateur à été introduit par Zitzler *et al.* en 2003 [212] et se base sur la notion d'efficacité epsilon introduite par Helbig et Pateva en 1994 [71]. L'ϵ-indicateur est un indicateur de convergence. C'est un opérateur binaire qui s'utilise pour comparer deux ensembles A et B (éventuellement réduits à des points), il est noté $I_\epsilon(A, B)$. Il peut néanmoins être utilisé comme un opérateur unaire si on fixe un ensemble de référence. L'ϵ-indicateur entre deux ensembles A et B correspond au facteur minimum par lequel le front A doit être translaté pour faiblement dominer le front B. Il est formellement défini comme suit :

$$I_\epsilon(A, B) = min_{\epsilon \in \mathbb{B}}\{\forall v \in B, \exists u \in A : u_i - \epsilon \leq v_i \ \forall i \in \{1, ..., p\}\} \tag{6.4}$$

La figure 6.8 illustre l'utilisation de l'ϵ-indicateur pour comparer la solution A à la solution B.

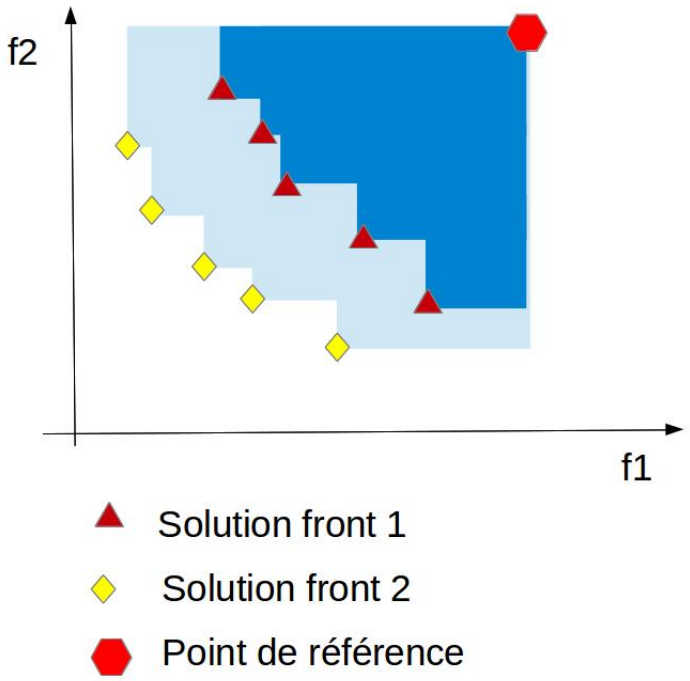

FIGURE 6.7 – Comparaison de deux fronts grâce à l'indicateur comparant leurs hypervolumes.

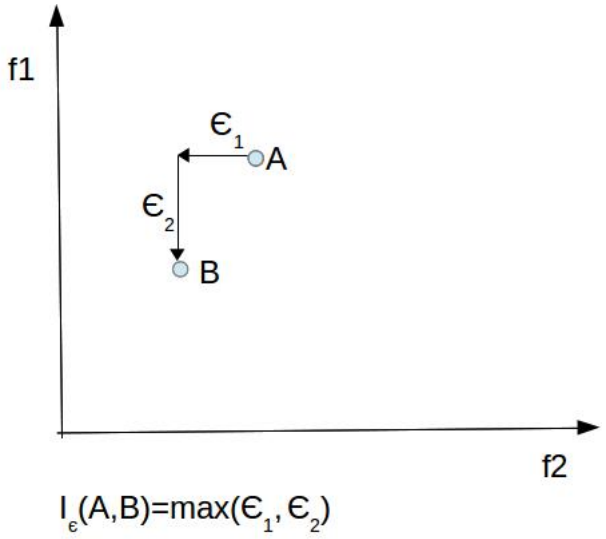

FIGURE 6.8 – ϵ-indicateur appliqué à la comparaison de 2 points.

Tout comme l'indicateur d'hypervolume, l'ϵ-indicateur est sensible à l'échelle des fonctions objectifs et par conséquent, son utilisation nécessite la normalisation des magnitudes de toutes les fonctions objectifs.

— **Indicateur *spread* :** Le *spread* est un indicateur unaire permettant de mesurer la qualité de la diversité d'un ensemble de solutions Pareto équivalents. Il a été introduit en 1999 par Ziztler et Thiele [211], et se définit comme ceci :

$$I_S(A) = \sqrt{\sum_{i=1}^{m} \max\{||a_i - b_i|| : a, b \in A\}}$$

Cet indicateur mesure l'expansion de l'ensemble mais ne mesure pas sa distribution. Il peut être utilisé lorsque le décideur souhaite privilégier la prise de décisions proches de l'optimum pour au moins un des objectifs.

4 Classification des approches de résolution

Cette section a pour but de donner une vue d'ensemble des approches existantes pour la résolution de problèmes d'optimisation combinatoire multi-objectif. Pour ce faire, nous allons tout d'abord présenter une classification des approches de résolution en fonction de la place accordé au décideur.

4.1 Optimisation multi-objectif et aide à la décision

Résoudre un problème d'aide à la décision multi-critère consiste à fournir au décideur la solution qui corresponde le mieux à ses désirs. Les approches de résolution sont donc fortement influencées par le rôle que prendra le décideur dans le processus de décision. Trois scénarios sur le rôle du décideur peuvent être distingués. Ils permettent de classifier les approches de résolution de la façon suivante :

— **Approches a *priori* :** Dans ce type d'approches, le décideur intervient en début de processus en fournissant des informations précises sur ses préférences. La résolution du problème se ramène alors à une modélisation des préférences du décideur permettant d'associer un poids à chaque objectif. Dans ce cas, on se ramène donc généralement à un problème mono-objectif que l'on résout avec une méthode d'optimisation classique dont une seule exécution fournira la solution recherchée. Cependant, la modélisation des préférences du décideur n'est pas une tâche facile. Il peut donc être nécessaire de mettre en place des processus de réajustement de cette modélisation en cours de résolution ou de lancer plusieurs exécutions avec des modélisations différentes. Ceci afin de trouver une solution qui satisfasse vraiment le décideur.

— **Approches a *posteriori* :** Dans ce type d'approches, au contraire, le décideur n'intervient qu'en fin de processus. L'objectif est donc de lui fournir l'ensemble des solutions Pareto optimales ou l'ensemble complet minimal afin qu'il puisse faire son choix. Pour cela, il faut résoudre un problème d'optimisation multi-objectif tel que défini dans les sections précédentes. L'avantage est qu'il n'est plus nécessaire de modéliser les préférences du décideur. En revanche, il faut fournir un ensemble de solutions, ce qui peut s'avérer coûteux en pratique. Par ailleurs le nombre de solutions obtenues peut rendre l'ensemble Pareto optimal difficile à analyser pour le décideur.

— **Approches interactives :** Dans ces approches, il existe une interaction directe et progressive entre le décideur et la méthode de résolution. Ainsi, à partir des connaissances acquises, le décideur peut définir ses préférences de manière compréhensible. Les échanges sont itérés plusieurs fois jusque satisfaction du décideur. L'inconvénient principal de ce type d'approches est que le décideur doit rester tout au long du processus de résolution.

Chacune de ces approches possède ses forces et ses faiblesses. En pratique, le choix du type d'approche utilisé dépend de la nature du problème et des disponibilités et habilités du

décideur. Dans ce manuscrit nous nous focalisons sur les méthodes *a posteriori*. Par la suite nous allons donc présenter avec plus de détails les approches de résolution existantes pour les problèmes d'optimisation combinatoire dans lesquels le décideur intervient *a posteriori*. Nous nous intéressons uniquement à la première phase du processus de résolution. C'est à dire à la méthode de résolution permettant de fournir un front Pareto et non à l'analyse de celui-ci par le décideur.

4.2 Méthodes de résolution

De même qu'en mono-objectif, deux types d'approches peuvent être utilisées. D'une part les approches basées sur des méthodes exactes et d'autre part, les méthodes d'approximation. Les méthodes exactes sont des méthodes qui garantissent l'obtention d'un ensemble de solutions Pareto optimales. Cependant, les problèmes d'optimisation multi-objectifs sont généralement des problèmes NP-difficiles. En effet, l'ensemble des problèmes qui sont NP-difficiles en mono-objectif le sont également en multi-objectif. D'autre part, de nombreux problèmes appartenant à la classe des problèmes polynomiaux P en mono-objectif sont NP-difficiles lorsqu'on les considère avec plusieurs objectifs [44]. Ceci rend difficile ou irraisonable l'application de méthodes exactes pour la résolution de beaucoup de problèmes multi-objectifs de grande instance. C'est pourquoi de nombreuses méthodes d'approximation ont été étudiées. Ces méthodes d'approximation peuvent être subdivisées en deux catégories : les heuristiques et les métaheuristiques. Les heuristiques sont des méthodes qui se basent sur les propriétés du problème à résoudre tandis que les métaheuristiques sont des méthodes génériques. Nous nous focaliserons ici sur les métaheuristiques. En outre, les méthodes approchées et exactes peuvent être combinées en des méthodes hybrides dont l'objectif est de tirer profit de la précision des méthodes exactes et de la rapidité des méthodes d'approximation. Néanmoins la coopération entre plusieurs méthodes est plus difficile en multi-objectif qu'en mono-objectif car elle pose de nouvelles problématiques et reste assez peu étudiée dans la littérature scientifique.

Dans cette partie, nous présenterons brièvement, pour ces trois types d'approches, les principales méthodes existantes.

Approches exactes

En optimisation multi-objectif, la plupart des méthodes de résolution sont basées sur des techniques de scalarisation. L'idée générale de ces méthodes est de se ramener à la résolution de plusieurs problèmes mono-objectif donnant chacun une solution Pareto optimale du front. Les trois principales méthodes de scalarisation sont :

— **La méthode de la somme pondérée :** L'idée de cette méthode est d'associer une pondération λ_k à chaque objectif f_k et de résoudre le problème P_λ suivant :

$$min(\sum_{k=1}^{m} \lambda_k f_k)$$

de sorte que :

$$\sum_{k=1}^{m} \lambda_k = 1$$
$$x \in X$$
$$\lambda_k \geq 0 \ \forall k$$

La solution de ce problème est une solution Pareto optimale. Il a été montré que pour toute solution supportée x^* il existe un vecteur λ telle que x^* soit une solution de P_λ. Une solution supportée étant une solution appartenant à la fermeture convexe de l'ensemble des solutions. En revanche, les solutions non supportées ne peuvent

être trouvées par cette méthode. Ainsi, pour les problèmes pour lesquels le front Pareto est convexe il sera possible de trouver l'ensemble du front Pareto avec cette méthode. En effet, l'ensemble Pareto optimal est alors défini par l'ensemble suivant :

$$\{x^{\lambda} = argmin_x(\sum_{k=1}^{m} \lambda_k f_k(x)) : \lambda = (\lambda_k)_k > 0, \sum_{k=1}^{m} \lambda_k = 1\}$$

En revanche si le front Pareto n'est pas convexe, il sera impossible de trouver l'ensemble du front avec cette approche. En effet, toutes les solutions non supportées sont inaccessibles.

Plusieurs méthodes de recherche d'un front Pareto se sont basées sur la méthode des sommes pondérées. Notamment, dans le cas de l'optimisation bi-objectif, des propositions de solutions pour adapter le choix des coefficients de pondérations de la somme afin d'obtenir un ensemble de solutions efficaces bien réparties ont été faites [93, 94].

— **La méthode d'ϵ-contraintes [64] :** Le principe de la méthode est de fixer un vecteur ϵ pour résoudre un problème mono-objectif de la forme :

$$min f_j(x)$$

de sorte que :

$$f_k(x) \leq \epsilon_k \forall k \neq j$$
$$x \in X$$

Il a été montré que les solutions de ces problèmes sont Pareto optimales et que toute solution Pareto optimale peut être trouvée par la méthode d'ϵ-contraintes. Cependant, les coefficients ϵ sont difficiles à choisir. En outre l'ajout des contraintes $f_k(x) \leq \epsilon_k \forall k \neq j$ rend souvent le problème NP-difficile même si sa version mono-objectif d'origine est polynomiale [44].

De même que pour la méthode des sommes pondérées des solutions ont été proposées pour adapter les choix des valeurs de ϵ afin d'obtenir un front Pareto bien réparti (voir par exemple [103]).

— **La méthode de Benson :** La scalarisation de Benson nécessite le choix d'une solution réalisable x_0 à partir de laquelle une solution Pareto optimale va être construite [20]. Le problème mono-objectif obtenu est le suivant :

$$min_x(\sum_{k=1}^{m} f_k(x)),$$

de sorte que :

$$f_k(x) \leq f_k(x_0) \forall k \neq j$$
$$x \in X$$

Cette méthode peut théoriquement permettre de trouver le front Pareto mais présente les mêmes désavantages que la méthode d'ϵ-contraintes. En effet il est difficile de choisir les solutions x_0 permettant d'obtenir un front bien diversifié et l'ajout des contraintes $f_k(x) \leq f_k(x_0) \forall k \neq j$ peut rendre le problème NP-difficile.

Il est à noter que ces méthodes de scalarisation sont souvent difficiles à mettre en oeuvre. En effet, soit elles ne permettent pas d'obtenir un ensemble complet de solutions. Soit elles impliquent la résolution de problèmes mono-objectifs NP-complet.

Néammoins, pour l'optimisation combinatoires bi-objectif, des méthodes exactes garantissant l'obtention d'un ensemble complet minimal existent. La méthode d'énumération ordonnée, par exemple, s'appuie sur un algorithme d'énumération ordonnée des solutions de la version mono-objective du problème pour explorer l'espace objectif de la version bi-objectif. La méthode de *Branch & Bound* , a également été étendue pour l'optimisation multi-objectif [163], l'idée est, que le concept des bornes, est remplacé par celui d'hypersurfaces définissant une région de l'espace objectif dominée ou dominante. La méthode la plus

répendue pour l'optimisation combinatoire bi-objectif est la méthode en deux phases [192]. La première phase consiste à chercher l'ensemble des solutions efficaces supportées externes grâce à la méthode des sommes pondérées. La deuxième phase utilise une méthode de *branch and bound* ou une méthode d'énumération ordonnée pour trouver l'ensemble des solutions non dominées par les solutions efficaces supportées (c'est à dire les solutions efficaces non supportées).

La méthode de programmation dynamique a été étendue à la résolution de problème multi-objectif [96]. C'est une des rares méthodes exacte d'optimisation multi-objectif qui puisse garantir de trouver un front Pareto complet minimal et qui puisse s'appliquer quel que soit le nombre d'objectifs. Cette méthode sera expliquée en détails par la suite et servira de base à la généralisation de DYNAMOP pour les problèmes multi-objectif.

Métaheuristiques

De nombreuses propositions ont été faites pour concevoir des métaheuristiques multi-objectif. L'objectif de ces méthodes est d'obtenir de bonnes approximations du front Pareto optimal. Les métaheuristiques multi-objectif peuvent être classées en deux sous catégories : D'une part les algorithmes évolutionnaires multi-objectifs, qui sont une généralisation de leurs homologues en mono-objectif et d'autre part, les métaheuristiques basées sur des recherches locales qui généralisent au cas multi-objectif les métaheuristiques à solutions uniques.

Les algorithmes évolutionnaires seront présentés en détails dans le chapitre suivant. Ils reprennent les même concepts de représentation génotypique, de croisement et/ou de mutation que leurs homologues en mono-objectif mais adaptent les concepts de sélection et d'élitisme afin de favoriser la convergence de la population vers un ensemble Pareto optimal bien diversifié. Il existe de nombreux paradigmes d'algorithmes évolutionnaires qui se différencient principalement de par les processus de sélection et d'élitisme utilisés. Nous pouvons ainsi distinguer deux grandes catégories d'algorithme évolutionnaire multi-objectif, d'une part ceux utilisant directement le principe de Pareto dominance dans le processus de sélection tels que MOGA [129], NSGA [164], NSGA II [40], SPEA [210] et SPEA II [209] et, d'autre part, ceux utilisant des indicateurs de performance tels que IBEA [207], HypE [14] et SMS-EMOA [22].

D'autre métaheuristiques à populations ont été adaptées au multi-objectifs, telles que les algorithmes de colonies de fourmis ou les algorithmes à essaims de particules. Les algorithmes de colonies de fourmis peuvent être classés en deux catégories. Ceux utilisant plusieurs colonies dont chacune s'occupe d'un objectif [121, 122] et ceux n'utilisant qu'une colonie mais des stratégies spécifiques pour le dépôt des phéromones [41, 62]. Ces stratégies sont généralement basées sur une somme pondérée des objectifs pouvant varier d'une fourmi à l'autre. L'adaptation des essaims particulaires est généralement basée sur l'utilisation d'une archive permettant de conserver les solutions potentiellement efficaces ou/et l'attribution de vecteurs de directions aux particules permettant de définir les objectifs qu'ils favorisent. Cette direction supplémentaire peut varier au fil des itérations et est fixée afin d'assurer une certaine distance entre les solutions et ainsi chercher à obtenir un front bien diversifié. Cependant il n'existe que très peu d'applications de ces méthodes à la résolution de problèmes d'optimisation combinatoire multi-objectif [31, 127, 141].

Dans le cadre de l'optimisation multi-objectif, les algorithmes de recherche locale sont étendus au cas où une population de solutions évoluant en parallèle est considérée. De plus, la recherche du meilleur voisin est adaptée pour se baser sur la relation de Pareto dominance. Les solutions efficaces trouvées sont généralement conservées dans une archive

qui est mise à jour au cours des itérations. Cette archive est alors également utilisée lors de la sélection des nouvelles solutions, c'est à dire qu'on sélectionne un voisin qui soit faiblement Pareto dominant par rapport à l'archive. Le premier algorithme de résolution de problème d'optimisation multi-objectif basé sur la recherche locale est probablement celui proposé par Serafini en 1992 [153]. Il s'agit d'une généralisation des algorithmes de recuit simulé. Depuis d'autres méthodes basées sur les recuits simulés ont été proposées telles que [35, 46, 184] ainsi que des méthodes basées sur la recherche locale ou tabou [67] et [11, 55, 97].

La principale problématique des métaheuristiques multi-objectifs est dans le mode de sélection des solutions à conserver pour la prochaine itération ou à utiliser pour construire de nouvelles solutions. Cette sélection doit servir à la fois la convergence de la population vers le front Pareto et sa diversité de manière à obtenir un ensemble de solutions bien réparties et étendu dans l'ensemble des solutions Pareto optimales. Différentes stratégies de sélections seront présentées dans le chapitre suivant, ces stratégies peuvent être utilisées avec n'importe quel type de métaheuristique multi-objectif.

Méthodes hybrides

De nombreuses méthodes combinant deux métaheuristiques ont été proposées. Les hybridations les plus courantes consistent à inclure une recherche locale dans un algorithme évolutionnaire multi-objectif. Ces méthodes sont basées sur l'observation que la plupart des algorithmes évolutionnaires multi-objectif classiques permettent d'obtenir un ensemble de solution Pareto équivalentes bien réparties mais dont la convergence est mauvaise alors que les algorithmes de recherche locale, à l'inverse, fournissent des solutions de bonne convergence mais de mauvaise diversité. Parmi elles, les méthodes [176, 177] permettent de résoudre efficacement des problèmes de routage de véhicules multi-objectif. Dans [82], ce type d'hybridation est appliqué avec succès au problème de couverture de graphe bi-objectif et dans [88] l'hybridation est utilisée pour résoudre un problème de voyageur de commerce avec profit. Pour chacune de ces méthodes la recherche locale est incluse au sein de l'algorithme évolutionnaire comme opérateur de mutation intelligent afin d'en améliorer la convergence. L'algorithme de Barichard et Hao [16] est basé sur la même idée mais la recherche locale est remplacée par un algorithme de recherche tabou. Cette méthode a été évaluée avec succès sur un problème de sac à dos multi-objectif. D'autres méthodes combinant plusieurs métaheuristiques ont été proposées telles que [181], où les solutions d'un algorithme de colonie de fourmis multi-objectif sont améliorées avec un algorithme de recherche locale ou encore [3] où un algorithme génétique multi-objectif est utilisé pour générer une population initiale pour un algorithme multi-objectif basé sur le recuit simulé.

En revanche assez peu d'hybridations entre méthodes exactes et approchées ont été proposées, ce qui est étonnant car ces approches sont prometteuses. Tout d'abord, il a été montré qu'il peut être intéressant d'utiliser la méthode de la somme pondérée pour générer une population possédant des solutions initiales efficaces [56, 128]. Brasseur et al. [18] ont proposé trois stratégies d'hybridation pour un problème d'ordonnancement de type *Job Shop*. Ce sont des hybridations entre la méthode à 2 phases [192] et un algorithme génétique multi-objectif. La première hybridation proposée dans [18] consiste à utiliser les solutions fournies par l'algorithme génétique pour définir des bornes qui seront utilisées par la méthode exacte. Dans les deux autres hybridations qu'ils proposent, la méthode exacte est utilisée pour intensifier la recherche dans les environs des meilleures solutions de l'algorithme génétique. La méthode proposée par Gandibleux et Fréville [54] pour résoudre un problème de sac à dos bi-objectif, combine une méthode exacte et un algorithme de

recherche tabou. La procédure exacte permet de trouver des coupes qui éliminent des portions de l'espace de recherche dans lesquels il ne peut exister de solution efficace. Ceci permet de restreindre l'espace de recherche de la recherche tabou. Dans [104] une méthode adaptative est proposée pour fixer les valeurs des coefficients ϵ de la méthode d'ϵ-contrainte et les sous problèmes mono-objectifs sont résolus par un algorithme évolutionnaire. Dans [87], un algorithme génétique est combiné avec un algorithme de *branch and cut* pour résoudre un problème de routage bi-objectif. Dans cette méthode l'algorithme de *branch and cut* est appliqué sur les solutions de l'algorithme génétique pour améliorer l'un des objectifs sans détériorer l'autre et donc assurer l'obtention d'une solution efficace.

5 Cadre applicatif : Le problème d'affectation d'unités multi-objectif

Traditionnellement, le problème d'affectation d'unités, ou *unit commitment problem*, est utilisé pour trouver le planning de mise en service des unités de production ainsi que les quantités exactes qu'elles produisent qui minimise les coûts de production. Néanmoins, l'intérêt porté à l'impact sur l'environnement dans le domaine de la production est de plus en plus important. En effet, en réaction à l'impact néfaste grandissant des activités industrielles sur l'environnement et sous pression de l'opinion publique, les gouvernements ont mis en place de nombreuses lois destinées à préserver l'environnement des activités humaines. Pour ces raisons, il devient avantageux de considérer une nouvelle modélisation du problème d'affectation d'unités qui permette la prise en considération de la minimisation des émissions en gaz à effet de serre tels que le CO_2 et le SO_2. Dans cette section, nous allons présenter une modélisation bi-objectif de ce problème, permettant de prendre en compte simultanément la minimisation des coûts de production et de l'émission de gaz à effet de serre. Puis nous présenterons un état de l'art des différentes méthodes déjà proposées pour le problème d'affectation d'unités avec prise en compte de l'impact écologiques.

5.1 Formalisation

Dans ce travail, les émissions en gaz nocifs à l'environnement, sont pris en compte en introduisant un objectif additionnel à minimiser au modèle présenté dans la section 2.2 du chapitre 2. Cet objectif est une fonction mesurant les émissions en SO_2 et CO_2 qui est représentée comme ceci :

$$f_{obj,2} = \sum_{t=1}^{T} \sum_{i=1}^{N} CE_i(p_{i,t}) u_{i,t}$$

Les notations utilisées sont les mêmes que celles introduites dans la section 2.2 du chapitre 2, et $CE_i(p_{i,t})$ représente la quantité de CO_2 et SO_2 émise par l'unité i pour produire la quantité $p_{i,t}$ durant la période de temps t :

$$CE_i(p_{i,t}) = b_{0,i} + b_{1,i} p_{i,t} + b_{2,i} p_{i,t}^2,$$

où $b_{0,i}$,$b_{1,i}$ et $b_{2,i}$ représentent des coefficients constants associés à l'unité de production i.

Cette modélisation a été reprise de la thèse de Viana [123]. En général, les unités ayant de forts coûts de production sont celles permettant d'émettre peu de gaz à effet de serre et inversement. Ainsi, les deux objectifs sont conflictuels et l'amélioration sur l'un des objectifs peut entraîner une détérioration sur l'autre objectif. C'est pourquoi, en général, il

n'y a pas une unique solution qui atteigne simultanément les valeurs optimales de chaque objectif. Un ensemble de solutions doit donc être proposé dans lequel chaque solution est meilleure que les autres sur un objectif, cet ensemble correspond au front Pareto. Dans cette thèse notre objectif sera donc de proposer un algorithme permettant d'obtenir, une approximation du front Pareto de bonne qualité, c'est-à -dire ayant une bonne convergence et une bonne diversité.

5.2 Analyse et complexité

La version mono-objectif étant déjà dans la classe des problèmes NP-Complet, la version multi-objectif est au moins aussi difficile.

En mono-objectif, une modélisation du problème comme un problème à deux niveaux hiérarchiques permet de se ramener facilement à un problème d'optimisation à variables binaires. En effet, il est possible de se focaliser sur le problème de planification des temps de marche et d'arrêt, qui est de fixer les variables binaires $u_{i,t}$ puis, d'obtenir en un temps polynomial les valeurs optimales des $p_{i,t}$ pour un planning marche/arrêt fixé. En effet, le problème de distribution de la production est alors un problème mono-objectif, continu et quadratique simple à résoudre.

De nombreuses méthodes de résolution du problème d'affectation d'unités mono-objectif sont donc basées sur ce découpage en deux sous-problèmes. Ce qui justifie d'ailleurs l'appellation classique du problème comme *unit commitment problem* qui met l'accent sur le problème du planning marche/arrêt. En particulier, les méthodes de programmation dynamique et les métaheuristiques pour le problème d'affectation d'unités utilisent toutes cette modélisation. C'est également le cas de la méthode de résolution proposée au chapitre 4 basée sur DYNAMOP.

En multi-objectif, le sous-problème de distribution multi-objectif a aussi de bonnes propriétés. Celui ci est, en effet, un problème d'optimisation bi-objectif continu dont les fonctions objectifs sont toutes deux convexes et dont l'ensemble de définition des variables est convexe. Or, il a été montré que pour ce type de problème toutes les solutions sont supportées [45]. C'est à dire que le front Pareto associé au sous-problème est défini par :

$$\{f_{1,\lambda}, f_{2,\lambda} | \lambda \in [0,1]\},$$

où $f_{1,\lambda} = f_1(p*_\lambda)$ et $f_{1,\lambda} = f_1(p^*_\lambda)$,avec p^*_λ solution du problème de distribution $\mathscr{D}(u,\lambda)$ defini par :

$$p^*_\lambda = arg(\min_p \sum_{t=1}^{T} \sum_{\substack{i \\ \text{s.t } u_i=1}} \lambda f_1((p_{i,t})_i)) + (1-\lambda) f_2((p_{i,t})_i)) \tag{6.5}$$

tel que :

$$\sum_{\substack{i \\ \text{s.t } u_i=1}} p_{i,t} = D_t \tag{6.6}$$

$$p_{i,min} \leq p_{i,t} \leq p_{i,max} \quad \forall i \text{ t.q. } u_i = 1 \tag{6.7}$$

Comme la version mono-objectif du problème est facile à résoudre, on peut facilement obtenir une bonne approximation de ce front en résolvant $\mathscr{D}(u,\lambda)$ pour différentes valeurs de λ.

Cependant, il n'existe pas de stratégie générique permettant d'exploiter une modélisation à deux niveaux multi-objectif pour construire des métaheuristiques explorant un

espace de recherche restreint. Afin de reprendre et d'exploiter les méthodes proposées pour la version mono-objectif du problème, nous allons donc devoir proposer une telle stratégie.

5.3 État de l'art

Bien que la prise en compte de l'impact environnemental de la production soit devenu un enjeu majeur, relativement peu d'études existent incluant la minimisation des émissions de gaz à effet de serre dans la résolution du problème d'affectation d'unités. En revanche, de nombreuses études ont été faites sur le sous-problème de distribution économique et écologique de la production parmi les unités planifiées comme étant allumées [4, 5, 74, 173]. Cependant, ces méthodes traitent le problème en incluant des composantes supplémentaires sinusoïdales aux fonctions objectif ou/et des contraintes non linéaires qui rendent le problème de distribution plus complexe qu'il ne l'est dans la version classique du problème d'affectation d'unités.

En comparaison au nombre d'études faites pour le problème de distribution multi-objectif, très peu d'études ont été faites intégrant la problématique écologique au problème d'affectation d'unités complet. De plus, parmi ces études, certaines traitent bien le problème des émissions en gaz en plus du problème de minimisation des coûts de production, mais se ramènent à une formulation mono-objectif. Certaines y parviennent en ajoutant une contrainte au modèle afin de fixer un niveau maximal d'émission à ne pas dépasser [26, 58, 120, 193]. D'autres considèrent bien un second objectif mais se ramènent à un problème mono-objectif en utilisant une technique de scalarisation [99, 148, 203].

Des méthodes de résolutions dans lesquelles le problème est modélisé comme un problème d'optimisation bi-objectif ont tout de même été proposées. Toutefois, il faut distinguer celles pour lesquelles le sous-problème de distribution de la production entre les unités allumées est traité comme un problème mono-objectif. Les méthodes [110, 206], par exemple, utilisent des algorithmes génétiques multi-objectifs manipulant des solutions binaires donnant les plannings des temps de marche et d'arrêt des unités ($u_{i,t}$), et trouvent la solution complète du problème (c'est-à-dire les valeurs des $p_{i,t}$) en résolvant le sous-problème de distribution comme un problème mono-objectif. La version mono-objectif du sous-problème est obtenue par la méthode de la somme pondérée, dont les pondérations sont fixées par rapport aux préférences de l'utilisateur.

Parmi les méthodes traitant vraiment le problème comme un problème multi-objectif que ce soit au niveau du problème de planification des temps de marche ou d'arrêt ou du problème de distribution, on peut tout d'abord citer le travail de Srinivasan [165]. Les auteurs utilisent un algorithme génétique pour fournir une population de solutions efficaces. Dans leur méthode, les solutions sont représentées par des vecteurs réels donnant des ratios correspondant à la contribution de chaque unité dans la production de la demande à chaque période de temps. Néanmoins l'algorithme génétique proposé n'explore pas directement le concept de Pareto dominance. En effet, l'algorithme appliqué est un algorithme génétique classique à l'exception du processus de sélection qui est basé sur une version spécifique de la sélection par tournoi. Dans cette version, deux individus sont sélectionnés aléatoirement dans la population et un objectif est également sélectionné aléatoirement parmi la liste des objectifs, puis les deux individus sont comparés par rapport à l'objectif sélectionné et le meilleur des deux pour cet objectif est conservé pour la prochaine génération.

La méthode proposée dans [123] est une méthode de recherche locale multi-objectif. Sa particularité est qu'elle utilise des voisinages dont la définition permet la prise en compte des contraintes et cherche ainsi à éviter les solutions non réalisables. Dans cette méthode la représentation utilisée est la représentation classique pour le problème d'affectation

d'unités, c'est à dire une représentation binaire fixant le planning de marche/arrêt des unités. La production de chaque unité est calculée en prenant en compte une direction de recherche, attachée à chaque voisinage de solutions, qui permet de définir l'importance accordée à chaque objectif et de résoudre le sous problème de distribution de la production en conséquence.

Dans [27], un algorithme de colonies d'abeilles manipulant des vecteurs binaires est utilisé pour construire, pour chaque unité, un planning de marche et d'arrêt satisfaisant les contraintes de temps minimal d'arrêt et de fonctionnement et permettant de répondre aux contraintes de respect des demandes et des réserves. Puis, un algorithme de colonies d'abeilles manipulant des vecteurs réels, est utilisé pour résoudre le problème de distribution économique et écologique de la production entre les unités planifiées allumées. La formulation du problème traité dans ce papier est toutefois légèrement différente de celle que nous avons choisi d'étudier. En effet, les fonctions de coût de production et de mesure de l'émission en gaz n'y sont pas quadratiques.

6 Conclusion

Dans ce chapitre, nous avons traité différents aspects de l'aide à la décision multicritère en général, et de l'optimisation multi-objectif en particulier.

Tout d'abord, nous avons défini ce que nous entendons par problème d'optimisation multi-objectif en en donnant une formalisation générale. Nous avons vu comment définir la solution de tels problèmes en introduisant les notions liées à la Pareto dominance et le concept d'ensemble complet minimal. Nous avons également expliqué comment la qualité d'une solution approchée peut être évaluée, ou comment faire pour comparer deux solutions approchées, à l'aide d'indicateurs de qualité.

Ensuite, nous avons présenté les différentes approches de résolution de problème d'optimisation multi-objectif. Nous avons tout d'abord passé en revue les principales méthodes exactes existantes, nous avons pu voir que l'application de ces méthodes est souvent difficiles, soit parce qu'elles ne peuvent prendre en compte que deux objectifs, soit parce qu'elles ne garantissent pas l'obtention d'un ensemble complet ou alors parce que leur application à certains problèmes serait trop coûteuse en temps de calcul et consommation mémoire. Nous avons également dressé un état de l'art des méthodes de type métaheuristiques. A l'inverse des méthodes exactes, ces méthodes sont très génériques, elles peuvent s'adapter à n'importe quel type de problème même si le nombre d'objectifs est important et sont peu coûteuse en terme de temps de calcul et de consommation mémoire. En revanche elles ne donnent aucune garantie quant à la convergence et à la complétude de l'ensemble solution fourni. Nous avons ensuite vu que les approches existantes peuvent être combinées pour donner naissance à des méthodes hybrides. Cependant nous avons pu voir qu'il existe relativement peu d'hybridations entre des méthodes exactes multi-objectif et des métaheuristiques multi-objectif. Dans cette partie, nous présenterons deux méthodes d'hybridation de ce type dont nous évaluerons l'intérêt et les possibilités de généralisation.

Dans ce chapitre, nous avons également présenté un problème applicatif, qui est une extension au cas multi-objectif du problème d'affectation d'unités étudié en première partie. Dans cette version, en plus de minimiser les coûts de production, nous cherchons à minimiser les émissions en gaz à effet de serre. Nous avons dressé un état de l'art des travaux ayant été faits pour résoudre ce problème et nous avons pu voir que très peu d'entre eux l'appréhendent complètement comme un problème bi-objectif. Par la suite, nous verrons que cela peut s'expliquer parce que, pour ce problème, le passage du mono-objectif au multi-objectif n'est pas évident car il n'est pas possible d'adapter simplement

les métaheuristique proposées pour la version mono-objectif.

Chapitre 7

MO-DYNAMOP

Dans ce chapitre vous trouverez :

- *Une explication détaillée de la méthode de programmation dynamique multi-objectif et des algorithmes évolutionnaires multi-objectif ;*
- *Une présentation de MO-DYNAMOP, qui est une hybridation entre la programmation dynamique multi-objectif et les algorithmes évolutionnaires multi-objectif que nous avons obtenue en étendant DYNAMOP au cas multi-objectif.*

Les travaux détaillés dans ce chapitre ont été présentés lors de la conférence MIC 2015 [80].

1 Introduction

Dans la première partie de ce manuscrit nous avons présenté DYNAMOP, une nouvelle approche de résolution pour une certaine classe de problèmes d'optimisation combinatoire mono-objectif. DYNAMOP est une métaheuristique basée sur la programmation dynamique et nous avons vu qu'elle permet d'obtenir de très bons résultats sur des problèmes d'optimisation combinatoire mono-objectif. Néanmoins, dans un cadre applicatif réel, l'optimisation combinatoire est généralement utilisée comme un outil d'aide à la décision permettant à l'utilisateur de faire les choix qui lui seront les plus profitables. Dans ce contexte, l'optimisation mono-objectif est souvent inadaptée. En effet, généralement, la prise d'une décision ne peut se baser sur un unique critère et, il est délicat d'établir des ordres de préférence entre les critères en jeu. Pour cette raison, l'optimisation multi-objectif est devenue un enjeu majeur. De plus, la littérature regorge d'exemples où l'adaptation à l'optimisation multi-objectif de méthodes performantes en mono-objectif donne d'excellents résultats. Tout ceci motive le travail qui sera présenté dans cette partie, à savoir une adaptation du concept de DYNAMOP pour l'optimisation multi-objectif, MO-DYNAMOP.

A priori, l'adaptation de DYNAMOP au multi-objectif semble intéressante. En effet, il s'agit d'une méthode hybride entre un algorithme évolutionnaire et une méthode de résolution exacte, la programmation dynamique multi-objectif. Or, la programmation dynamique multi-objectif est une méthode très puissante, puisqu'elle permet de traiter une large classe de problèmes, sans contraintes forte sur la nature des fonctions objectif ou de l'espace de recherche et qu'elle permet d'obtenir un front de Pareto complet. Cependant, comme son homologue en mono-objectif, elle souffre de la fameuse *"malédiction de la dimensionnalité"*, qui est encore aggravée par le nombre d'objectifs à traiter et rend souvent son application impossible à la résolution de problèmes de taille réelle. À contrario, les algorithmes évolutionnaires multi-objectif, ont généralement des difficultés à converger vers le front Pareto optimal, mais sont peu coûteux en temps de calcul. Il semble donc qu'il pourrait être intéressant d'hybrider ces deux approches de manière à tirer avantage de leur

qualités respectives tout en en minimisant les inconvénients. C'est ce que l'on cherchera à faire en proposant MO-DYNAMOP.

Dans ce chapitre, nous allons expliquer en détail MO-DYNAMOP, l'adaptation de DYNAMOP à la résolution de problèmes multi-objectif. Pour ce faire, nous allons tout d'abord expliquer en détails les principes de la programmation dynamique multi-objectif ainsi que des algorithmes évolutionnaires multi-objectif. Nous commencerons ensuite par introduire l'idée générale de la méthode, puis nous consacrerons une section pour détailler chacun de ces modules. Enfin nous conclurons par une discussion sur l'intérêt potentiel de MO-DYNAMOP et les améliorations possibles à apporter.

2 Résoudre des problèmes multi-objectifs avec la programmation dynamique

La programmation dynamique a été étendue à la recherche du front Pareto des problèmes d'optimisation multi-objectif par Klötzler en 1978 [96]. Dans cette section, nous allons expliquer en détail le principe de cette généralisation. Pour cela nous allons tout d'abord expliquer l'idée générale sous-tendant la programmation dynamique multi-objectif, puis nous donnerons une formalisation de cette méthode.

2.1 Principe

La programmation dynamique multi-objectif se fonde sur une généralisation du principe d'optimalité de Bellman au concept de dominance. Cette généralisation peut se formuler de la façon suivante :

"Une politique non dominée est composée de sous-politiques non dominées"

Cette méthode est utilisée pour résoudre des problèmes d'optimisation multi-objectifs où l'on cherche l'ensemble des séquences de décisions Pareto optimales $(d_1, d_2, ..., d_n)$. L'algorithme se base sur le constat qu'une séquence Pareto optimale $(d_1, d_2, ..., d_n)$ sera composée de sous-séquences Pareto optimales $(d_1, d_2, ..., d_k)$ avec $k < n$. L'idée est que l'on peut construire le front Pareto du problème en combinant les solutions issues des fronts Pareto de sous-problèmes correspondant au choix de séquences de décisions plus courtes. Comme dans la programmation dynamique classique, la solution est construite de manière ascendante. C'est-à-dire que les fronts Pareto associés aux courtes séquences sont construits en premiers puis sont utilisés pour construire les fronts associés aux séquences plus longues.

2.2 Équation fonctionnelle

Un algorithme de programmation dynamique multi-objectif peut être défini par son équation fonctionnelle. Cette équation est une généralisation de l'équation fonctionnelle de la programmation dynamique classique et peut se formuler comme ceci :

$$\mathscr{F}(f(S)) = \mathscr{F}(\cup_{d \in D(S)} \{R(S,d) \circ e, e \in \mathscr{F}(f(T(S,d)))\})$$

Elle utilise exactement les même concepts d'états (noté S), d'espace de décisions (noté $D(S)$) et de fonction de transition (noté $T(S,d)$) que ceux utilisés en mono-objectif (voir section 2.2 du chapitre 2). En revanche les éléments suivants sont spécifiques à la programmation dynamique multi-objectifs :

— **Ensemble solution dans l'espace objectif :** $f(S)$ représente l'ensemble des vecteurs objectifs associés aux solutions réalisables du sous problème défini par l'état S.
— **Ensemble Pareto dominant :** $\mathscr{F}(E)$ désigne l'ensemble des éléments de E qui sont Pareto dominants pour cet ensemble. Ainsi $\mathscr{F}(f(S))$ désigne l'ensemble des solutions Pareto optimales pour le sous problème défini par l'état S. L'objectif de la programmation dynamique est donc de calculer $\mathscr{F}(f(S^*))$, où S^* est l'état objectif, défini de la même manière que dans le cas mono-objectif (voir section 2.2 du chapitre 2).
— **Fonction coût :** La fonction coût ou fonction *reward* R est une fonction à valeur dans $\mathbb{R}^m$. Elle dépend d'un état S et d'une décision d et représente les coûts ou les profits pour chaque objectif qui peuvent être attribués à la décision d si elle est prise à partir de l'état S. On notera $R_i(S,d)$ le coût associé au i^{eme} objectif. Comme en mono-objectif, la fonction coût $R(S,d)$ doit être séparable par rapport au coûts ou profits qui ont été attribués à toutes les autres décisions. Elle ne doit dépendre que de S et de d. La valeur d'une solution au problème est la combinaison des coûts pour la séquence des décisions prises en partant de l'état initial S^*.
— **États terminaux :** Les états terminaux sont définis comme dans le cas mono-objectif, c'est-à-dire à l'aide d'une fonction de base f_b. Mais le cas multi-objectif implique de redéfinir les évaluations des états terminaux en des termes ensembliste :
  — Si $f_b(S) = 1$, c'est-à-dire si S est un état terminal valide alors $f(S) = \mathscr{F}(f(S)) = \{0\}$, où 0 correspond au vecteur nul de dimension m.
  — Si $f_b(S) = 2$, c'est-à-dire si S est un état terminal invalide alors $f(S) = \mathscr{F}(f(S)) = \{INF\}$, où INF correspond au vecteur de dimension m dont les valeurs INF_i sont égales à $\pm\infty$ en fonction de la nature de l'objectif i.

2.3 Représentation graphique

De façon similaire au cas mono-objectif, un problème de programmation dynamique multi-objectif peut être modélisé comme un problème de plus court chemin dans un graphe. Ce graphe, appelé graphe d'états est, comme en mono-objectif, un graphe orienté et acyclique. Il est défini de la même manière, ainsi chaque état S de l'espace d'états est modélisé par un sommet et chaque décision $d \in D(S)$ est modélisée par un arc reliant S à $T(S,d)$. La différence est que les arcs sont évalués par plusieurs objectifs calculés avec la fonction coût $R(S,d)$. L'état initial S^* correspond à un sommet qui n'est le sommet final d'aucun arc. Les états terminaux valides correspondent à des états à partir desquels aucun arc ne part. En revanche si l'état S_T est un état final mais n'est pas un valide (si $f_b(S_T) = 2$), un sommet fictif S_f est ajouté de sorte que l'arc joignant le sommet associé à S_T à S_f soit évalué par le vecteur INF. L'objectif du problème est alors de trouver l'ensemble des chemins Pareto optimaux reliant le sommet initial S^* à un sommet terminal. La Figure 2.3 illustre cette modélisation dans le cas où m, le nombre d'objectifs, est égal à 2. Le sommet vert correspond au sommet initial, le sommet rouge est un sommet terminal correspondant à un état terminal valide, le sommet rose correspond à un état terminal non valide et le sommet orange à un état fictif.

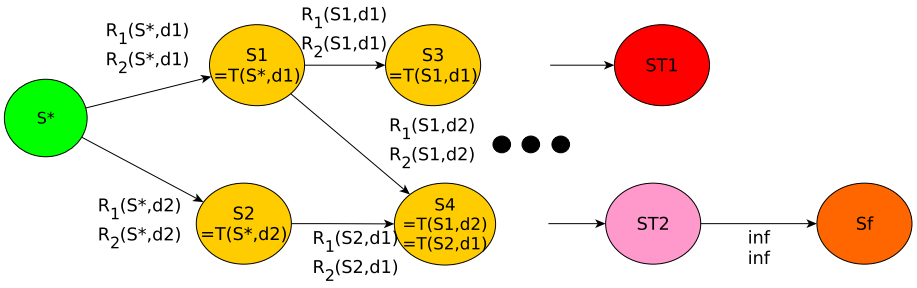

FIGURE 7.1 – Représentation graphique de la programmation dynamique multi-objectif.

3 Les algorithmes évolutionnaires pour les problèmes d'optimisation multi-objectif

De nombreuses propositions ont été faites pour adapter les algorithmes génétiques ou, plus généralement, évolutionnaires, à la recherche d'un front Pareto optimal d'un problème d'optimisation muti-objectif. Le principe général de ces algorithmes est le même qu'en mono-objectif, mais, le but est de faire converger la population vers le front Pareto optimal. La sélection et le remplacement ne peuvent plus s'appuyer directement sur les valeurs des fonctions objectifs associées aux solutions, car celles-ci ne permettent pas d'obtenir un ordre total. C'est pourquoi de nouveaux critères d'évaluation des individus sont mis en place. Le choix de ces critères constitue la principale distinction entre les algorithmes proposés dans la littérature. Ils doivent permettre de mesurer l'impact de la solution sur la qualité de l'approximation du front de Pareto par la population. Plus précisément, ils doivent permettre de mesurer l'apport de la solution sur la convergence et la diversification de l'ensemble solution. Ainsi, la conception d'algorithmes évolutionnaires multi-objectifs se base sur ces trois principes [208] :

— Assignation d'une fitness : Le rôle de ce processus est de privilégier les solutions proches du front Pareto. Il doit permettre d'assurer la convergence de l'ensemble solution fourni par l'algorithme vers le front Pareto.
— Assignation d'une valeur de diversification : Cette stratégie a pour but d'éviter la convergence de la population vers un petit sous-ensemble de solutions du front Pareto et donc d'assurer la bonne diversité de l'ensemble solution.
— Élitisme : La notion d'élitisme a pour but de préserver et d'utiliser les solutions élites qui sont les solutions non dominées trouvées par l'algorithme.

Dans cette section nous présenterons les différentes stratégies existantes pour assigner une valeur de qualité ou fitness et une valeur de diversification à une solution, ainsi que les stratégies d'élitisme existantes. A partir de là, il sera facile d'introduire les principaux algoritmes évolutionaires multi-objectifs.

3.1 Assignation d'une fitness

Cette mesure permet de classer les individus de la population en fonction de leur proximité au front Pareto optimal. Nous proposons de classer les méthodes d'assignation de fitness en deux familles. D'une part, les méthodes basées sur la relation de dominance et d'autre part les méthodes basées sur l'utilisation d'un indicateur de qualité.

Méthodes basées sur une relation de dominance :

Une relation de dominance, telle que la relation de Pareto dominance, est utilisée pour classer les solutions de la population. Cette méthode a été introduite pour la première fois en 1989 par Goldberg [60]. Il existe trois stratégies principales [208] :

— **"Rang de dominance" [49] :** Cette stratégie consiste à assigner à chaque individu un indice de qualité qui correspond au nombre de solutions de la population qui dominent cet individu. Plus ce nombre est faible plus la qualité de l'individu est élevée.

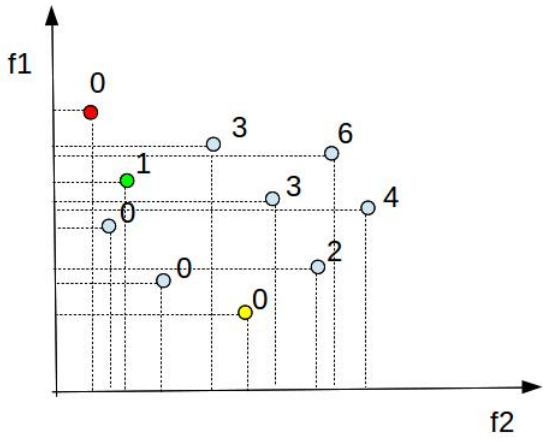

FIGURE 7.2 – Rang de dominance.

La figure 7.2 montre un exemple de solutions représentées dans l'espace objectif évaluées avec la méthode "rang de dominance" dans le cadre d'un problème où les objectifs doivent être minimisés.

— **"Compte de dominance" :** Cette stratégie consiste à assigner à chaque individu un indice de qualité qui correspond au nombre de solutions de la population qui sont dominées par cet individu. Plus ce nombre est grand plus l'individu est de bonne qualité.

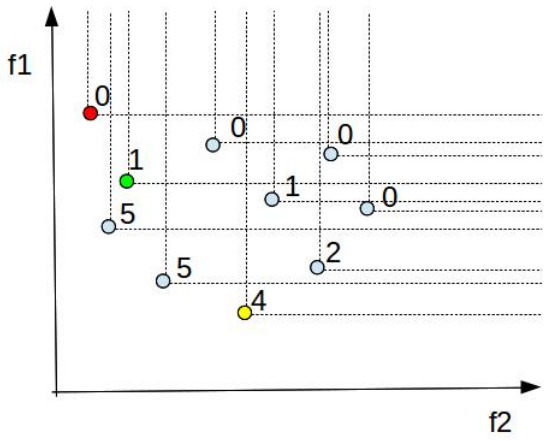

FIGURE 7.3 – Compte de dominance.

La figure 7.3 montre les mêmes solutions que la figure 7.2 mais cette fois évaluées avec la stratégie "Compte de dominance". Dans ce cas les solutions sont classifiées

par ordre décroissant par rapport à la valeur qui leur est attribuée. Nous pouvons observer que les résultats obtenus sont différents. En effet, dans le cas où le *Rang de dominance* est utilisé, il y a 7 niveaux hiérarchiques dans la classification des solutions alors qu'avec *compte de dominance* il n'y en a que 6. De plus, la solution rouge est classée première si le *Rang de dominance* est utilisé mais dernière dans le cas où on utilise compte de dominance. Pour la solution jaune, le résultat est également différent, avec le *Rang de dominance* elle est classée première, mais ce n'est plus le cas avec *compte de dominance*. De même la solution verte est classée dans le second niveau de hiérarchisation des solutions avec la première technique d'évaluation mais dans l'avant dernier avec la seconde technique.

— **"Profondeur de dominance" [60] :** Dans cette stratégie les individus sont regroupés par front Pareto de différents rangs. C'est le rang du front Pareto auquel il appartient qui permet d'évaluer un individu. Plus ce rang est faible mieux c'est. Les fronts sont construits récursivement comme ceci :
  — Les individus du front de rang 0 sont ceux qui ne sont dominés par aucune solution de la population.
  — Les individus du front de rang n sont ceux qui ne sont dominés par aucune des solutions de l'ensemble de la population à laquelle ont été enlevés tous les individus de rang inférieur à n.

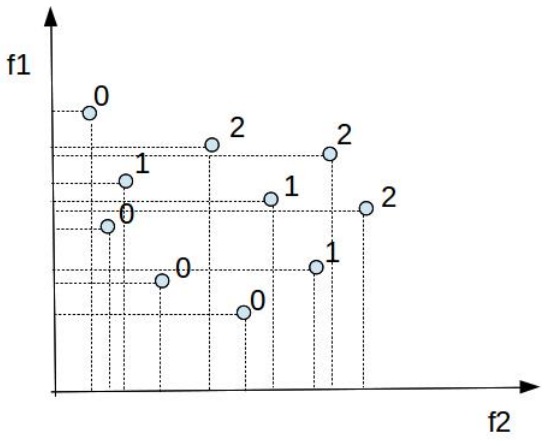

FIGURE 7.4 – Profondeur de dominance.

La figure 7.4 montre toujours la même population mais maintenant évaluée avec la stratégie "Profondeur de dominance". Une fois encore la classification des individus est différente.

Méthodes basées sur un indicateur de qualité :

Ces méthodes utilisent un indicateur de qualité (tel que présenté dans la section 3.2 du chapitre précédent) pour formuler le but de l'optimisation. Ainsi, le choix de l'indicateur représente l'objectif général du processus de recherche et les solutions de la population sont évaluées en fonction de cet indicateur. Nous verrons par la suite plusieurs exemples d'algorithmes utilisant un indicateur pour évaluer la qualité des individus.

3.2 Assignation d'une valeur de diversification

L'objectif de la mesure de diversification est d'évaluer chaque individu en fonction de l'impact qu'il a sur le critère de diversification de la population. Là encore plusieurs

méthodes ont été proposées :

— **Les méthodes noyaux :** L'estimation de la densité est la somme d'une fonction K, appelée noyau, mappée sur la distance entre une solution et les autres solutions de la population. La distance entre deux solutions est la distance entre leurs images dans l'espace objectif. On peut par exemple utiliser la distance euclidienne ou la distance de Manhattan. La figure 7.5 illustre cette méthode avec la distance euclidienne. La technique appelée *Partage de fitness* est souvent utilisée et fait partie des méthodes noyaux [61]. Cette technique consiste à dégrader la valeur de fitness (mesure de qualité) $f(x)$ associée à chaque solution en tenant compte du nombre de solutions se trouvant à proximité dans l'espace objectif. La nouvelle valeur de fitness devient alors :

$$f'(x) = \begin{cases} 0 & \text{si } m(x) = 0 \\ \frac{f(x)}{m(x)} & \text{sinon} \end{cases} \tag{7.1}$$

où $m(x)$ est parfois appelé compteur de niche. Il se calcule de la manière suivante :

$$m(x) = \sum_{y \in Population} sh(dist(x,y)) \tag{7.2}$$

$$sh(d(x,y)) = \begin{cases} 1 - \frac{dist(x,y)}{\sigma} \text{ si } dist(x,y) \leq \sigma \\ 0 \text{ sinon} \end{cases} \tag{7.3}$$

Le paramètre σ est un seuil au-delà duquel deux solutions sont considérées comme suffisamment distantes et ne faisant pas partie d'une même niche.

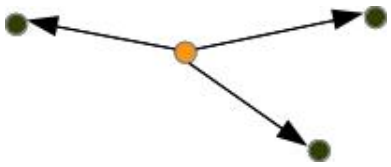

FIGURE 7.5 – Méthodes à noyaux.

— **Plus proche voisin :** Avec cette méthode l'apport à la diversité de la population d'une solution est mesuré par la distance entre une solution et ses k solutions voisines les plus proches ou par le volume de l'hypersphère définie par ses k solutions voisines les plus proches. La figure 7.6 illustre cette mesure.

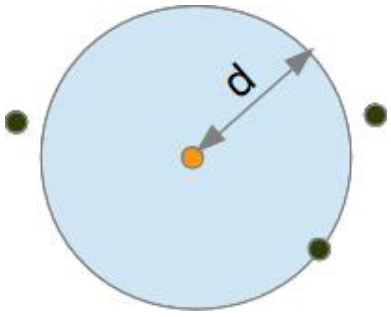

FIGURE 7.6 – Le plus proche voisin de la solution orange se trouve à la distance d.

— **Méthode des histogrammes :** L'espace de recherche est divisé en hypercubes de même taille, la mesure de diversification d'une solution est alors le nombre de

solutions appartenant au même hypercube. Cette méthode est illustrée par la Figure 7.7.

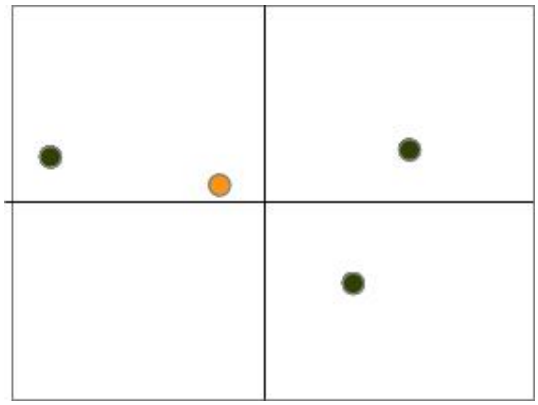

FIGURE 7.7 – Méthodes des histogrammes.

3.3 Elitisme

L'élitisme est un procédé utilisé pour garantir la sauvegarde des meilleures solutions d'une génération à l'autre. Il y a deux approches possibles. Soit les meilleurs solutions sont conservées au sein de la population, c'est-à-dire qu'on utilise une méthode garantissant leur survie au processus de remplacement. Soit elles sont sauvegardées dans une archive. Cette archive peut être de taille fixe ou simplement bornée et est mise à jour à chaque itération en se basant sur la relation de dominance. Si l'archive est utilisée dans le processus de sélection on parle d'élitisme actif, dans le cas contraire on parle d'élitisme passif.

3.4 Présentation des algorithmes

Nous allons présenter les algorithmes évolutionnaires d'optimisation multi-objectif les plus utilisés. Nous considérerons séparément ceux utilisant directement le principe de dominance pour la procédure d'assignation de fitness et ceux utilisant des indicateurs de qualité.

Algorithmes utilisant directement le principe de dominance

Ces algorithmes sont ceux utilisant une procédure d'assignation de fitness basée sur la relation de dominance. Cette fitness permet de classer les individus en fonction de leur proximité avec le front Pareto mais elle ne garantit pas la construction d'une approximation du front ayant une bonne diversification. C'est pourquoi les individus se voient également assigner une mesure de diversification. Ainsi les individus sont classés en considérant dans un premier temps leur fitness puis leur mesure de diversification.

Les principaux algorithmes de ce type sont les suivants :

- **MOGA (*Multi-Objective Genetic Algorithm*) [129] :**
  - Assignation de fitness : Rang de dominance.
  - Mesure de diversité : Méthode du noyaux.
  - Élitisme : Pas d'élitisme.
- **NSGA *(Non-dominated Sorting Genetic Algorithm)* [164] :**
  - Assignation de fitness : Profondeur de dominance.

— Mesure de diversité : Plus proche voisin en utilisant la *crowding distance*.
— Élitisme : Pas d'élitisme.

— **NSGA II *(Non-dominated Sorting Genetic Algorithm II)* [40] :**
— Assignation de fitness : Profondeur de dominance.
— Mesure de diversité : Plus proche voisin en utilisant la *crowding distance*.
— Élitisme : Lors de l'étape de remplacement les individus parents et enfants sont classés selon leurs fitness, puis à fitness égale selon la valeur de leur apport en diversification, les meilleurs individus de ce classement sont ceux gardés pour constituer la génération suivante.

— **SPEA2 *(Strength Pareto Evolutionary Algorithm 2)* [209] :**
— Assignation de fitness : Dominance compte puis rang de dominance.
— Mesure de diversité : Plus proche voisin.
— Élitisme : Utilisation d'une archive de taille fixe.

3.5 Algorithmes utilisant des indicateurs

Plus récemment des méthodologies basées sur l'utilisation d'un ou plusieurs indicateurs de qualité permettant d'évaluer l'apport d'une solution d'un point de vue convergence et/ou diversification ont été proposées. Ces méthodes n'utilisent pas forcément les deux processus d'assignation car l'indicateur de performance peut être choisi par rapport au critère que l'on souhaite évaluer (convergence et/ou diversification). Parmi ces algorithmes les plus couramment utilisés sont :

— **IBEA (*Indicator-Based Evolutionary Algorithm*) [207] :**
— Assignation de fitness : Soit un indicateur binaire I, la fonction fitness se calcule comme suit :

$$fitness(x) = \sum_{y \in P} (-e^{-I(y,x)/\kappa}) \tag{7.4}$$

où κ est un réel à fixer. Cette fonction mesure la perte en qualité si la solution x est retirée de la poulation P.
— Mesure de diversité : Aucune.
— Elitisme : Remplacement élitiste. Les individus ayant les meilleurs fitness parmi les parents et les enfants sont sélectionnés pour appartenir à la génération suivante.

— **SMS-EMOA (*Multiobjective selection based on dominated hypervolume*) [22] :**
— Assignation de fitness : Profondeur de dominance.
— Mesure de diversité : Indicateur hypervolume.
— Élitisme : Lors de l'étape de remplacement les individus parents et enfants sont classés selon leurs fitness, puis à fitness égales selon la valeur de leur apport en diversification, les meilleurs individus de ce classement sont ceux gardés pour constituer la génération suivante.

Les différents algorithmes introduits dans cette section nous seront utiles dans la suite de ce manuscrit. En effet, la méthode que nous présentons dans la section suivante, MO-DYNAMOP, pourra utiliser n'importe lequel de ces algorithmes évolutionnaires.

4 MO-DYNAMOP : Idée générale

L'idée générale de MO-DYNAMOP est similaire à celle de DYNAMOP (cf chapitre 3), sinon que la programmation dynamique classique est remplacée par son homologue en multi-objectif et que l'algorithme génétique est remplacé par un algorithme évolutionnaire multi-objectif.

L'objectif de MO-DYNAMOP est donc d'utiliser un algorithme évolutionnaire multi-objectif pour chercher l'ensemble des chemins Pareto optimaux dans le graphe d'états associé à une équation fonctionnelle définissant un problème de programmation dynamique multi-objectif.

En fait, MO-DYNAMOP est simplement un algorithme évolutionnaire multi-objectif qui manipule des solutions représentées sous forme de chemins dans un graphe d'états multi-objectif. La figure 7.8 représente un graphe d'états dans le cas d'un problème d'optimisation bi-objectif. Dans ce graphe, le chemin composé des sommets rouges peut correspondre à un individu solution de MO-DYNAMOP. L'espace de recherche, c'est à dire l'ensemble des chemins solutions, est l'ensemble des chemins reliant l'état initial S^* à un état terminal. Concrétement une solution est donc représentée par une séquence d'états $(S_t)_{t=0...T}$, de sorte que l'état S_0 corresponde à l'état initial S^* et que l'état S_T corresponde à un état terminal.

Avec cette représentation, chaque objectif i associé à une solution est évalué comme la somme des valeurs des arcs correspondant à cet objectif, c'est-à-dire comme la somme des $R_i(S_t, T^{-1}(S_t, S_{t+1})$ que nous noterons par la suite $R_i(S_t, S_{t+1})$. Comme pour DYNAMOP,

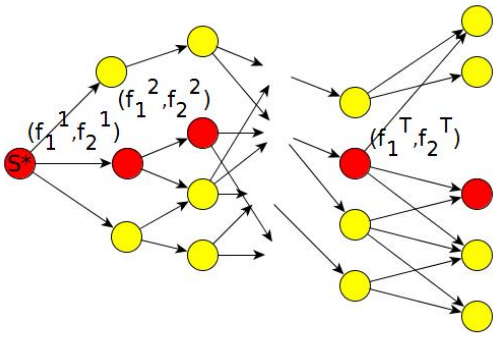

FIGURE 7.8 – Une solution est représentée comme un chemin du graphe d'états multi-objectif.

des opérateurs évolutionnaires pensés pour tirer parti de cette représentation sont proposés. Le principe de ces opérateurs restent inchangé. Ainsi, la mutation est conçue de façon à minimiser le nombre d'arcs qu'elle modifie et le croisement de manière à construire un nouvel individu possédant un nombre minimal d'arcs n'appartenant pas à l'un de ses parents. L'intérêt des opérateurs de mutation proposés est le même qu'en mono-objectif, à savoir que cette approche prend en considération la propriété de localité de l'opérateur et permet donc que la solution mutée soit phénotypiquement proche de la solution d'origine. Ceci confère à la mutation une meilleur localité et un plus grand pouvoir d'intensification.

De même, en ce qui concerne le croisement, l'intérêt des opérateurs proposés tient toujours à l'argument de l'héritabilité phénotypique. En effet, si l'individu enfant hérite des valeurs R_i de ses parents, il y a un lien direct entre les valeurs objectifs de l'individu

enfant et celles des individus parents. Néanmoins, cet argument est moins fort qu'en mono-objectif. En effet, dans le cadre de l'optimisation multi-objectif, même avec une bonne séparabilité des fonctions objectifs, il est plus courant d'obtenir un individu de mauvaise qualité à partir de deux individus de bonnes qualités et inversement. Ceci est dû au fait qu'il est possible qu'un individu composé de sous-solutions Pareto optimales (c'est à dire de sous-chemins Pareto optimaux) soit éloigné du front Pareto (si chacune de ses sous-solutions est bonne pour un objectif différent par exemple).

De manière similaire à ce qui est fait dans DYNAMOP, des opérateurs évolutionnaires intelligents sont utilisés en complément. Leur principe est de chercher un front de solutions Pareto équivalentes dans un graphe d'états restreint défini à partir d'une (ou de plusieurs) solution(s) courante(s). Pour ce faire une méthode d'optimisation multi-objectif exacte, telle que le programmation dynamique multi-objectif, est utilisée. Les solutions ainsi obtenues sont ajoutées à la population enfant.

La figure 7.9 illustre les principales modifications qui sont appliquées à DYNAMOP pour l'adapter à l'optimisation multi-objectif.

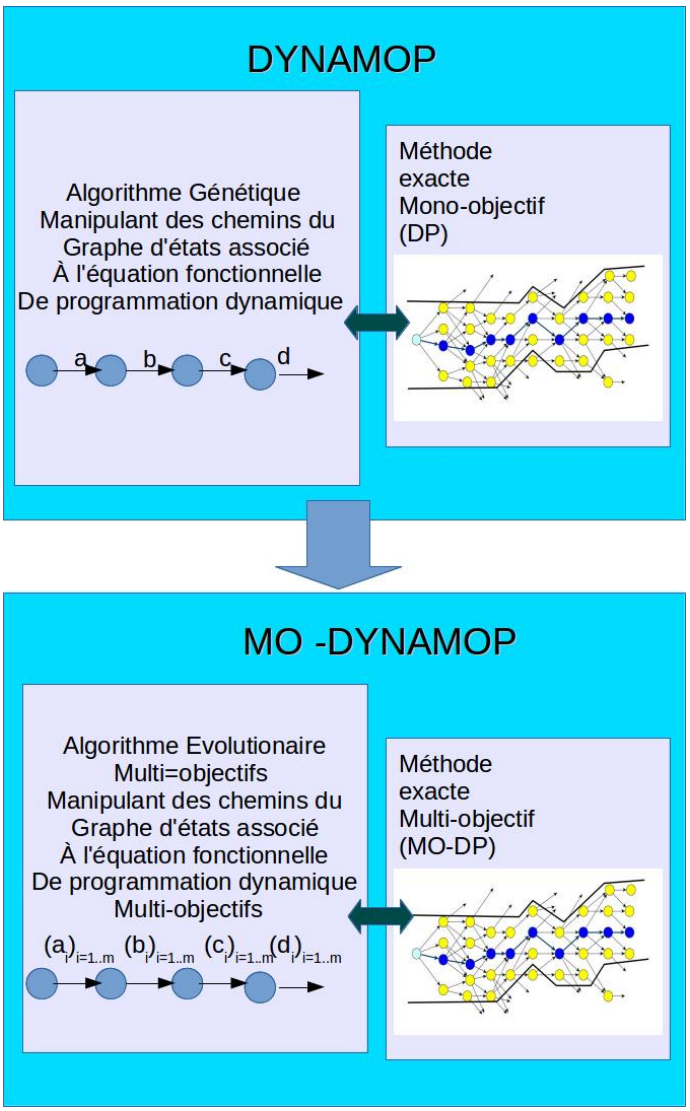

FIGURE 7.9 – Passage de DYNAMOP à MO-DYNAMOP.

5 Détails des modules de MO-DYNAMOP

Dans cette section nous présentons MO-DYNAMOP en détails. Pour ce faire nous présentons les spécificités de chaque module.

5.1 Choix des composantes du graphe d'états et représentation des solutions

La représentation des solutions sera totalement déterminée par le choix des composantes du graphe d'états. En effet, comme expliqué précédemment, une solution est représentée par un chemin du graphe d'états, autrement dit par une séquence d'états. Le choix des composantes de ce graphe, c'est à dire de l'espace d'état $\mathcal{S}$, des espaces de décisions $D(S)$, de la fonction coût R, de la fonction de transformation T et de la fonction de base f_b est donc une étape primordiale.

De même que dans le cas mono-objectif, ces choix doivent être motivés par l'idée que la recherche d'un sous-chemin liant deux états du graphe doit être simple, c'est-à-dire ne pas nécessiter un temps de calcul déraisonnable. Il peut donc être intéressant de relaxer le problème. Ce qui revient à élargir la définition des espaces de décisions $D(S)$ en autorisant certaines décisions qui ne permettent pas de respecter toutes les contraintes. Dans ce cas, la violation des contraintes relâchées doit être gérée en ajoutant des pénalisations sur toutes les composantes R_i de la fonction *reward*.

Une fois toutes les composantes de l'équation fonctionnelle de la programmation dynamique fixées, la représentation d'un individu correspond à une séquence d'états d'un chemin valide du graphe liant l'état initial S^* à un état terminal $(S_t)_t$ et à la séquence des valeurs des arcs $(e_{t,i})_{t,i}$ correspondants, pour chaque objectif i.

Autrement dit le couple $(S_t, e_t)_{t=1..n}$ correspond à un individu valide si et seulement si il existe une séquence $(d_t)_t$ telle que :

$$d_t \in D(S_{t-1}) \ \forall t$$
$$S_t = T(S_{t-1}, d_t) \ \forall t$$
$$e_{t,i} = R_i(S_{t-1}, d_t) \ \forall t, i$$

$$f_b(S_t) \begin{cases} = 0 & si \quad t < n \\ > 0 & sinon \end{cases}$$

En posant $S_0 = S^*$.

5.2 Évaluation

La stratégie d'évaluation est la même qu'en mono-objectif, excepté qu'elle est mise en oeuvre pour chaque objectif. Le but est de tirer profit de la séparabilité des fonctions objectif par rapport aux arcs. Autrement dit, après une mutation on ré-évalue l'individu en utilisant une stratégie d'évaluation itérative qui peut être définie comme suit : Si à la génération i l'individu S^i est obtenu en modifiant l'état t, S_t^{i-1}, de l'individu S^{i-1} issu de la génération $i-1$, alors l'évaluation de l'objectif j de l'individu S^i est effectuée via les opérations suivantes :

$$e^i_{t,j} \leftarrow R_j(S^i_{t-1}, S^i_t)$$
$$e^i_{t+1,j} \leftarrow R_j(S^i_t, S^i_{t+1})$$

$$fit_j(S^i) = fit_j(S^{i-1}) - e^{i-1}_{t,j} - e^{i-1}_{t+1,j} + e^i_{t,j} + e^i_{t+1,j}$$

Suite à un croisement, les individus enfants obtenus sont évalués en utilisant une Δ-évaluation. Ce qui signifie que si la solution S est obtenue en croisant les solutions S^1 et S^2 et que les arcs E_i sont ceux hérités du parent S^i et les arcs E_n sont ceux qui sont propres à l'individu enfant S, l'évaluation de chaque objectif j de S se fait de la manière suivante :

$$fit_j(S) = \sum_{i \in E_1} e^1_{i,j} + \sum_{i \in E_2} e^2_{i,j} + \sum_{i \in E_n} R_j(S_{i-1}, S_i)$$

5.3 Opérateurs évolutionnaires

Les opérateurs évolutionnaires de MO-DYNAMOP se construisent exactement comme ceux de DYNAMOP (Voir chapitre 3). En effet, excepté que plusieurs valeurs sont associées à chaque arc, la représentation est inchangée et les opérateurs utilisés dans le cadre mono-objectif restent bien adaptés.

5.4 Opérateurs intelligents : hybridations

Comme dans la version mono-objectif, des opérateurs intelligents sont utilisés pour améliorer la convergence de l'algorithme. L'objectif de ces opérateurs est d'introduire des sous-chemins Pareto optimaux pour les sous-problèmes associés. Ces sous-chemins Pareto optimaux pouvant correspondre à des blocs de constructions de solutions Pareto optimale et les opérateurs évolutionnaires étant construits de sorte à conserver au mieux ces blocs, leur introduction dans la population peut impacter de manière très positive la convergence de l'algorithme. Pour construire de tels opérateurs des méthodes de résolution exactes sont utilisées sur des sous-problèmes dont la définition est basée sur la structure du graphe d'états. Les stratégies de construction d'opérateurs intelligents sont très similaires à celles utilisées dans le cas mono-objectif, si ce n'est que la méthode exacte utilisée sera plutôt une méthode multi-objectif. Nous expliquons ici en détails les principales stratégies permettant de définir des opérateurs intelligents de mutation et de croisement dans le cas multi-objectif.

Mutation intelligente

L'objectif de la mutation intelligente est de générer un ou plusieurs chemins de meilleure qualité (en terme de convergence) que le chemin d'origine. Pour ce faire, l'idée est la même qu'en mono-objectif, un graphe d'états restreint est défini autour du chemin d'origine et une méthode exacte est utilisée pour chercher les chemins Pareto optimaux dans ce graphe restreint. A priori, la méthode exacte la plus appropriée est la programmation dynamique multi-objectif. Néanmoins si le sous-problème associé au graphe restreint est résolvable plus rapidement par le biais d'une autre méthode rien n'empêche de l'utiliser. Plusieurs types de stratégies peuvent être mise en place et éventuellement combinées pour construire un graphe d'états restreint :

— Restriction en longueur : Cette stratégie a déjà été présentée dans le chapitre 3. Elle consiste à considérer le graphe restreint constitué de tous les sous-chemins liant deux états S_1 et S_2 du chemin d'origine. L'objectif étant de remplacer le sous-chemin d'origine par un sous-chemin Pareto optimal.

— Restriction en largeur : Cette stratégie a également été présentée dans le chapitre 3. Son principe est de définir un graphe d'états restreint en largeur autour du chemin à muter. Ceci est fait en fixant la valeur de certaines variables d'état ou bien en

limitant les valeurs qu'elles peuvent prendre à un petit intervalle centré autour de leurs valeurs d'origine.
— Restriction du nombre d'objectifs : Cette stratégie consiste à réduire le nombre d'objectifs en en combinant certains avec la méthode des sommes pondérées. Ceci permet de réduire la taille des front Pareto associés à chaque sous problème et donc peut réduire les temps de calcul considérablement. Si tous les objectifs sont combinés, on se ramène à un problème mono-objectif.

Si plusieures solutions Pareto équivalentes sont trouvées et que la population enfant n'est pas restreinte, toutes les solutions sont incluses dans la population enfant puis triées par le processus de sélection générant la prochaine génération. Si toutefois une borne est posée sur la taille de la génération enfant, on sélectionnera en priorité les solutions du front dominant fortement la solution d'origine s'il en existe, puis la stratégie est de sélectionner un ensemble de solutions bien réparties parmi le front Pareto généré.

Croisement intelligent

Les opérateurs de croisement intelligent de DYNAMOP peuvent tous être étendus au cas multi-objectifs. La différence réside simplement dans le fait, que pour le cas multi-objectifs, ils génèrent un ensemble de solutions Pareto équivalentes de meilleurs qualités que les solutions parentes. Comme pour les mutations intelligentes, des stratégies doivent être mises en place pour sélectionner les solutions de cet ensemble qui seront insérées dans la population enfant.

Ainsi les trois approches introduites aux chapitre 3 sont étendues au cas multi-objectif :
— Recombinaison de sous-politiques : En notant $(S_i^c)_{i=1...k}$, l'ensemble des états communs aux chemins à croiser, le principe de ce croisement est de sélectionner entre chaque couple d'états communs successifs S_i^c et S_{i+1}^c les sous chemins des chemins parents reliant ces deux états qui dominent au sens de Pareto. Les individus enfants sont alors l'ensemble des solutions Pareto équivalentes que l'on peut obtenir en combinant efficacement les sous-chemins ainsi sélectionnés.
— Construction intelligente de chemin de transition : S'il existe de nombreux chemins de transitions de taille minimale (au sens du nombre d'arcs) possibles lors du croisement, une stratégie peut être mise en place pour en sélectionner un parmi ceux qui sont les plus efficaces au sens de Pareto. Pour ce faire une méthode exacte peut être utilisée.
— Construction d'un graphe restreint à partir des chemins parents : L'idée est de définir un graphe d'état restreint prenant en compte les caractéristiques des deux parents et de chercher les chemins qui, dans ce graphe restreint, sont Pareto optimaux.

6 Conclusion

Dans ce chapitre nous avons proposé une extension de DYNAMOP pour l'optimisation multi-objectif : MO-DYNAMOP.

Les avantages potentiels de MO-DYNAMOP sont les mêmes que ceux de DYNAMOP à savoir que :
— Grâce à la représentation des solutions sous forme de séquence d'états, des opérateurs évolutionnaires prenant en compte les dépendances entre les décisions sont utilisés. De ce fait, les mutations sont conçues de manière à avoir de bonnes propriétés de localité et les croisement à avoir de bonnes propriétés d'héritabilité phénotypique ;

— la séparabilité de la fonction d'évaluation par rapport aux gènes permet de mettre en place des méthodes d'évaluation itérative et de Δ-évaluation qui peuvent permettre de réduire considérablement les temps de calcul ;
— La notion d'états empruntée à la programmation dynamique facilite la définition de sous-problèmes de petite taille et donc la proposition d'hybridation avec des méthodes exactes qui seront utilisées pour la résolution de ces sous-problèmes ;
— MO-DYNAMOP peut s'appliquer à une classe de problèmes assez large regroupant beaucoup de problèmes d'application industrielle. En fait, l'ensemble des problèmes auxquels MO-DYNAMOP peut s'appliquer est l'équivalent multi-objectif de l'ensemble des problèmes auxquels DYNAMOP peut s'appliquer.

Chapitre 8

Métaheuristiques avec représentation indirecte et système de décodage multi-objectif

L'étude faite dans ce chapitre constitue une autre contribution originale. En effet, dans ce chapitre, nous soulevons la question de la représentation indirecte avec un système de décodage basé sur une méthode multi-objectif. Cette problématique est issue de l'étude du problème d'affectation d'unités multi-objectif et les solutions que nous proposons sont testées avec succés sur ce problème. Néanmoins, nous pensons que l'idée de représentation indirecte peut être réutilisée. Nous avons donc pris soin d'introduire la problématique et nos solutions de manière aussi générique que possible.

Cette problèmatique du décodage a été présentée lors de plusieurs rencontres scientifiques :

- *EMO 2014 à Guimaraes, en Espagne [81] (publication LNCS) ;*
- *MIC 2015 à Agadir au Maroc [80] ;*
- *aux journées franciliennes de la recherche opérationnelle (JFRO) à Paris en juin 2015.*

1 Introduction

Ce chapitre a été motivé par une problématique de décodage rencontrée lorsque nous avons voulu étendre des métaheuristiques resolvant le problème d'affectation d'unités mono-objectif au multi-objectif.

Comme vu dans le chapitre 4, en mono-objectif, le problème d'affectation est généralement séparé en deux sous-problèmes : d'une part, le problème de planification des temps de marche et d'arrêt des unités de production et d'autre part le problème de distribution économique de la production entre les unités allumées. L'avantage de cette séparation réside dans le fait que le problème de distribution économique peut facilement être résolu avec une technique de λ-itération [150]. Les métaheuristiques proposées pour résoudre le problème d'affectation d'unités les plus performantes tirent toutes profit de cette séparation en utilisant une représentation indirecte avec un système de décodage basé sur la λ-itération [83, 84, 101]. Cette stratégie est illustrée par la figure 2.1 du chapitre 2. L'idée est que la métaheuristique manipule des vecteurs binaires représentant les plannings marche/arrêt des unités de production et la solution complète (la production exacte de chaque unité) est décodée avec la méthode de λ-itération.

Pour la version multi-objectif du problème, des métaheuristiques qui manipulent des solutions partielles fixant les valeurs des variables binaires (marche/arrêt) du problème ont également été proposées [110, 206]. Mais, le sous-problème de distribution étant aussi un problème multi-objectif, plusieurs solutions équivalentes pour le problème de distribu-

tion peuvent être associées à un unique planning des temps de marche et d'arrêt. Afin de contourner ce problème et de n'associer qu'une solution complète à chaque solution partielle, un vecteur poids est fixé par l'utilisateur en fonction de ses préférences et le sous-problème est résolu comme un problème mono-objectif défini par la méthode de la somme pondérée à partir du vecteur poids. Le désavantage d'une telle approche est bien sûr que le sous-problème ne soit pas traité comme un problème multi-objectif, ce qui entraine qu'une partie des solutions Pareto optimales du problème complet sont inaccessibles.

Dans le cadre de l'application de MO-DYNAMOP au problème d'affectation d'unités multi-objectif, nous serons également confrontés à ce problème de décodage. En effet, en multi-objectif, la définition de l'espace d'états ne change pas par rapport au cas mono-objectif, un chemin fixe donc, comme dans le cas mono-objectif, uniquement le planning des temps de marche et d'arrêt. Nous sommes donc particulièrement concernés par la difficulté du décodage et nous souhaitons mettre en place une technique prenant en compte l'ensemble des solutions Pareto optimales du sous-problème. De plus, nous pensons que la problèmatique de décodage pourrait concerner d'autres problèmes que le problème d'affectation d'unités et nous pensons donc qu'il s'agit d'une piste de recherche intéressante, d'autant plus que nous n'avons trouvé aucune étude à ce sujet.

Dans ce chapitre nous allons donc chercher à mettre en œuvre différentes techniques génériques de décodage permettant de ne pas réduire le sous problème à un problème mono-objectif. Ces techniques seront ensuite réutilisées dans le chapitre suivant pour appliquer MO-DYNAMOP au problème d'affectation d'unités.

Ce chapitre s'organisera comme ceci : Nous commencerons par formuler précisément la problématique du décodage en multi-objectif, en précisant le type de problèmes d'optimisation concernés et en expliquant en détails les difficultés. Nous proposerons ensuite deux méthodes de décodage que nous appliquerons sur le problème d'affectation d'unités en utilisant un algorithme génétique basique. Puis nous ferons une étude de comparaison des méthodes proposées entre elles et avec la méthode utilisée dans la littérature où le sous-problème de distribution est ramené à un problème mono-objectif. Enfin, nous conclurons sur le potentiel des différentes propositions.

2 Formalisation du problème

Dans cette section nous allons formaliser les problématiques posées par une représentation indirecte dans le cadre d'un problème multi-objectif.

2.1 Représentation indirecte dans le cas mono-objectif

Dans le cas mono-objectif, les problèmes concernés par la représentation indirecte sont ceux à deux niveaux hiérarchiques, c'est-à-dire ceux modélisables sous la forme :

$$\begin{array}{c} \min\limits_{x,y} f(x,y) \\ x \in X \\ y \in Y(x) \end{array} \tag{8.1}$$

tels que pour x fixé le sous problème Π_x :

$$\begin{array}{c} \min\limits_{y} f(x,y) \\ y \in Y(x) \end{array} \tag{8.2}$$

soit résolvable en un temps de calcul raisonnable. En effet, dans ce genre de cas, il est possible de proposer des métaheuristiques dont l'objectif est de fixer la valeur de la variable x,

de sorte que la solution du sous-problème associé (8.2) soit la meilleure possible. Ainsi les métaheuristiques basées sur une modélisation à deux niveaux, parcourent l'espace de recherche restreint X et utilisent un système de décodage pour obtenir une solution complète (x, y) à partir d'une solution partielle x et ainsi pouvoir évaluer x. La figure 8.1 illustre ce procédé. Généralement les métaheurisques utilisant une représentation indirecte sont très

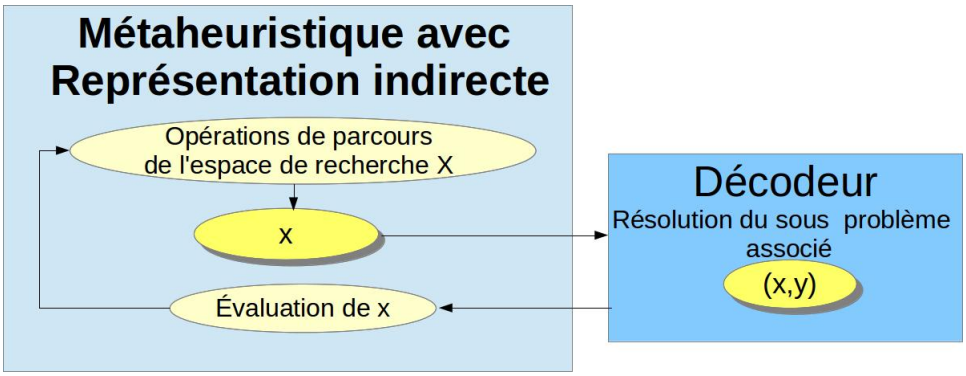

FIGURE 8.1 – Représentation indirecte pour les problèmes mono-objectifs.

efficaces car ce procédé permet de réduire considérablement l'espace de recherche.

2.2 Problèmes multi-objectifs à deux niveaux hiérarchiques

Certains problèmes multi-objectifs peuvent également êtres modélisés comme des problèmes à deux niveaux hiérarchiques :

$$\begin{gathered} \min_{x,y}\{f_1(x,y), f_2(x,y)...f_m(x,y)\} \\ x \in X \\ y \in Y(x) \end{gathered} \tag{8.3}$$

tels que pour x fixé le sous problème Π_x :

$$\begin{gathered} \min_{y}\{f_1(x,y), f_2(x,y)...f_m(x,y)\} \\ y \in Y(x) \end{gathered} \tag{8.4}$$

soit solvable en un temps raisonnable par une méthode exacte ou une heuristique de qualité.

2.3 Difficultés de la représentation indirecte en multi-objectif

La représentation indirecte donnant d'excellents résultats en mono-objectif, il est tentant de la généraliser au cas multi-objectif. Cependant en multi-objectif, si la variable y a une influence sur plusieurs objectifs, le sous problème (8.4) est un problème multi-objectif. Ainsi pour une valeur de x fixée il existe un ensemble de solutions Pareto équivalentes, noté $\mathcal{P}_x$, pour le problème Π_x, (8.4). De ce fait, pour mettre en place une métaheuristique avec représentation indirecte deux approches s'offrent à nous :

— **Approche 1 (utilisation d'un décodeur mono-objectif) :** L'idée de cette approche est d'associer à chaque élément x manipulé par la métaheuristique multi-objectif une unique solution $y \in \mathcal{P}_x$, mais dans ce cas une stratégie doit être mise en place pour sélectionner cet élément parmi l'ensemble des solutions Pareto optimales, $\mathcal{P}_x$, du problème Π_x. La figure 8.2 permet de visualiser cette approche et la problématique qu'elle implique.

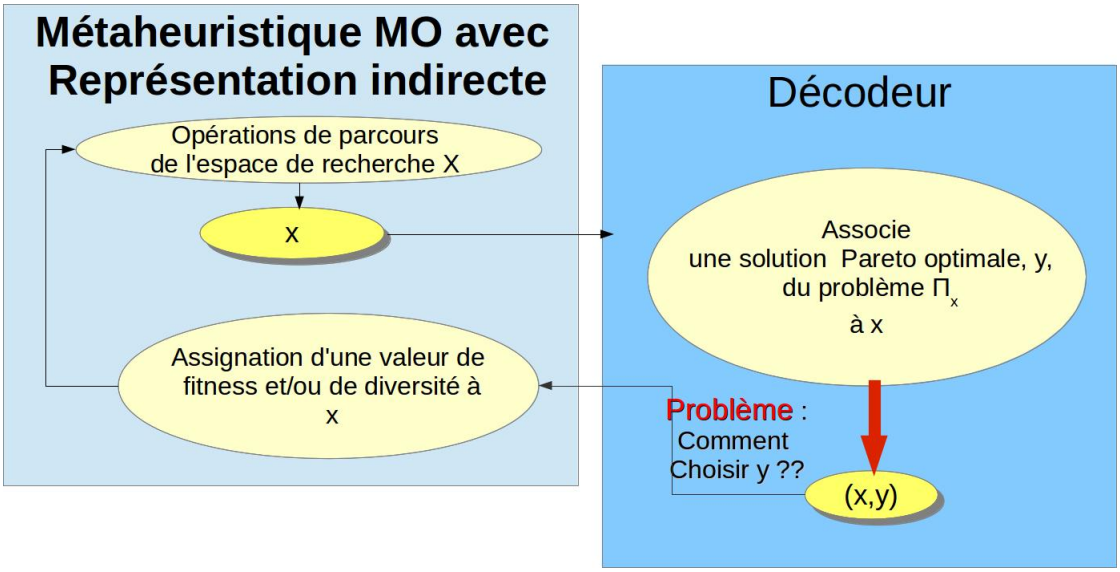

FIGURE 8.2 – Représentation indirecte pour les problèmes multi-objectifs version 1.

— **Approche 2 (utilisation d'un décodeur multi-objectif) :** L'idée de cette approche est d'associer à chaque élément x manipulé par la métaheuristique multi-objectif l'ensemble des solutions Pareto optimales, $\mathcal{P}_x$, du sous problème associé Π_x. Mais dans ce cas, il faut mettre en place une stratégie pour assigner à x une valeur de fitness et/ou une valeur de diversification prenant en considération l'ensemble des solutions décodées $\{(x, y), y \in \mathcal{P}_x\}$ qui lui sont associées. La figure 8.3 permet

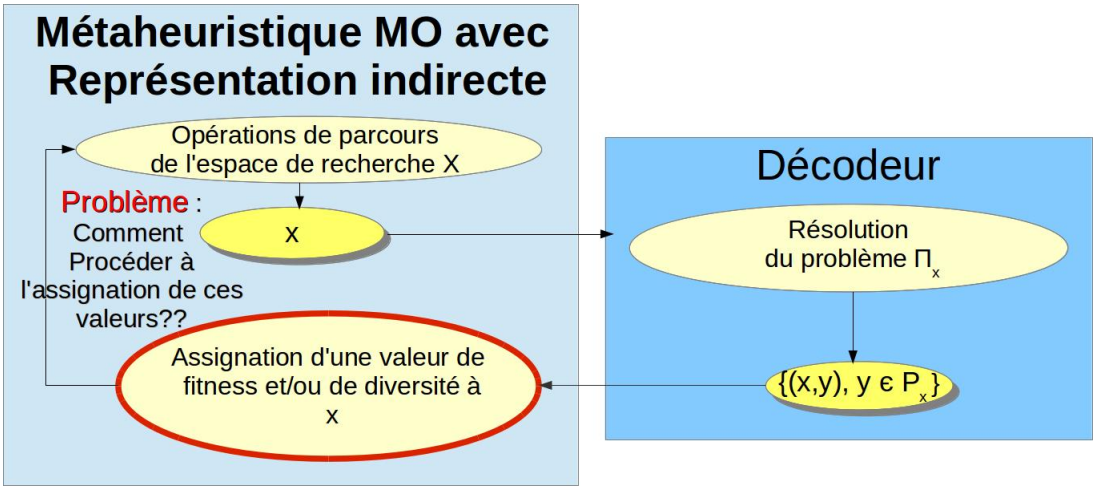

FIGURE 8.3 – Représentation indirecte pour les problèmes multi-objectifs version 2.

de visualiser cette approche et la problématique qu'elle implique.

3 Stratégies de décodage en multi-objectif

Le passage d'une métaheuristique mono-objectif avec représentation indirecte à une métaheuristique multi-objectif avec cette même représentation pose donc certaines problématiques. La première approche a déjà été utilisée dans la littérature scientifique dans

le cadre du problème d'affectation d'unités. La deuxième approche, n'a en revanche, à notre connaissance, jamais été abordée. Dans cette section, nous allons présenter les solutions existantes exploitant la première approche, puis proposer de nouvelles solutions. Les diverses solutions seront comparées dans la section suivante.

3.1 Décodeur mono-objectif

Nous présentons ici des stratégies permettant d'associer une unique solution Pareto optimale $y \in \Pi_x$, à la solution partielle x fournie par la métaheuristique.

Méthode de scalarisation invariante

Cette méthode consiste à transformer le sous problème Π_x en un problème mono-objectif par la méthode de la somme pondérée :

$$x \xrightarrow{\min_y \sum_{i=1}^{m} \alpha_i f_i(x,y)} (x, y)$$

Les coefficients de pondération α_i sont choisis a priori en fonction de critères de préférence.

L'inconvénient de cette méthode est qu'une partie du front Pareto reste inaccessible. Avec cette approche, il n'est en effet même pas possible d'assurer que toutes les solutions supportées soient atteignables. Les algorithmes proposés dans [110, 206] utilisent cette méthode dans le cadre du problème d'affectation d'unités de production.

Méthode de scalarisation embarquée

Cette méthode consiste à attacher à chaque solution x un vecteur poids α de sorte que la solution utilise ce vecteur poids pour scalariser les objectifs lors de l'étape de décodage. Ainsi, la métaheuristique manipule des solutions de la forme :

$$(x, \alpha),$$

avec $x \in X$ et $\alpha = (\alpha_i)_{i=1\ldots m}$ tel que pour tout i, $\alpha_i \in [0, 1]$ et $\sum_{i=1}^{m} \alpha_i = 1$.

La figure 8.4 permet de visualiser cette solution.

Dans le cadre des métaheuristiques multi-objectifs basées sur la recherche locale, cette approche a déjà été proposée [123]. Pour ce type de métaheuristique, cette approche est assez naturelle car elle est, en fait, même utilisée en dehors de la problématique de décodage, ceci afin de guider le choix de la solution à sélectionner parmi celles qui sont Pareto équivalentes dans le voisinage [35]. L'idée est d'associer à chaque itération un vecteur poids α à chaque solution en fonction de son voisinage. L'objectif étant de se servir de ce vecteur poids pour assurer la diversité du front obtenu. Par exemple la stratégie peut être la suivante :

- Soit x' la solution de la population la plus proche de x qui n'est pas dominée par x, alors les valeurs des coefficients α_i tels que $f_i(x) \leq f_i(x')$ sont légèrement augmentés et ceux des autres coefficients légèrement diminués ;
- s'il n'existe pas un tel élément x', α est modifié aléatoirement.

Dans le cadre des métaheuristiques multi-objectifs évolutionnaires, nous n'avons trouvé aucun exemple d'approche similaire dans la littérature. Pour ce type de métaheuristiques,

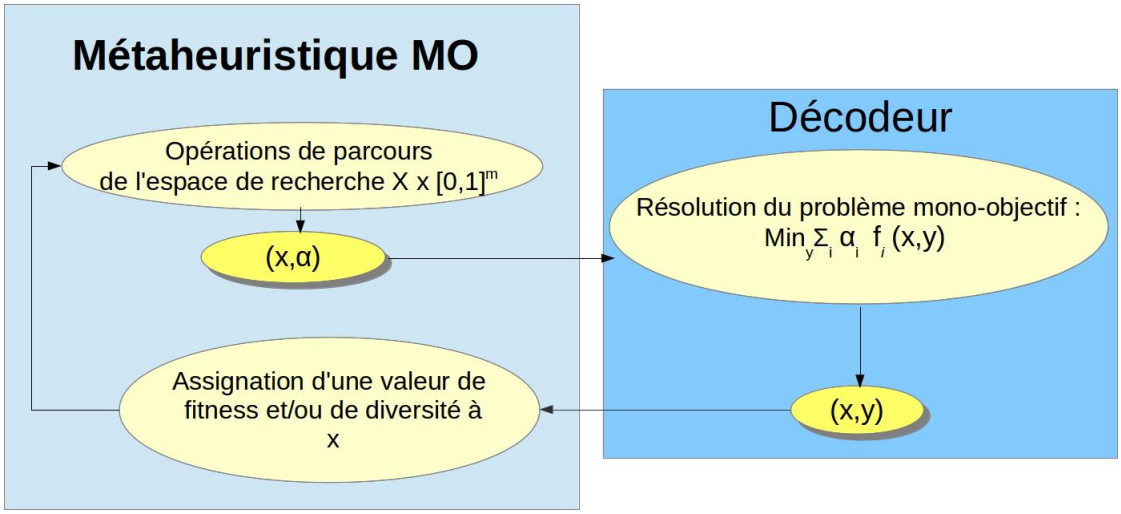

FIGURE 8.4 – Représentation indirecte pour les problèmes multi-objectifs avec méthode de décodage par scalarisation embarquée.

nous proposons de traiter α via un opérateur de mutation. Cet opérateur modifie α aléatoirement sur la base d'une loi normale centrée réduite autour de la valeur initiale de celui-ci.

3.2 Décodeur multi-objectif

Nous allons maintenant présenter des méthodes de décodage utilisant la seconde approche, c'est-à-dire associant à une solution partielle x, l'ensemble des solutions $\{(x, y), y \in \mathcal{P}_x\}$, où $\mathcal{P}_x$ est l'ensemble des solutions Pareto optimales du problème Π_x.

Il nous faut donc proposer une stratégie pour assigner une valeur de fitness et de diversité à une solution partielle qui est décodée par un ensemble de solutions complètes. Les approches classiques d'assignation de fitness et de diversité ont été introduites au chapitre 7 section 3.

Nous allons proposer deux méthodes d'adaptation des processus d'assignation en fonction de la nature des mesures sur lesquels ils se basent.

Adaptation des processus d'assignations basés sur des indicateurs unaires

Un indicateur unitaire I_u permet de mesurer la qualité ou la diversité d'un sous-ensemble A de l'espace objectif. Dans le cas classique où une représentation directe est utilisée si la fitness ou l'apport à la diversité d'une solution x est évaluée avec un opérateur unaire I_u, celui-ci est appliqué au singleton $\{f(x)\}$, avec $f(x) = (f_1(x), f_2(x), ..., f_m(x))$.

Dans le cas d'une représentation indirecte nous adaptons le processus d'assignation en appliquant I_u sur l'ensemble $f(\{(x, y), y \in \mathcal{P}_x\})$.

Adaptation des autres processus d'assignation

Dans cette partie nous allons expliquer comment les autres processus d'assignation basés sur la relation de Pareto dominance et sur une mesure de distance ou sur l'utilisation d'un indicateur binaire, sont adaptés pour prendre en compte l'ensemble $\{(x, y), y \in \mathcal{P}_x\}$.

Nous distinguons les processus d'assignation de fitness et de diversité. Par la suite nous noterons P la population de solutions partielles x et P_c la population de solutions complètes obtenue en décodant tous les éléments de P.

— **Assignation d'une valeur de fitness :** Dans le cas classique, où une représentation directe est utilisée, une valeur de fitness $fit(x)$ est associée à une solution x par le biais d'une fonction $m_f(x, P - \{x\})$ comparant x aux autres éléments de la population P. Dans le cas d'une représentation indirecte, le processus d'assignation de fitness est adapté comme ceci :

$$fit(x) = \underset{y \in \mathcal{P}_x}{opt}\ fit((x, y)) \tag{8.5}$$

avec :

$$fit((x, y)) = m_f((x, y), P_c - \{(x, z), z \in \mathcal{P}_x\}) \tag{8.6}$$

C'est à dire que la fitness de x est la meilleure fitness parmi celles attribuées aux solutions complètes associées (x, y) en ne les comparant qu'aux solutions complètes issues de solutions partielles x' différentes de x ($x' \in P - \{x\}$).

— **Assignation d'une valeur de diversité :** Dans le cas classique, où une représentation directe est utilisée, une valeur de diversité $div(x)$ est associée à une solution x par le biais d'une fonction $m_d(x, P - \{x\})$ comparant x aux autres éléments de la population P. Dans le cas d'une représentation indirecte, le processus d'assignation de diversité est adapté comme ceci :

$$div(x) = \underset{y \in \mathcal{P}_x, fit((x,y)) = fit(x)}{opt}\ div((x, y)) \tag{8.7}$$

avec :

$$div((x, y)) = m_d((x, y), P_c - \{(x, z), z \in \mathcal{P}_x\}) \tag{8.8}$$

Ainsi la valeur de diversité de x correspond à la meilleure diversité attribuée à un des éléments (x, y) ($y \in \mathcal{P}_x$) parmi ceux ayant la meilleure valeur de fitness sans prendre en compte les autres solutions complètes (x, z) issues de x.

Exemples

Afin d'illustrer les méthode d'adaptation des processus d'assignation proposés ci-dessus, nous allons maintenant donner quelques exemples concrets :

— **Adaptation de l'assignation d'une valeur de fitness avec l'indicateur unaire hypervolume :**

  — Dans la cas classique où une représentation directe est utilisée, l'attribution d'une valeur de fitness à une solution x de la population P avec l'indicateur unaire hypervolume se fait comme ceci :

$$fit(x) = hypervolume(P) - hypervolume(P - \{x\})$$

  Cet indicateur a été présenté en détail au chapitre 6 section 3.

  — Dans le cadre d'une représentation indirecte avec décodeur multi-objectif, ce processus d'assignation de fitness s'adapte en substituant l'ensemble $\{(x, y), y \in \mathcal{P}_x\}$ au singleton $\{x\}$:

$$fit(x) = hypervolume(P_c) - hypervolume(P_c - \{(x, y), y \in \mathcal{P}_x\})$$

  Ainsi la fitness de x représente l'hypervolume perdu si on supprime de l'espace objectif l'ensemble $f(\{(x, y), y \in \mathcal{P}_x\})$.

  La figure 8.5, illustre le processus d'assignation de fitness avec l'indicateur unaire de différence d'hypervolumes pour une population de quatre solutions. Avec le

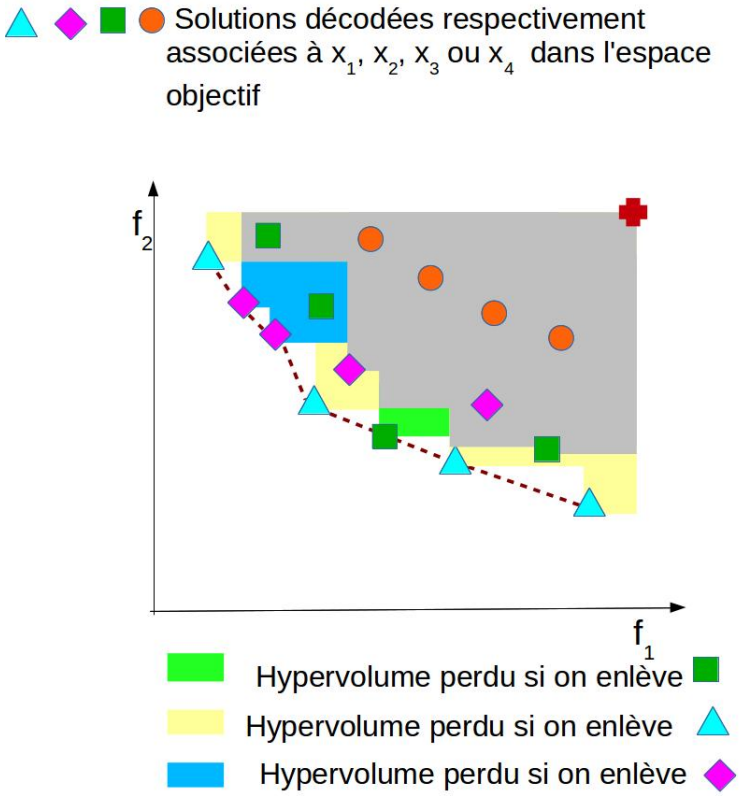

FIGURE 8.5 – Opérateur unaire différence d'hypervolumes appliqué à une population de quatre solutions avec décodeur multi-objectif.

processus d'assignation de fitness proposé, la valeur de fitness attribuée à la solution x_1 correspond à l'aire de l'espace coloré en jaune, celle de la solution x_2 correspond à l'aire de l'espace coloré en bleu et celle de la solution x_3 à l'aire de l'espace coloré en vert. Quant à la fitness de l'individu x_4, elle est nulle car sa suppression n'entraîne pas de perte d'hypervolume.

— **Adaptation du processus d'assignation de fitness par la méthode de rang de dominance (NSGA-II) :** Dans ce cas la fonction $m_f(x, P - \{x\})$ est la fonction qui renvoie le rang de dominance de x dans l'ensemble P. Dans le cas d'un décodage multiple, les solutions (x, y) associées à x sont toutes Pareto équivalentes, ainsi le rang associé à (x, y) dans l'ensemble $P_c - \{(x, y), y \in \mathcal{P}_x\}$ est le même que celui qui lui est associé dans l'ensemble P_c, et le processus d'assignation s'adapte comme ceci :

$$fit(x) = \min_{y \in \mathcal{P}_x} rang((x, y))$$

La figure 8.6 illustre ce processus dans le cadre d'un problème avec deux objectifs à minimiser. Sur cette figure, trois solutions génotypiques sont représentées dans l'espace objectif, u_1, u_2 et u_3. u_1 est décodée par l'ensemble des solutions phénotypiques illustrées par des carrés, u_2 par l'ensemble des solutions phénotypiques illustrées par des triangles et u_3 par l'ensemble des solutions phénotypiques illustrées par des ronds. La valeur de fitness associée à chacune de ces solutions génotypiques correspond au meilleur rang parmi ceux des solutions phénotypiques obtenues en les décodant. Ainsi la valeur de fitness associée à u_1 et à u_3 est 1 et celle associée à u_2 est 2.

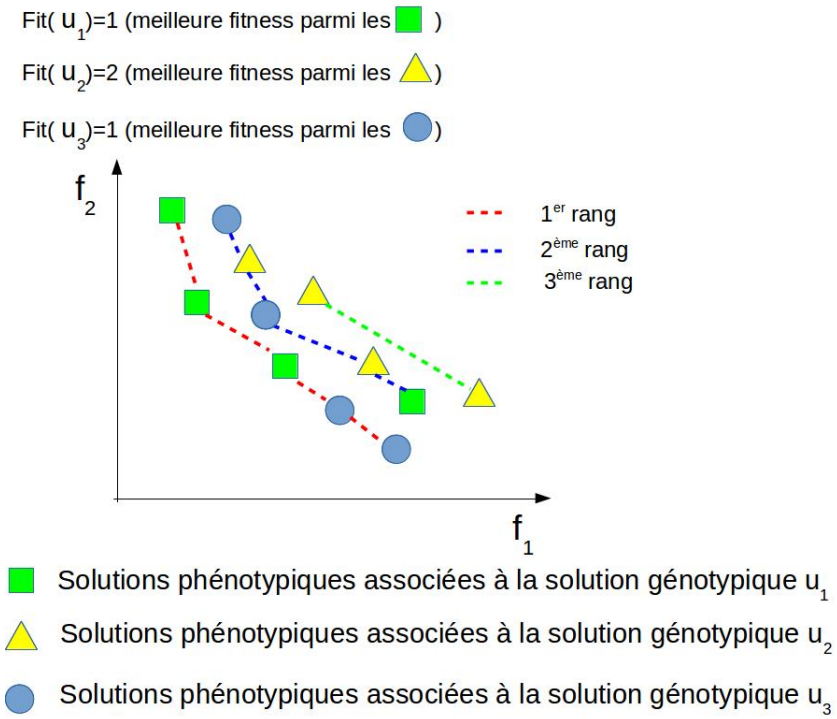

FIGURE 8.6 – Adaptation de l'assignation de fitness avec rang de dominance au métaheuristiques utilisant une représentation indirecte avec décodeur multi-objectif.

— **Adaptation du processus d'assignation de fitness d'IBEA :** Dans IBEA la fitness associée à une solution x est donnée par la fonction :

$$fit(x) = \sum_{y \in P - \{x\}} (-e^{-I(y,x)/\kappa}) \tag{8.9}$$

où κ est un réel à fixer, et I est un indicateur binaire. En se conformant à la formule 8.5, son adaptation pour des métaheuristiques utilisant une représentation indirecte avec décodeur multi-objectif est :

$$fit(x) = \max_{y \in \mathcal{P}_x} \sum_{z \in P_c - \{(x,y), y \in \mathcal{P}_x\}} (-e^{-I(z,(x,y))/\kappa}) \tag{8.10}$$

— **Adaptation du processus d'assignation de diversité de NSGA-II :** Dans NSGA-II la valeur de diversité associée à la solution x correspond à sa distance de *crowding* (d_c) avec les éléments de la population ayant la même fitness que x. Ainsi $m_d(x, P) = d_c(x, P_{fit(x)})$, où $P_{fit(x)}$ est l'ensemble des éléments de P dont le rang (la fitness) est celui de x, et le processus de diversité s'adapte pour le décodage multiple comme ceci :

$$div(x) = \max_{y \in \mathcal{P}_x, fit((x,y)) = fit(x)} d_c((x,y), (P_c - \{(x,y), y \in \mathcal{P}_x\})_{fit(x)}) \tag{8.11}$$

Les figures 8.7 et 8.8 illustrent le procédé d'assignation de diversité dans le cas de NSGA-II. Pour calculer la diversité de l'individu dont la représentation phénotypique correspond au front de carrés, nous considérons chaque carré dont le rang est celui attribué à l'individu, c'est-à-dire le meilleur rang parmi ceux attribués aux

carrés qui est 1. Une valeur de diversité est attribuée à chacun de ces carrés en calculant sa distance de crowding avec les autres éléments du front mais en considérant que les autres carrés ne font pas parti de la population. Chacune des images illustre le calcul de la valeur de diversité du carré coloré en jaune en évaluant sa distance aux solutions phénotypiques colorés en rouge. La meilleure valeur de diversité ainsi trouvée est celle attribuée à l'individu génotypique, ici $d3$.

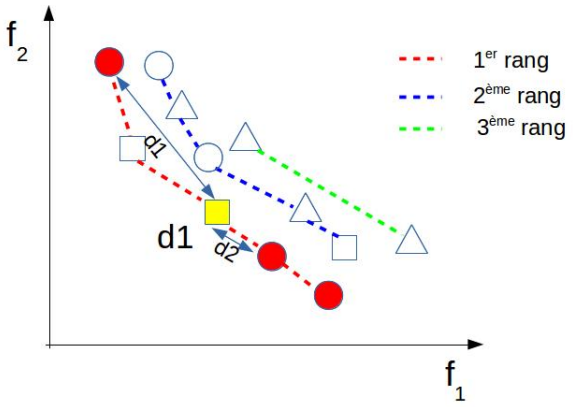

FIGURE 8.7 – Attribution d'une diversité à l'individu dont la représentation phénotypique est le front constitué de carrés : étape 1. Distance Max=$d1$.

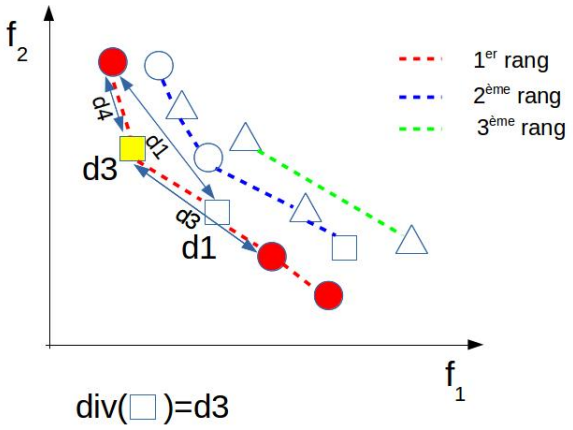

FIGURE 8.8 – Attribution d'une diversité à l'individu dont la représentation phénotypique est le front constitué de carrés : étape 2. Distance Max=$d3$.

4 Application au problème d'affectation d'unités

Dans cette section nous allons appliquer les méthodes de décodage proposées dans la section précédente à un algorithme génétique basique traitant le problème d'affectation d'unités de production.

4.1 Modélisation à deux niveaux hiérarchiques du problème d'affectation d'unités multi-objectif

Comme indiqué précédemment, le problème d'affectation d'unités mono-objectif est souvent modélisé comme un problème à deux niveaux hiérarchiques. Le premier niveau correspond au problème de planification des temps de marche et d'arrêt des unités et le deuxième niveau est le problème de distribution économique de la production entre les unités allumées pour un planning des temps de marche et d'arrêt donné.

Cette séparation reste intéressante dans le cas multi-objectif, en effet le problème de distribution associé un planning u, Π_u, se formule alors comme ceci :

$$\min_p(\sum_{t=1}^{T}\sum_{i=1}^{N} a_{0,i} + a_{1,i}p_{i,t} + a_{2,i}p_{i,t}^2, \sum_{t=1}^{T}\sum_{i=1}^{N} b_{0,i} + b_{1,i}p_{i,t} + b_{2,i}p_{i,t}^2)$$

tel que :

$$\begin{array}{cc} \sum_{i=1}^{n} p_{i,t} = D_t & \forall t \\ p_{i,min} \leq p_{i,t} \leq p_{i,max} & \forall t, i, u_{i,t} = 1 \\ p_{i,t} = 0 & \forall t, i, u_{i,t} = 0 \end{array} \tag{8.12}$$

Or il s'agit d'un problème d'optimisation bi-objectif continu dont les fonctions objectifs sont toutes deux convexes et dont l'ensemble de définition des solutions réalisables est convexe. Il a été montré que pour ce type de problème toutes les solutions sont supportées [45]. C'est à dire que le front Pareto associé au sous-problème est défini par :

$$\{f_{1,\alpha}, f_{2,\alpha} | \alpha \in [0,1]\},$$

où $f_{2,\alpha} = f_2(p_\alpha^*)$ et $f_{1,\alpha} = f_1(p_\alpha^*)$,avec p_α^* solution du problème de distribution $\Pi_{u,\alpha}$ défini par :

$$p_\alpha^* = arg(\min \sum_{t=1}^{T}\sum_{i=1}^{N} \alpha(a_{0,i} + a_{1,i}p_{i,t} + a_{2,i}p_{i,t}^2) + (1-\alpha)(b_{0,i} + b_{1,i}p_{i,t} + b_{2,i}p_{i,t}^2)) \tag{8.13}$$

tel que les contraintes (8.12) soient respectées.

Il est donc possible d'obtenir une bonne approximation de ce front en résolvant $\Pi_{u,\alpha}$ pour différentes valeurs de α avec la méthode de λ-itération.

4.2 Un algorithme génétique basique

Afin de tester les techniques de décodage proposées précédemment, nous allons utiliser un algorithme génétique simple basé sur NSGA-II qui traite le problème d'affectation d'unités multi-objectif en utilisant une représentation indirecte. Cet algorithme est en fait une adaptation au cas multi-objectif de l'algorithme BGA présenté au chapitre 4 section 4.3. Nous l'appellerons MOBGA.

Représentation

L'algorithme manipule une population de solutions partielles associant à chaque unité un planning de temps de marche et d'arrêt. Une solution est représentée par un vecteur binaire u de taille $N \times T$, tel que les T premiers éléments donnent le planning de l'unité 1, les T suivants donnent le planning de l'unité 2 et cetera. La figure 8.9 illustre la représentation génotypique d'une solution. Cette solution doit ensuite être décodée pour connaître les valeurs des variables $p_{i,t}$ et les valeurs des objectifs (solution phénotypique).

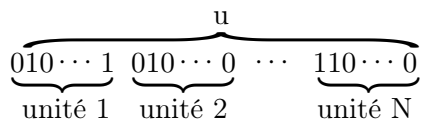

FIGURE 8.9 – Représentation génotypique d'une solution.

Opérateur évolutionnaires

Les opérateurs de mutation, de croisement et de correction de BGA sont réutilisés tels quels excepté le croisement 2-points intelligent qui est adapté pour comparer les parents en fonction de la valeur de fitness (c'est à dire le rang) qui leur est attribuée.

Fonction objectif

Les valeurs objectifs sont associées aux solutions complètes (u, p) décodées à partir des solutions partielles u. Ces valeurs correspondent à la fonction objectif f_1, qui représente le coût de production, et à la fonction f_2 qui mesure les émissions en SO_2 et CO_2. En outre des pénalités doivent être ajoutées pour exclure les solutions non réalisables. Dès lors les valeurs objectifs associées à une solution décodée sont :

$$\begin{cases} obj_1 &= f_1 + c_p \sum_{t=1}^{T}(\sum_{i=1}^{N} p_{i,t} - D_t) + c_p((T^{off}_{i,min} - T^{off}_{i,t-1})(u_{i,t} - u_{i,t-1}) + \\ & (T^{on}_{i,min} - T^{on}_{i,t-1})(u_{i,t-1} - u_{i,t})), \\ obj_2 &= f_2 + c_p \sum_{t=1}^{T}(\sum_{i=1}^{N} p_{i,t} - D_t) + c_p((T^{off}_{i,min} - T^{off}_{i,t-1})(u_{i,t} - u_{i,t-1}) + \\ & (T^{on}_{i,min} - T^{on}_{i,t-1})(u_{i,t-1} - u_{i,t})), \end{cases} \tag{8.14}$$

avec c_p une constante positive de grande taille.

4.3 Application des techniques de décodages

Nous allons ici, présenter en détail l'application des trois types de décodeurs à l'algorithme présenté ci-dessus.

Décodage par la méthode de scalarisation invariante

La méthode de scalarisation invariante est appliquée en utilisant la pondération $(1, 1)$ pour résoudre le sous-problème de distribution optimale. Ainsi le décodage d'une solution partielle u se fait en résolvant le problème $\Pi_{u,\frac{1}{2}}$. La figure 8.10 illustre cette approche de décodage.

Décodage par la méthode de scalarisation embarquée

Pour cette approche, un entier, $\alpha \in [\![0, 100]\!]$, est ajouté à la représentation sous sa forme binaire. Une solution u est alors décodée en résolvant le problème $\Pi_{u,\frac{\alpha}{100}}$. La figure 8.11 illustre le décodage et la représentation. En outre les opérateurs de croisement et de mutation sont adaptés afin que leur impact sur la valeur α soit significatif :

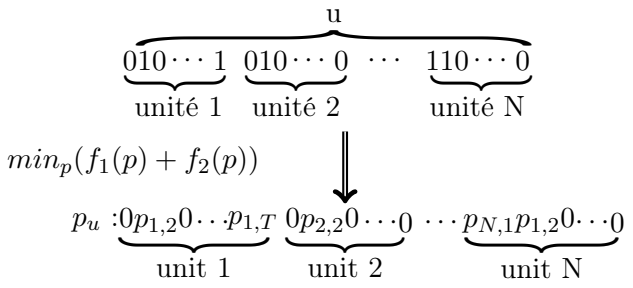

FIGURE 8.10 – Décodage par la méthode de scalarisation invariante.

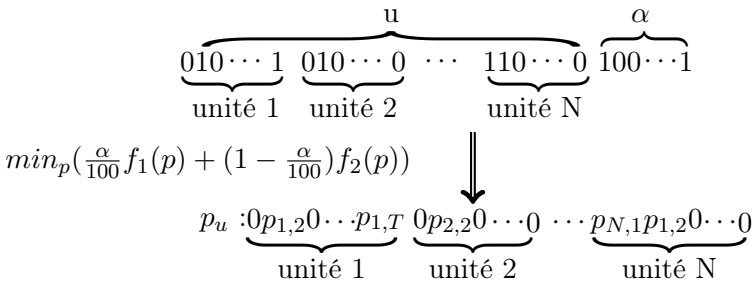

FIGURE 8.11 – Décodage par la méthode de scalarisation embarquée.

— **Croisements :** Les croisements 1-point et 2-points ne s'appliquent que sur la partie du génotype correspondant au vecteur u. La valeur α associée à l'individu enfant est obtenue par une combinaison affine des valeurs α_1 et α_2 associées aux individus parents :

$$\alpha = a \times \alpha_1 + (1 - a) \times \alpha_2 \,;$$

où a est choisi aléatoirement en suivant une loi uniforme dans l'intervalle $[0, 1]$.

— **Mutations :** Les mutations décrites dans la partie précédentes ne s'appliquent que sur la partie du génotype correspondant à u. α est modifié par le biais d'une nouvelle mutation qui remplace la valeur de α par une nouvelle valeur choisie aléatoirement entre 0 et 100 en suivant une lois de distribution normale centrée réduite autour de la valeur initiale de α.

Il est à noter, qu'étant donné que les solutions du sous-problème de distribution sont toutes supportées, cette stratégie permet de joindre n'importe quelle solution Pareto optimale du problème complet.

Décodage avec décodeur multi-objectif

Dans cette approche, une solution génotypique u est décodée par un ensemble de solutions phénotypiques Pareto équivalentes $\mathcal{P}_u$. Cet ensemble de solutions phénotypiques s'obtient en résolvant le problème Π_u. La figure 8.12 aide à comprendre comment les solutions génotypiques sont représentées dans l'espace objectif. La ligne rouge continue représente le front Pareto optimal associé au problème complet d'affectation d'unités. Ce front peut être non convexe. Chacun des fronts convexes, composé de ronds, de carrés ou de losanges est obtenu en décodant une unique solution génotypique.

Plus précisément, l'ensemble des solutions décodées, $\mathcal{P}_u$, associé à la solution génoty-

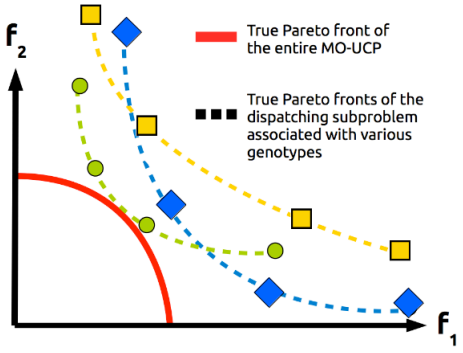

FIGURE 8.12 – Représentation phénotypique des solutions génotypiques avec décodage multiple.

pique u est défini par :

$$\{p_u^{\frac{k}{n_\lambda}}, k = 0...n_\lambda\}, \tag{8.15}$$

où n_λ est un paramètre à fixer et $p_u^{\frac{k}{n_\lambda}}$ est le vecteur à valeurs réelles solution du problème $\Pi_{u,\frac{k}{n_\lambda}}$. La figure 8.13 illustre le décodage d'une solution u dans le cas où n_λ est égal à 3.

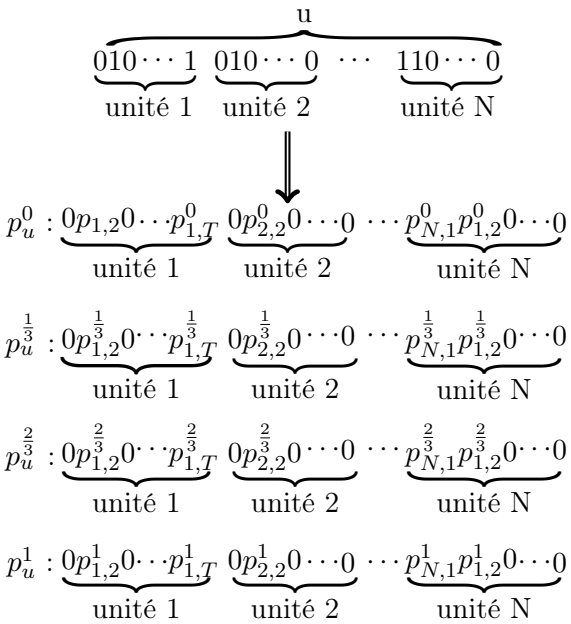

FIGURE 8.13 – Décodage d'une solution génotypique u avec $n_\lambda = 3$

Comme chaque solution génotypique est décodée par un ensemble solutions phénotypiques, les stratégies d'assignation d'une valeur de fitness et de diversité à un génotype doivent être adaptées. Les stratégies d'adaptation des processus d'assignation de NSGA-II qui ont été présentées dans la section 3.2 sont donc utilisées.

5 Analyse comparative des solutions proposées sur le problème d'affectation d'unités

Dans cette section nous allons présenter une analyse comparative des trois approches de décodage présentées ci-dessus. Nous présentons tout d'abord le protocole expérimental puis les résultats obtenus.

5.1 Protocole expérimental

Toutes les méthodes présentées ont été implémentées à l'aide du logiciel Paradiseo [111]. Paradiseo est un logiciel open-source qui fournit des *framework* permettant d'implémenter facilement des métaheuristiques.

Instances

Les instances utilisées sont issues de la thèse de Ana Viana [123]. Ce sont en fait les mêmes instances que celles que nous avons utilisées pour le cas mono-objectif et qui sont décrites par les tableaux 4.1 et 4.2 du chapitre 4, excepté que des coefficients supplémentaires b_i sont donnés pour décrire la fonction d'émissions en SO_2 et CO_2. Ces coefficients sont décrits dans le tableau 8.1.

unit	**unit1**	**unit2**	**unit3**	**unit4**	**unit5**	**unit6**	**unit7**	**unit8**	**unit9**	**unit10**
b_1	712	570	700	860	350	370	480	660	665	670
b_2	12,9	10,26	10,60	15,50	7,70	9,26	3,74	5,92	7,27	7,79
$b_3(\times 10^{-4})$	4	3	22	11	10	22	30	40	13	23

Tableau 8.1 – Coefficient de la fonction de mesure des émissions de gaz à effet de serre.

Mesure de performance

Les différentes mesures pouvant être choisies pour comparer les performances de méthodes de résolution de problèmes d'optimisation multi-objectif ont été décrites dans la section 3.2 du chapitre 6. Pour notre étude nous avons choisi d'utiliser les versions unaires de l'ϵ-*indicateur* et l'indicateur d'*hypervolume-différence*, car ils sont complémentaires. Ces indicateurs sont utilisés pour le paramétrage des algorithmes et pour les tests de comparaison entre les méthodologies de décodage.

Paramétrage

Une analyse de sensibilité est effectuée pour fixer la valeur des paramètres associés à chacune des méthodologies pour les différentes instances. Cette analyse est effectuée grâce au package *Irace* [113] du logiciel de statistique *R*. Les paramètres fixés via cette analyse sont :

- La probabilité pour un individu de subir la mutation $1 - bit - flip$: valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- La probabilité pour un individu de subir la *window-mutation* : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- La probabilité pour un individu de subir un croisement 1-point : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.

— La probabilité pour un individu de subir un croisement 2-points : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.

Pour le cas de la méthode basée sur le système de décodage par scalarisation embarquée, la probabilité d'une mutation sur α est également un paramètre à fixer, celui-ci est testé par *Irace* dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}. Pour le cas du décodeur multiple, la valeur de n_α est également fixée en utilisant *Irace*, sa valeur est testée dans l'intervalle $[\![2, 10]\!]$.

La taille de la population est fixée à 100 et le critère d'arrêt utilisé lors du paramétrage est un temps d'exécution maximal de $15 \times N$ secondes (avec N le nombre d'unités).

Mise en œuvre des tests expérimentaux

Afin de comparer les différentes méthodes, chacune d'entre elle est exécutée 20 fois sur chaque instance en utilisant 20 "graines" fixes. C'est-à-dire, que les variables aléatoires des algorithmes vont réagir de la même façon pour les différentes méthodes et donc, en l'occurrence, les populations initiales sont les mêmes. De ce fait, on peut dire que les tests sont appariés.

C'est pourquoi les tests statistiques que nous effectuons sont des tests de Friedman, qui permettent de déterminer si une différence statistiquement significative existe entre les résultats fournis par les méthodes, suivi, si la différence est significative, de tests post-hocs qui permettent de comparer les méthodes deux à deux afin de déterminer si l'une est meilleure que l'autre. Dans tous les cas une p-value de 0,01 est utilisée.

Ces tests sont réalisés grâce au logiciel Pisa [23], qui permet de comparer statistiquement des échantillons de solutions de problème multi-objectif par rapport à un indicateur de performance donné.

Le critère d'arrêt utilisé est un temps maximal. Pour choisir ce temps limite, nous avons, dans un premier temps, lancé des exécutions de nos méthodes sur chaque instance en utilisant comme critère d'arrêt 100 générations sans amélioration, dans le sens où l'hypervolume associé à la population ne grandit plus. Puis, nous avons pris comme temps limite, le temps moyen que la méthode la plus lente met à converger, cette méthode était toujours celle utilisant le décodage multiple.

5.2 Comparaisons

Les tableaux 8.14 et 8.15 présentent les résultats des comparaisons statistiques pour les différentes instances en utilisant l'indicateur d'hypervolume différence et l' ϵ-indicateur. Dans ces tableaux, le symbole $\succ$ signifie que la méthode de décodage nommée dans la ligne est statistiquement significativement meilleure que celle nommée dans la colonne. Le symbole $\prec$, indique à l'inverse, que la méthode nommée dans la ligne est statistiquement significativement moins bonne que celle nommée dans la colonne. Enfin le symbole $\simeq$ indique une équivalence statistique entre les deux méthodes.

En observant les tableaux, on peut noter que pour toutes les instances le décodeur multi-objectif est meilleur que les décodeurs utilisant une méthode de scalarisation. Les figures 8.16 et 8.17 montrent l'ensemble des fronts obtenus pour les 20 exécutions de chaque approche pour les instances de 10 unités et 80 unités. On peut voir que la méthode de décodage multi-objectif permet d'obtenir des front beaucoup mieux répartis que les deux autres méthodes.

Pour toutes les instances, excepté l'instance de 80 unités, le décodeur utilisant une méthode de scalarisation dont la pondération peut varier est statistiquement meilleur que celui pour lequel cette pondération est fixe quel que soit l'indicateur. Sur la figure 8.16

Décodeur	Poids fixe	Scalarisation embarquée	Multi
cas 10 unités			
I_{HD}	0.3774	0.3150	0.12725
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 20 unités			
I_{HD}	0.5045	0.1824	0.14016
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 40 unités			
I_{HD}	0.3835	0.21259	0.1598
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\prec$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 80 unités			
I_{HD}	0.6774	0.62529	1.542e-04
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 100 unités			
I_{HD}	1.2	1.2	0.0004358
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-

FIGURE 8.14 – Résultats des comparaisons statistiques pour l'indicateur d'hypervolume.

on peut observer que les fronts correspondants sont très différents. En effet, le front des solutions obtenues en utilisant la méthode de scalarisation embarquée est plus étendu que celui utilisant la méthode de décodage par scalarisation invariante. Pour l'instance de 80 unités en revanche, les résultats sont différents selon l'indicateur considéré, pour l'indicateur d'hypervolume différence, la méthode basée sur le décodage par scalarisation invariante est statistiquement meilleure que celle basée sur le décodage avec scalarisation embarquée mais on a les résultats inverses pour l'indicateur epsilon. Il semble que pour cette instance, il soit assez difficile pour l'algorithme de trouver des solutions réalisables et nous pensons que ce problème peut être renforcé de par le fait que l'espace de recherche pour la méthode de scalarisation embarquée soit agrandi car les valeurs α sont à fixer. Ce qui entraîne que finalement les solutions réalisables trouvées par l'algorithme sont très proches et leurs images sont concentrées dans une petite portion de l'espace objectif. La figure 8.17 montre les fronts Pareto obtenus en utilisant les différents types de décodage pour le cas 80 unités.

Décodeur	Poids fixe	Scalarisation embarquée	Multi
cas 10 unités			
I_ϵ^+	0.4199	0.3358	0.13969
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 20 unités			
I_ϵ^+	0.5080	0.2918	0.12212
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 40 unités			
I_ϵ^+	0.4396	0.2488	0.2052
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\prec$	-	$\succ$
Multi	$\succ$	$\succ$	-
cas 80 unités			
I_ϵ^+	0.9535	0.9340	0.0009245
Poids fixe	-	$\prec$	$\succ$
Scalarisation embarquée	$\prec$	-	$\prec$
Multi	$\succ$	$\succ$	-
cas 100 unités			
I_ϵ^+	0.9999	0.9979	0.0003935
Poids fixe	-	$\prec$	$\prec$
Scalarisation embarquée	$\succ$	-	$\prec$
Multi	$\succ$	$\succ$	-

FIGURE 8.15 – Résultats des comparaisons statistiques pour l'ε-indicateur.

6 Conclusion

Dans ce chapitre, nous avons tout d'abord formalisé le concept de problème à deux niveaux hiérarchiques multi-objectif. En nous basant sur cette formalisation, nous avons ensuite introduit la problématique de l'utilisation d'une représentation indirecte, par des métaheuristiques, si celle-ci implique un problème de décodage multi-objectif.

Pour répondre à cette problématique, nous avons proposé trois méthodes de décodage.

La première méthode est appelée méthode de scalarisation invariante. Elle consiste à transformer le problème de second niveau, traité par le décodeur, en un problème mono-objectif en utilisant une somme pondérée des objectifs dont les coefficients de pondération sont fixés à priori par l'utilisateur. Cette méthode assez naïve est très simple à mettre en œuvre, mais elle présente deux inconvénients majeurs :

— La scalarisation du problème de second ordre entraîne que certaines solutions Pareto optimales du problème complet ne sont pas accessibles.

— Il y a quelque chose d'illogique, dans le fait de fixer a priori des poids de pondérations à attribuer aux différents objectifs pour le sous-problème et de traiter le problème complet avec une approche d'optimisation multi-objectif a posteriori. En effet l'approche a posteriori est généralement utilisée lorsque l'on ne possède pas

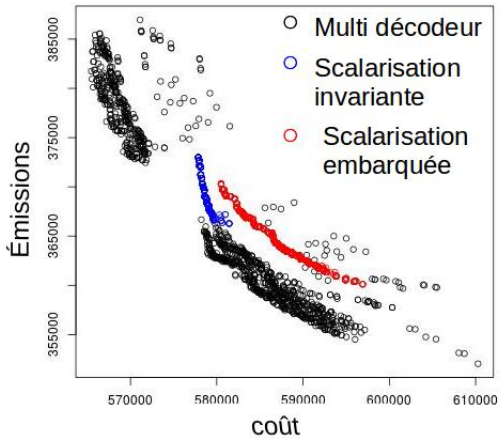

FIGURE 8.16 – Comparaison entre le front obtenu en utilisant le décodeur par scalarisation invariante (bleu), le décodeur avec scalarisation embarquée (rouge) et le décodeur multi-objectif (noir) pour le cas 20 unités.

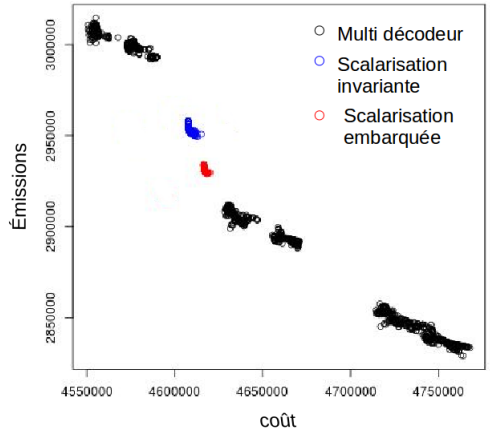

FIGURE 8.17 – Comparaison entre le front obtenu en utilisant le décodeur par scalarisation invariante (bleu), le décodeur avec scalarisation embarquée (rouge) et le décodeur multi-objectif (noir) pour le cas 80 unités.

d'information sur les préférences du décideur.

La seconde méthode que nous avons proposée, la méthode de scalarisation embarquée, tend à réduire ces deux inconvénients. Dans cette méthode, les coefficients de pondération utilisés lors du décodage sont inclus dans la représentation des solutions et peuvent être modifiés par les opérateurs de parcours de l'espace de recherche de la métaheuristique. Néanmoins, si le front Pareto du problème de second niveau n'est pas convexe, il reste possible que certaines solutions Pareto optimales ne soient pas accessibles à la métaheuristique.

La troisième méthode consiste à utiliser un décodeur multi-objectif, c'est à dire que

nous ne cherchons plus à associer une unique solution complète à chaque solution partielle, mais, à l'inverse, nous associons le front des solutions Pareto optimales pour le problème de second niveau à la solution partielle correspondante. Pour calculer ce front n'importe quelle méthode peut être utilisée. Cependant, il nous a alors fallu mettre en place l'assignation de valeur de fitness et de diversité aux solutions partielles. Pour cela, nous avons proposé des techniques génériques d'adaptation des méthodes d'assignation de fitness et de diversité à une solution partielle pour le cas où celle-ci peut être décodée par plusieurs solutions complètes.

Ensuite, nous avons appliqué ces trois techniques de décodage sur le problème d'affectation d'unités bi-objectif. En effet, nous avons vu que ce problème peut être modélisé comme un problème à deux niveaux hiérarchiques multi-objectifs. Le premier niveau correspond au problème de planification des temps de marche et d'arrêt des unités et le second niveau, au problème de distribution optimale. Le problème de second niveau est, dans ce cas, un problème continu et convexe qui peut être résolu en utilisant la méthode des sommes pondérées. Ainsi, dans ce cas, la méthode de décodage par scalarisation embarquée, permet théoriquement, d'accéder à l'ensemble des solutions Pareto optimales. Pour ces applications, nous avons utilisé NSGA-II avec des opérateurs évolutionnaires très basiques.

Nous avons ainsi pu faire des tests statistiques pour comparer les trois méthodes de décodage sur ce problème. Ces tests ont mis en avant l'intérêt de la méthode de décodage multi-objectif que nous avons proposée. Cette méthode donne en effet des fronts de bien meilleure qualité que les deux autres. Nous avons aussi pu remarquer que la méthode de scalarisation embarquée donne, dans l'ensemble, des résultats de meilleure qualité que la méthode de scalarisation invariante. Cependant, sur ce cas d'étude, nous avons pu remarquer que les valeurs des poids de pondérations attachées aux solutions ont tendance à converger, c'est-à-dire qu'après un certain nombre de générations les valeurs de pondérations α associées aux individus de la population sont toutes dans un sous-intervalle restreint de [0,1], ce qui nuit à la qualité des résultats.

Ainsi, concernant nos perspectives pour ce travail, nous pensons qu'il pourrait être intéressant d'étudier la méthode de scalarisation embarquée en essayant d'appliquer des méthodes de diversifications plus intenses sur les valeurs de pondérations et en utilisant éventuellement un système d'archivage.

Nous pensons également, que nos techniques d'adaptation des méthodes d'assignation de fitness et de diversité pour la méthode de décodage multiple pourraient être affinées. En effet, dans notre étude, mis à part pour le cas où les assignations utilisent un indicateur unaire, nous ne prenons pas en compte la qualité du front de solutions phénotypiques associées à une solution génotypique dans son ensemble lors du processus de sélection, or nous pensons que la prise en compte de cette information peut être bénéfique.

Afin de montrer qu'il est possible de généraliser l'idée de représentation indirecte avec décodeur multi-objectif, nous souhaiterions également l'appliquer à d'autres problèmes d'optimisation combinatoire multi-objectif.

Enfin, d'un point de vue plus pratique, nous aimerions mettre en place des outils génériques pour Paradiseo, qui permettent d'utiliser facilement le concept de décodage multiple.

Pour conclure, nous pensons que la problématique de décodage posée dans ce chapitre peut être un axe de recherche très intéressant. Nous pensons en particulier que l'idée du décodage multiple est prometteuse et que son application à d'autres problèmes pourrait donner de très bons résultats.

Chapitre 9

Application de MO-DYNAMOP au problème d'affectation d'unités multi-objectif

Dans ce dernier chapitre nous appliquons MO-DYNAMOP au problème d'affectation d'unités multi-objectif. Pour cela, nous utilisons une représentation indirecte avec un décodeur multi-objectif en nous basant sur l'étude faite dans le chapitre précédent. Sur ce problème, nous testons DYNAMOP en utilisant différents paradigmes d'algorithmes évolutionnaires tels que NSGA-II, IBEA et SMS-EMOA. Une étude comparative est faite entre les différentes versions de MO-DYNAMOP ainsi obtenues. Nous comparons également MO-DYNAMOP à trois algorithmes issus de la littérature.

Les travaux détaillés dans ce chapitre ont été présentés lors de la conférence MIC 2015 à Agadir, au Maroc [80].

1 Introduction

Nous avons pu voir dans le chapitre 6 que le problème d'affectation d'unités a rarement été étudié dans sa version multi-objectif. De plus, parmi les méthodes de la littérature résolvant le problème tel que nous l'avons formulé, nous n'avons relevé que deux méthodes qui traitent simultanément le sous-problème de planification des temps de marche et d'arrêt des unités et le sous-problème de la distribution de la production entre les unités allumées comme des problèmes multi-objectifs. Cette rareté peut sûrement s'expliquer par la difficulté de réutiliser les méthodes ayant été proposées pour le cas mono-objectif. En effet, ces méthodes s'appuient sur une stratégie de représentation indirecte qui n'a jamais été étudiée pour le cas multi-objectif.

Dans ce chapitre nous appliquons MO-DYNAMOP au problème d'affectation d'unités multi-objectif. Dans le cas mono-objectif, DYNAMOP avait permis d'obtenir des résultats dépassant ceux de la littérature et les avantages de cette méthode restant valables dans le cas multi-objectif, l'intérêt d'appliquer MO-DYNAMOP au cas multi-objectif est palpable.

Nous verrons que le problème de la représentation indirecte se pose pour appliquer MO-DYNAMOP au problème d'affectation d'unités. Cependant, nous avons étudié, dans le chapitre précédent, différentes stratégies permettant d'adapter le principe de la représentation indirecte au cas multi-objectif. Cette étude va nous permettre d'appliquer MO-DYNAMOP au problème d'affectation d'unités multi-objectif. Nous utiliserons la stratégie du décodeur multi-objectif, car elle donne les meilleurs résultats sur MOBGA. Cette stratégie sera testée sur trois types d'algorithmes évolutionnaires, NSGA-II, IBEA et SMS-EMOA.

Le travail présenté dans ce chapitre a donc un double intérêt, d'une part il permet de tester l'efficacité de MO-DYNAMOP sur un problème, qui du fait de sa difficulté et de son importance pratique, est très intéressant. Et d'autre part, il permet d'étudier l'application de la méthode de décodage proposée au chapitre 8 à différents types d'algorithmes évolutionnaires.

Étant donné que la définition du graphe d'états utilisée pour la programmation dynamique est essentiel à l'application de MO-DYNAMOP, nous commencerons ce chapitre en expliquant en détail comment la programmation dynamique multi-objectif peut s'appliquer à la résolution du problème d'affectation d'unités bi-objectif. Nous expliquerons ensuite, comment MO-DYNAMOP s'applique au problème, ainsi que les techniques de décodage multiple mises en place selon l'algorithme évolutionnaire utilisé. Des tests sont finalement effectués afin, d'une part, d'évaluer la qualité de MO-DYNAMOP en fonction de l'algorithme évolutionnaire multi-objectif utilisé et, d'autre part, de comparer MO-DYNAMOP aux méthodes issues de la littérature. Après analyse des résultats, nous conclurons en discutant de l'intérêt des approches proposées et des ouvertures possibles de ce travail.

2 Programmation dynamique multi-objectif pour le problème d'affectation d'unités bi-objectif

L'objectif de cette section est d'expliquer comment la programmation dynamique peut être utilisée pour résoudre le problème d'affectation d'unités multi-objectif. Pour cela nous allons présenter les éléments de l'équation fonctionnelle de la programmation dynamique pour ce problème.

Espace d'états

La deuxième fonction objectif étant totalement séparable par rapport aux décisions, son ajout ne modifie pas la définition de l'espace d'états par rapport au cas mono-objectif. Ainsi la définition de l'espace d'états donnée dans la section 2.1 du chapitre 4 reste valide. Un état S_t, associé à une période de temps t, est défini par un vecteur d'entiers $(S_{t,i})_{i=1...N}$, de sorte que $S_{t,i}$ est défini dans l'intervalle $[\![-T^{off}_{i,min} - T_{cs,i}; T^{up}_{i,min}]\!] - \{0\}$ et est à interpréter comme ceci :

- Si $S_{t,i} = T^{up}_{i,min}$ alors au temps t l'unité i est éteinte depuis au moins $T^{up}_{i,min}$ heures.
- Si $0 < S_{t,i} < T^{up}_{i,min}$ alors au temps t l'unité i est allumée depuis $S_{t,i}$ heures.
- Si $S_{t,i} = -T^{off}_{i,min} - T_{cs,i}$ alors au temps t l'unité i est éteinte depuis au moins $T^{off}_{i,min} + T_{cs,i}$ heures.
- Si $-T^{off}_{i,min} - T_{cs,i} < S_{t,i} < 0$ alors au temps t l'unité i est éteinte depuis $-S_{t,i}$ heures.

La fonction de base est également inchangée par rapport au cas mono-objectif.

Espace de décisions

L'espace des décisions se définit comme pour le cas mono-objectif. Une décision est définie par un vecteur binaire $(d_i)_i$ précisant pour chaque unité i si celle-ci sera allumée ou éteinte durant la prochaine période de temps.

$$d \in D(S_t) \Leftrightarrow \forall i$$

- Si $0 < S_{t,i} < T^{up}_{i,min}$ alors $d_i = 1$
- Si $S_{t,i} = T^{up}_{i,min}$ alors $d_i \in \{-1, 1\}$

— Si $-T^{off}_{i,min} < S_{t,i} < 0$ alors $d_i = -1$
— Si $-T^{off}_{i,min} \leq S_{t,i}$ alors $d_i \in \{-1, 1\}$
— $\sum_{i,d_i>0} P_{min,i} \leq D_t$ et $\sum_{i,d_i>0} P_{max,i} \geq D_t + R_t$

Fonction coût

L'utilisation de la fonction coût dans le cas du problème d'affectation d'unités multi-objectif est un peu spécifique. En effet, une décision d telle que définie précédemment ne précise pas les quantités qui sont produites par chacune des unités. Une décision d peut donc être considérée comme un ensemble indénombrable de décisions $p \in P^d$ donnant la production exacte de chaque unité de production.

La fonction coût va donc associer à un arc non pas un vecteur objectif mais un ensemble de vecteurs objectifs correspondant à l'ensemble des solutions Pareto optimales du problème suivant :

$$\min_p \{f^t_{obj,1}(p) = \sum_i CF_i(p), f^t_{obj,2}(p) = \sum_i CE_i(p)\} \tag{9.1}$$

tel que :

$$\begin{array}{l} P_{min,i} \leq p_i \leq P_{max,i} \quad \forall i, d_i = 1 \\ \sum_{i,d_i=1} p_i = D_{t+1} \end{array} \tag{9.2}$$

Pour approximer le front Pareto associé, on résout la version scalarisée définie par :

$$\min_p \alpha f_{obj,1}(p) + (1-\alpha) f_{obj,2}(p)$$

tel que les contraintes (9.2) soient respectées, pour $\alpha \in \{\frac{k}{n_\alpha}, k = 0...n_\alpha\}$, où n_α est une constante à fixer. On note $R^\alpha(S, d)$ la solution de ce problème.

Fonction de transformation

Comme les définitions des états et des décisions ne changent pas, la fonction de transformation reste également inchangée par rapport au cas mono-objectif :

$$T(S_t, d) = S_{t+1}$$

tel que :

$$S_{t+1,i} = \begin{cases} T^{up}_{i,min} & \text{si } S_{t,i} = T^{up}_{i,min} \text{ et } d_i = 1 \\ -1 & \text{si } S_{t,i} = T^{up}_{i,min} \text{ et } d_i = -1 \\ -T^{off}_{i,min} - T_{cs,i} & \text{si } S_{t,i} = -T^{off}_{i,min} - T_{cs,i} \text{ et } d_i = -1 \\ 1 & \text{si } S_{t,i} = -T^{off}_{i,min} - T_{cs,i} \text{ et } d_i = 1 \\ S_{t,i} + d_i & \text{sinon} \end{cases} \tag{9.3}$$

Opérateur $\circ$

Soit $\mathcal{F}^k(S)$ l'ensemble des éléments du front Pareto associé à l'état S obtenus en scalarisant le sous-problème de distribution avec la valeur $\alpha = \frac{k}{n_\alpha}$, alors si $e \in F^k(S)$:

$$R(S, d) \circ e = \{R^{\frac{k}{n_\alpha}}(S, d) + e, k = 1...n_\alpha\}$$

3 MO-DYNAMOP pour le problème d'affectation d'unités

Dans cette section nous expliquons comment MO-DYNAMOP est appliqué au problème d'affectation d'unités multi-objectif.

3.1 Représentation

Une solution est représentée comme une séquence d'états correspondant à un chemin valide. Le même choix est fait qu'au chapitre 4, c'est-à-dire que la contrainte interdisant la prise de décisions ne permettant pas le respect de la demande et de la réserve est relâchée. Bien sûr, la prise de telles décisions devra être pénalisée par la fonction coût. Au final, une solution est donc représentée exactement comme dans le cas mono-objectif et l'ensemble des génotypes valides est donc le même.

3.2 Décodage et évaluation

Le décodage d'un individu se fait arc par arc. Soit P_S, l'ensemble des solutions phénotypiques attachées à la solution génotypique, $S = (S_t)_t$. Si $P_{S_t,S_{t+1}}$ est l'ensemble des solutions $\{P^{\alpha}_{S_t,S_{t+1}}, \alpha = \frac{k}{n_\alpha}, k = 0...n_\alpha\}$ obtenues en résolvant les sous problèmes $\Pi^{\alpha}_{S_t,S_{t+1}}$ définis par :

$$\min_p \alpha f^t_{obj,1}(p) + (1-\alpha) f^t_{obj,2}(p)$$

tel que :

$$\begin{array}{ll} P_{min,i} \leq p_i \leq P_{max,i} & \forall i, S_{t+1,i} > 0 \\ \sum_{i,S_{t+1,i}>0} p_i = D_{t+1} & \end{array} \tag{9.4}$$

alors, P_S est défini par :

$$P_S = \{(P^{\alpha}_{S_t,S_{t+1}})_{t=0...T-1}, \alpha = \frac{k}{n_\alpha}, k = 0...n_\alpha\}$$

La figure 3.2 illustre le procédé de décodage. L'ensemble des solutions phénotypiques P_S

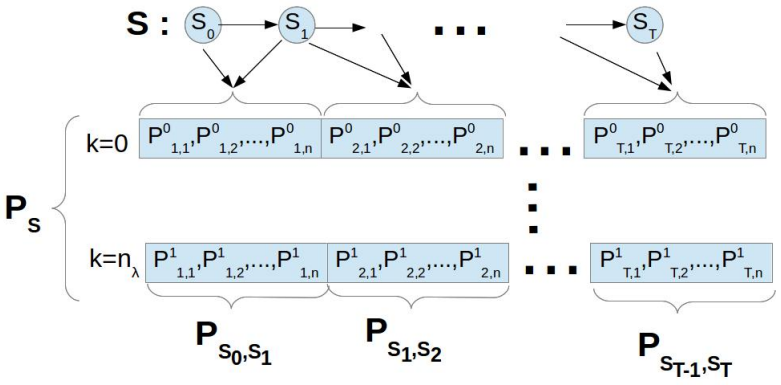

FIGURE 9.1 – Processus de décodage arc par arc.

attaché à une solution S est conservé. Les valeurs $R^{\alpha}_i(S_t, S_{t+1}) = f^t_{obj,1}(P^{\alpha}_{S_t,S_{t+1}})$ sont également conservées sous forme d'une matrice à deux dimensions $(e^t_{i,\alpha})_{i=1,2,\alpha}$. Dans le

cas où il n'est pas possible de répondre à la demande ou de respecter la réserve minimale, c'est-à-dire dans le cas où l'une des conditions suivantes est vérifiée :

$$\begin{array}{ll} \sum\limits_{i,S_{t+1,i}>0} P_{max,i} & \leq D_{t+1} + R_{t+1} \\ \sum\limits_{i,S_{t+1,i}>0} P_{min,i} & \geq D_{t+1} \end{array} \tag{9.5}$$

on pose que $P^{\alpha}_{S_t,S_{t+1}} = 0$ pour toutes les valeurs de α et, pour chaque objectif i et toutes valeurs de α, que $e^t_{i,\alpha} = c_p(\max(D_t + R_t - \sum\limits_{i,S_{t+1,i}>0} P_{max,i}, 0) + \max(\sum\limits_{i,S_{t+1,i}>0} P_{min,i} - D_t, 0))$ où c_p est une constante positive de grande valeur.

La séparabilité du décodage et de l'évaluation par rapport aux arcs permet de réduire le nombre d'opérations et donc le temps de calcul. En effet, lorsqu'un individu est modifié par un opérateur, il suffira de reconstruire les ensembles $P_{St,S_{t+1}}$ et de recalculer les valeurs $e^{t+1}_{i,\alpha}$ uniquement pour les arcs ayant été introduits par les opérateurs.

3.3 Opérateurs évolutionnaires

Nous présentons maintenant les opérateurs évolutionnaires de croisement et de mutation qui sont utilisés dans cette application de DYNAMOP. Il y a deux types d'opérateurs, les opérateurs classiques, qui se basent sur le hasard pour générer de nouvelles solutions à partir de solutions de la population, et les opérateurs intelligents qui utilisent la programmation dynamique pour améliorer des solutions existantes.

Opérateurs classiques

L'espace de recherche étant inchangé par rapport au cas mono-objectif, les opérateurs de mutation et de croisement sont les mêmes que ceux proposés au chapitre 4 respectivement dans les sections 3.3 et 3.4.

Opérateurs intelligents

Les opérateurs de mutation intelligents sont simplement des extensions de ceux proposés au chapitre 4 section 3.5 au cas multi-objectif. Ainsi les principes de constructions de graphe d'états restreints sont gardés tels quels mais c'est la programmation dynamique multi-objectif qui est utilisée. Néanmoins, afin de réduire la taille des fronts de Pareto associés aux arcs à un vecteur objectif et ainsi réduire les temps de calcul, le sous-problème de distribution est résolu pour une valeur de α aléatoirement fixée. Ces opérateurs fournissent un front Pareto de solutions Pareto équivalentes. Les solutions de ce front sont toutes ajoutées à la population enfant, la population constituant la prochaine génération est ensuite constituées des N_P meilleurs individus parmi les parents et les enfants, où N_P est un paramètre de l'algorithme.

3.4 Gestion de l'archive

Une archive non bornée est utilisée pour conserver les meilleurs solutions phénotypiques trouvées par l'algorithme. Cette archive est mise à jour après chaque itération.

4 Application de différents algorithmes évolutionnaires avec décodeur multi-objectif

MO-DYNAMOP est testé avec plusieurs algorithmes évolutionnaires, la différence entre ces algorithmes réside dans leurs stratégies d'assignation de valeur de fitness et/ou de diversité aux solutions. Dans le cas du problème d'affectation d'unités de production multi-objectif, ces stratégies doivent être adaptées afin de prendre en considération le fait que de multiples solutions phénotyiques sont attachées à chaque solution génotypique. Une méthodologie générique d'adaptation de ces stratégies d'assignation a été proposée au chapitre 8, et c'est cette méthodologie que nous utilisons ici.

4.1 NSGA-II

L'algorithme NSGA-II a été présenté au chapitre 6. Dans cet algorithme, la valeur de fitness assignée à une solution est son rang de dominance et sa valeur de diversité correspond à sa distance de *crowding* avec les autres solutions de même rang.

L'adaptation de ces processus d'assignation a déjà été expliquée en détail au chapitre 8 section 4.3. Pour rappel, la fitness attribuée à une solution génotypique S est le meilleur rang parmi ceux des solutions phénotypiques P_S, qui lui sont associées :

$$fit(S) = \min_{p \in P_S} rang(p)$$

La figure 8.6 du chapitre 8 illustre ce procédé d'assignation d'une valeur de fitness. La valeur de diversité correspond alors à plus grande distance de *crowding* entre un élément de P_S de rang $fit(S)$ et l'ensemble des solutions phénotyiques de rang $fit(S)$ issues d'un autre génotype. La figure 8.7 du chapitre 8 illustre ce procédé d'assignation d'une valeur de diversité.

4.2 IBEA

L'algorithme IBEA a également été présenté au chapitre 6. C'est un algorithme qui se base sur un indicateur de performance binaire I pour évaluer une solution complète x par rapport à un critère de convergence et/ou de diversification :

$$fit(x) = \sum_{y \in P - \{x\}} (-e^{-I(\{y\},\{x\})/\kappa})$$

où κ est un réel à fixer et P représente l'ensemble de la population. L'objectif de cette fonction fitness est de mesurer la perte en qualité si la solution x est retirée de la population P.

Une première idée naturelle pour étendre IBEA au cas où un ensemble de solutions phénotypiques Pareto équivalentes, P_S, est attaché à une unique solution génotypique, S, est de garder la même définition de la fonction fitness mais en utilisant l'indicateur binaire I pour comparer deux à deux les ensembles de solutions phénotypiques associés aux solutions génotypiques. Ainsi la méthode d'assignation de fitness serait :

$$fit(S) = \sum_{S' \in P - \{S\}} (-e^{-I(P_{S'},P_S)/\kappa}) \tag{9.6}$$

Mais cette méthode n'est pas consistante avec la relation de Pareto dominance. En effet, il est possible, avec cette méthode, qu'une solution S_1 dont tous les éléments phénotypiques associés P_{S_1} sont dominés obtienne une meilleure évaluation qu'une solution S_2 dont l'ensemble des solutions phénotypiques associées P_{S_2} est non dominé.

Ceci est lié au fait qu'un ensemble de solutions phénotypiques P_1 peut être simultanément meilleur pour l'indicateur I qu'un ensemble P_2 et qu'un ensemble P_3 mais être, pour ce même indicateur, moins performant que l'union de P_2 et P_3. C'est à dire qu'il est possible qu'il existe trois ensembles de solutions phénotyiques Pareto équivalentes P_1, P_2 et P_3 tels que :

$$\begin{aligned} I(P_2, P_1) &> I(P_1, P_2) \\ I(P_3, P_1) &> I(P_1, P_3) \\ I(P_3 \cup P_2, P1) &< I(P_1, P_3 \cup P_2) \end{aligned}$$

Cette situation est illustrée par la figure 4.2, dans le cas ou l'indicateur binaire est l'indicateur d'hypervolume différence.

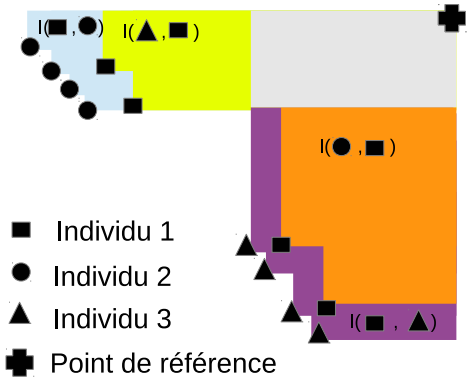

FIGURE 9.2 – Dans cet exemple l'utilisation de la formule d'attribution de fitness 9.6 avantagerait l'individu 1 sur les individus 2 et 3, alors même que les individus 2 et 3 incluent des solutions phénotypiques de meilleure convergence.

Pour cette raison, la proposition faite au chapitre précédent attribuant à la solution génotypique la valeur de fitness :

$$fit(x) = \max_{p_s \in P_S} \sum_{p \in P_c - P_S} (-e^{-I(p,p_s)/\kappa}) \tag{9.7}$$

où P_c est l'ensemble des solutions phénotypiques, semble pouvoir être plus profitable. Nous avons néanmoins implémenté les deux approches et nous en ferons une analyse comparative. La première approche sera, par la suite, notée IBEAV1 et la seconde IBEAV2.

4.3 SMS-EMOA

SMS-EMOA a été introduit au chapitre 6. Dans cet algorithme évolutionnaire, le processus de sélection s'effectue en deux étapes. D'abord les individus de meilleur rang de dominance sont sélectionnés (assignation de fitness), puis, pour distinguer les solutions de même rang quand le nombre de places disponibles restantes dans la population est limité, l'indicateur unitaire d'hypervolume différence est utilisé (assignation de diversité).

L'adaptation du processus d'assignation de fitness à la prise en compte de solutions phénotypiques multiple, se fait comme pour le cas de NSGA-II :

$$fit(S) = \min_{p \in P_S} rang(p)$$

Et comme expliqué au chapitre 8, la mesure de diversification devient :

$$fit(S) = hypervolume(P_c) - hypervolume(P_c - P_S)$$

Cette mesure est illustrée par la figure 8.5 du chapitre 8.

5 Protocole expérimental

Dans cette section le protocole utilisé pour évaluer les différentes approches proposées et les comparer à des méthodes de la littérature est expliqué en détail.

Instances

Deux jeux d'instances sont utilisés. Le premier est issu de la thèse de Ana Maria Marques de Moura Gomes Viana [123] et a été présenté au chapitre 8 section 5.1. Le second est issu de l'article [110], présenté au chapitre 6, qui propose un algorithme évolutionnaire mémétique multi-objectif. Ce second jeu d'instances ne comporte que deux instances, une instance de 10 unités et une instance de 100 unités. L'instance de 10 unités est similaire à celle que nous avons utilisée pour le cas mono objectif et qui est décrite par les tableaux 5.2 et 4.2 du chapitre 4, excepté que des coefficients supplémentaires b_i sont donnés pour décrire la fonction d'émissions en SO_2 et CO_2. Ces coefficients sont décrits dans le tableau 9.1 . L'instance de 100 unités est obtenue en dupliquant l'instance de 10 unités.

unité	1	2	3	4	5	6	7	8	9	10
b_0	42.9	42.9	40.27	40.27	13.86	13.86	330	330	350	360
b_1	-0.5112	-0.5112	0.5455	-0.5455	0.3277	0.3277	-3.9023	-3.9023	-3.9524	-3.9864
b_2	0.0046	0.0046	0.0068	0.0068	0.0042	0.0042	0.0465	0.0465	0.0465	0.0470

Tableau 9.1 – Coefficient de la fonction de mesure des émissions de gaz à effet de serre.

Mesure de performance

Les indicateurs de performances qui sont utilisés sont l'indicateur d'hypervolume différence et l'ϵ-indicateur. Ces indicateurs ont été présentés dans le chapitre 6.

Paramétrage

Une analyse de sensibilité est effectuée pour fixer la valeur des paramètres associés à chacune des méthodologies pour les différentes instances. Cette analyse est effectuée grâce au package *Irace* [113] du logiciel de statistique *R*. Les paramètres fixés via cette analyse sont :

- La probabilité pour un individu de subir une mutation : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- La probabilité pour un individu d'intervenir dans un croisement : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- La probabilité d'une mutation intelligente du premier type : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- La probabilité d'une mutation intelligente du second type : valeur testée dans l'intervalle $[0, 1]$ avec une précision de 10^{-3}.
- Nombre d'unités modifiées par la mutation intelligente du second type : valeur testée dans l'intervalle $[\![2, 10]\!]$.

— Nombre de solutions phénotypiques attachées à une solution génotyique, n_α : valeur est testée dans l'intervalle $[\![3, 15]\!]$.

La taille de la population est fixée à 20 individus et le critère d'arrêt utilisé lors du paramétrage est un temps d'exécution maximal de $30 \times N$ secondes.

Mise en œuvre des tests expérimentaux

Les comparaisons sont faites en lançant 20 exécutions de chaque méthode. Pour chaque méthode, les mêmes 20 "graines" sont utilisées. Ces graines permettent de définir les valeurs renvoyées par les fonction d'aléas (*random*), ainsi les méthodes sont appliquées au même ensemble de populations initiales et nous pouvons considérer que les tests sont appariés. Tous les tests sont effectués en utilisant le logiciel Pisa [23] sur la base du test de Friedman qui permet de tester la significativité des différences entre les méthodes et d'un test post-hoc permettant de comparer les méthodes deux à deux. La p-valeur limite utilisée pour rejeter l'hypothèse nulle lors d'un test statistique est toujours de 0,01.

Nous comparerons les applications des différents algorithmes évolutionnaires (NSGA-II, IBEA, SMS-EMOA) à MO-DYNAMOP entre elles et à deux algorithmes de la littérature, qui ont été présentés dans l'état de l'art du chapitre 6, mCON [123] et MOMAS [110]. L'étude comparative inclut également l'algorithme génétique, MOBGA, présenté dans le chapitre 8, utilisant le décodeur multiple auquel une archive non bornée a été ajoutée par soucis d'équité.

Dans un premier temps nous comparons les applications des différents algorithmes évolutionnaires (NSGAII, IBEA, SMS-EMOA) à MO-DYNAMOP pour le problème d'affectation d'unités de production multi-objectif. Nous comparerons également ces quatre versions de MO-DYNAMOP à mCON et à MOBGA. Nous utilisons à chaque fois une population de 20 individus et un nombre maximal de 500 générations comme critère d'arrêt. Cela correspond aux paramétrages utilisés par l'auteure de mCON pour générer les fichiers de résultats qu'elle nous a fournis et que nous utilisons pour réaliser notre analyse de comparaison statistique.

La comparaison avec MOMAS se fait seulement en comparant les meilleures valeurs extrémales que nous obtenons au cours de 20 exécutions à celles fournies dans l'article [110]. En effet, comme nous ne disposons pas de fichiers de résultats plus détaillés nous n'avons pas pu effectuer de tests statistique rigoureux. Les 20 exécutions sont lancées en utilisant la même taille de population que celle donnée dans l'article présentant MOMAS, c'est-à-dire N. Nous posons que l'algorithme s'arrête après que $50 \times N$ générations se soient écoulées, le critère d'arrêt utilisé pour MOMAS étant également de $50 \times N$ générations.

6 Résultats

Dans cette section nous présentons les résultats de nos expérimentations et de notre analyse statistique.

6.1 Comparaison à mCON et MOBGA

Le tableau 9.3 présente les résultats des comparaisons statistiques entre MOBGA, mCON [123] et les différentes versions de MO-DYNAMOP pour l'indicateur d'hypervolume différence. Le tableau 9.4 présente quand à lui les résultats obtenus en utilisant l'indicateur ϵ-différence et le tableau 9.5, donne les meilleures valeurs trouvées par chacun des algorithmes au cours des 20 exécutions pour chacun des objectifs.

Dans ces deux tableaux "IBEAV1" réfère à l'adaptation d'IBEA pour laquelle l'indicateur est utilisé pour comparer deux à deux des ensembles de solutions phénotypiques associées aux solutions génotypiques et "IBEAV2" désigne la version pour laquelle l'indicateur est utilisé pour comparer deux à deux des solutions phénotyiques.

Le symbole "$\succ$" est utilisé pour signifier que la méthode correspondant à la ligne est statistiquement significativement meilleure que la méthode méthode correspondant à la colonne du tableau. Le symbole "$\prec$" signifie à l'inverse que la méthode correspondant à la ligne est significativement moins efficace que celle correspondant à la colonne, enfin le symbole "$\simeq$" signifie qu'il n'existe pas de différence statistiquement significative entre les deux méthodes.

Nous pouvons tout d'abord noter que pour toutes les instances toutes les versions de MO-DYNAMOP sont significativement meilleures que mCON et MOBGA et que mCON est significativement meilleure que MOBGA et ceci pour les deux indicateurs.

Les résultats des comparaisons entre les différentes versions de MO-DYNAMOP varient en revanche d'une instance à l'autre.

Pour le cas 10 unités avec l'indicateur d'hypervolume différence, Les version de MO-DYNAMOP basées sur IBEAV1 et NSGAII sont toutes 2 meilleures que celles basées sur IBEAV2 et les performances de IBEAV1, NSGA-II et SMS sont statistiquement équivalentes. SMS est également équivalent à IBEAV2. En revanche, si on considère l'indicateur ϵ, l'utilisation de IBEAV1 donne des résultats significativement inférieurs aux 3 autres algorithmes évolutionnaires alors que NSGA-II est également dominant pour cet indicateur.

Pour le cas du système de 20 unités, en utilisant l'indicateur d'hypervolume différence, les versions de MO-DYNAMOP basées sur IBEAV1 et NSGA-II sont toutes deux statistiquement meilleures que celles basées sur IBEAV2 et SMS-EMOA. La seconde version de IBEA et SMS-EMOA sont statistiquement équivalentes. En utilisant l'indicateur epsilon, les résultats sont les mêmes sinon que la méthode basée sur NSGA-II domine celle basée sur IBEAV1 et que SMS-EMOA domine IBEAV2.

Pour le système composé de 40 unités, la méthode utilisant la première version d'IBEA domine toutes les autres méthodes quel que soit l'indicateur. En fait les résultats sont les mêmes pour les deux indicateurs et permettent de classifier les approches comme ceci :

$$IBEAV1 \succ NSGA-II \succ IBEAV2 \succ SMS-EMOA$$

Pour le système de 80 unités, en considérant l'indicateur d'hypervolume le classement des méthodes est le suivant :

$$SMS-EMOA \succ NSGA-II \succ IBEAV1 \succ IBEAV2$$

Pour l'indicateur ϵ en revanche, c'est NSGA-II qui domine toutes les autres méthodes, IBEAV1 domine SMS-EMOA et IBEAV2. SMS-EMOA et IBEAVE sont statistiquement équivalentes.

Enfin, pour le système de 100 unités, si nous considérons l'indicateur d'hypervolume différence, les deux versions d'IBEA sont équivalentes et dominent les autres approches. SMS-EMOA est statistiquement meilleur que NSGAII. En revanche, avec l'indicateur epsilon, la seconde version de IBEA est statistiquement meilleure que la première.

Nous avons fait une étude comparative entre les différentes méthodes de résolution prenant en compte les résultats sur toutes les instances, afin de déterminer si le choix d'une des méthodes peut etre préféré aux autres pour résoudre l'ensemble des instances. Comme les étude précédentes, cette étude s'est basée sur le test statistique de comparaison de Friedmann, qui nous a permis d'établir que le choix de la méthode a une influence significative sur les résulats avec un risque d'erreur inférieur à 1%. Ce test réalisé, nous

avons effectué un test post-hoc qui a permis d'établir que l'ensemble des variantes de MO-DYNAMOP sont préférables à mCON et à MOBGA et que mCON est préférable à MOBGA. Il n'y a pas de différence significative entre les différentes version de MO-DYNAMOP si ce n'est que la première version basée sur IBEA est préférable à la seconde.

6.2 Comparaison à MOMAS

Le tableau 9.2 présente les solutions extrémales minimales obtenues avec MOMAS [110] et celles obtenues avec MO-DYNAMOP en utilisant NSGA-II. Nous pouvons voir que pour les deux instances les solutions extrêmes obtenues avec MO-DYNAMOP sont meilleures ou équivalentes à celles obtenues avec MOMAS. Nous pouvons également noter que pour l'instance de 10 unités la solution trouvée pour l'objectif de minimisation des coûts par les deux algorithmes correspond à la solution optimale de la version mono-objectif du problème. Pour l'instance de 100 unités, pour ce même objectif, l'écart relatif entre la solution extrémale fournie par MOMAS est de 0,1456% alors que celui avec la solution extrémale de MO-DYNAMOP n'est que de 0,0368%.

Nb. Unités	objectif	MOMAS	MO-DYNAMOP
10	coût	563 938	563 938
	émission	33 062	**32 862**
100	coût	5 605 918	**5 599 830**
	émission	329 938	**326 343**

Tableau 9.2 – Comparaison avec les solutions extrémales obtenues avec MOMAS.

7 Conclusion

Dans ce chapitre, nous avons appliqué MO-DYNAMOP au problème d'affectation d'unités multi-objectif.

Pour cela, nous avons utilisé le principe de décodeur multi-objectif introduit au chapitre 8 avec trois types d'algorithmes évolutionnaires : NSGA-II, IBEA et SMS-EMOA. Pour IBEA, nous avons appliqué deux versions d'adaptation du processus d'assignation de valeur de fitness à un génotype : dans la première version, IBEAV1, l'indicateur binaire est directement utilisé pour comparer deux à deux les fronts Pareto associés aux solutions génotypiques. Mais ce processus d'assignation de fitness peut être inconsistant avec la notion de Pareto dominance. La seconde version, IBEAV2, correspond à la stratégie d'adaptation proposée dans le chapitre 8.

Nous avons ensuite fait des comparaisons statistiques entre les quatre différentes versions de MO-DYNAMOP, MOBGA, l'algorithme évolutionnaire introduit au chapitre 8 qui utilise le décodeur multiple et mCON.

Nous avons pu voir que pour toutes les instances et quelque soit l'indicateur de qualité considéré, toutes les versions de MO-DYNAMOP sont significativement meilleures que les deux autres algorithmes et mCON est statistiquement meilleur que BGA.

En revanche, en ce qui concerne la comparaison des différentes versions de DYNAMOP, les résultats obtenus sont différents pour chaque instance. Pour les instances de 10 et 20 unités NSGA-II semble être l'approche la plus intéressante puisqu'elle donne les meilleurs

			DYNAMOP				mCON	MOBGA
			IBEAV1	IBEAV2	NSGAII	SMS-EMOA		
cas 10 unités		I_{HD}	0.03512	0.04684	0.03480	0.04087	0.3071	0.30507
	DYNAMOP	IBEAV1	X	$\succ$	$\succeq$	$\succeq$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\succeq$	$\succ$	$\succ$
		NSGAII	$\succeq$	$\succ$	X	$\succeq$	$\succ$	$\succ$
		SMS-EMOA	$\succeq$	$\succeq$	$\succeq$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succeq$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\succeq$	X
cas 20 unités		I_{HD}	0.6089	0.6154	0.6081	0.6156	0.62594	0.63205
	DYNAMOP	IBEAV1	X	$\succ$	$\succeq$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\succeq$	$\succ$	$\succ$
		NSGAII	$\succeq$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\succeq$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 40 unités		I_{HD}	0.03069	0.05929	0.04590	0.08613	0.4348	0.5123
	DYNAMOP	IBEAV1	X	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\succ$	$\succ$	$\succ$
		NSGAII	$\prec$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\prec$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 80 unités		I_{HD}	0.086	0.007754	0.004175	0.002763	0.02685	0.06798
	DYNAMOP	IBEAV1	X	$\succ$	$\prec$	$\prec$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\prec$	$\succ$	$\succ$
		NSGAII	$\succ$	$\succ$	X	$\prec$	$\succ$	$\succ$
		SMS-EMOA	$\succ$	$\succ$	$\succ$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 100 unités		I_{HD}	0.1107848	0.008414	0.1094	0.1057	0.1337	
	DYNAMOP	IBEAV1	X	$\succeq$	$\succ$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\succeq$	X	$\succ$	$\succ$	$\succ$	$\succ$
		NSGAII	$\prec$	$\prec$	X	$\prec$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\prec$	$\succ$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X

FIGURE 9.3 – Comparaisons statistiques basée sur l'indicateur d'hypervolume entre les différentes versions de DYNAMOP, mCON [123] et l'algorithme génétique de base MOBGA.

résultats quelque soit l'indicateur de qualité considéré. Pour l'instance de 40 unité IBEAV1 domine les autres approches pour les deux indicateurs de qualité. Pour l'instance de 80

			DYNAMOP				mCON	MOBGA
			IBEAV1	IBEAV2	NSGAII	SMS-EMOA		
cas 10 unités		I_ϵ^+	0.1188	0.10314	0.05502	0.09733	0.2853	0.28313
	DYNAMOP	IBEAV1	X	$\prec$	$\prec$	$\prec$	$\succ$	$\succ$
		IBEAV2	$\succ$	X	$\prec$	$\prec$	$\succ$	$\succ$
		NSGAII	$\succ$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\succ$	$\succ$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\simeq$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\simeq$	X
cas 20 unités		I_ϵ^+	0.8722	0.8764	0.8717	0.8746	0.87511	
	DYNAMOP	IBEAV1	X	$\succ$	$\prec$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\succ$	$\succ$	$\succ$
		NSGAII	$\succ$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\prec$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 40 unités		I_ϵ^+	0.04605	0.07934	0.06395	0.08222	0.4389	0.474
	DYNAMOP	IBEAV1	X	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\prec$	X	$\prec$	$\succ$	$\succ$	$\succ$
		NSGAII	$\prec$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\prec$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 80 unités		I_ϵ^+	0.072567	0.005148	0.003784	0.006196	0.0223	0.034
	DYNAMOP	IBEAV1	X	$\prec$	$\prec$	$\prec$	$\succ$	$\succ$
		IBEAV2	$\succ$	X	$\prec$	$\succ$	$\succ$	$\succ$
		NSGAII	$\succ$	$\succ$	X	$\succ$	$\succ$	$\succ$
		SMS-EMOA	$\succ$	$\prec$	$\prec$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X
cas 100 unités		I_ϵ^+	0.10898	0.008382	0.1212	0.1187	0.12	0.1523
	DYNAMOP	IBEAV1	X	$\prec$	$\succ$	$\succ$	$\succ$	$\succ$
		IBEAV2	$\succ$	X	$\succ$	$\succ$	$\succ$	$\succ$
		NSGAII	$\prec$	$\prec$	X	$\prec$	$\succ$	$\succ$
		SMS-EMOA	$\prec$	$\prec$	$\succ$	X	$\succ$	$\succ$
	mCON		$\prec$	$\prec$	$\prec$	$\prec$	X	$\succ$
	MOBGA		$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	X

FIGURE 9.4 – Comparaisons statistiques basées sur l'indicateur epsilon entre les différentes versions de DYNAMOP, mCON [123] et l'algorithme génétique de base BGA.

unités, en considérant l'indicateur d'hypervolume différence, SMS-EMOA est l'algorithme donnant les meilleurs résultats, mais si on considère l'indicateur d'epsilon différence, NS-

	cas 10 unités			cas 20 unités	
	Objectif	coût	émission	coût ($\times 10^6$)	émission
DYNAMOP	IBEAV1	563 977	351 901	1,12 432	696 542
	IBEAV2	56 4406	351 901	1,12 505	697 401
	NSGAII	563 938	351 660	1,12 330	696369
	SMS-EMOA	563 938	351 832	1,12 384	696 518
mCON		566514	363806	1, 12 776	726 215
MOBGA		565465	352042	1,12 759	704 024

	cas 40 unités			cas 80 unités	
	Objectif	coût($\times 10^6$)	émission ($\times 10^6$)	coût ($\times 10^6$)	émission ($\times 10^6$)
DYNAMOP	IBEAV1	2,24 425	1,38 876	4,49 530	2, 79135
	IBEAV2	2,24 629	1,39 863	4,49 467	2,81 288
	NSGAII	2,24 338	1,38 876	4,49 381	2,79 061
	SMS-EMOA	2,24 496	1,39 060	4,48 803	2,78 588
mCON		2,25 050	1,46 117	4,49 961	2,93 957
MOBGA		2,267 550	1,41 314	4,55 070	2,82 913

	cas 100 unités		
	Objectif	coût ($\times 10^6$)	émission ($\times 10^6$)
DYNAMOP	IBEAV1	5,64 208	3,49 707
	IBEAV2	5,60 945	3,52 293
	NSGAII	5,62 235	3,49 774
	SMS-EMOA	5,61 273	3,49 018
mCON		5,62 483	3,68 086
MOBGA		5,72 815	3,56 775

FIGURE 9.5 – Solutions extrémales obtenues avec les différentes versions de DYNAMOP, mCON [123] et l'algorithme génétique de base BGA.

GAII permet d'obtenir des résultats significativement meilleurs à ceux obtenus en utilisant les autres algorithmes évolutionnaires. Enfin, pour l'instance de 100 unités, l'analyse statistique montre qu'il est plus avantageux d'utiliser IBEAV2, vis-à-vis des deux indicateurs de qualité.

En faisant des tests considérant simultanément toutes les instances, nous avons pu montrer que toutes les versions de DYNAMOP sont équivalentes pour le problème d'affectation d'unités excepté que la première version de IBEA est préférable à la seconde.

Il est d'ailleurs intéressant de constater que IBEAV1 donne de très bons résultats et est statistiquement significativement meilleur que IBEAV2 lorsque l'on considère l'ensemble des instances. En effet, a priori le problème d'inconsistance de cette approche, par rapport à la Pareto dominance, que nous avions relevé (illustré par la figure 4.2), nous faisait présager que les résultats pourraient être mauvais. Il est possible que ce problème n'ait pas une influence négative forte sur le problème d'affectation d'unités sur ces instances. Un élément explicatif pourrait être que l'expansion des fronts Pareto associés à chaque solution génotypique varie peu et il serait intéressant de vérifier cette hypothèse dans un travail futur. Aussi, l'avantage possible de cette approche est qu'elle permet de prendre

en considération la qualité du front associé à la solution génotypique plutôt que de considérer uniquement chaque solution le composant individuellement, ce qui permet prendre en compte dans l'évaluation des solutions génotypiques leur capacité, une fois décodée, à dominer une grande partie de l'espace objectif.

Nous avons également comparé MO-DYNAMOP à MOMAS, mais comme nous n'avions pas de fichiers de résultats pour mener une étude statistique, nous nous sommes simplement comparé en examinant les solutions extrêmes trouvées par les deux algorithmes. Nous avons pu constater que les résultats de MO-DYNAMOP sont meilleurs ou équivalents pour chaque borne.

L'étude menée dans ce chapitre, nous permet d'affirmer que MO-DYNAMOP est une méthode très efficace pour traiter le problème d'affectation d'unités bi-objectif.

Conclusion générale

Les travaux présentés dans ce mémoire de thèse traitent de la conception d'une métaheuristique hybride entre programmation dynamique et algorithme génétique et de son application à la production d'énergie. Dans cette partie, nous allons tout d'abord présenter les principales contributions issues de ces travaux. Puis, nous discuterons des perspectives d'amélioration et des directions de recherche qui pourraient faire suite à ce travail.

Synthèse des contributions

DYNAMOP : Dans ce manuscrit nous avons présenté DYNAMOP, une méthode originale utilisant un algorithme génétique pour guider la programmation dynamique. Cette approche présente de nombreux avantages.

Tout d'abord, la représentation génotypique des solutions sous forme de chemin d'un graphe d'états permet d'avoir une quasi séparabilité de la fonction d'évaluation par rapport aux gènes (états). En effet, la fonction d'évaluation correspond alors à la somme des valeurs des arcs. Or, cette séparabilité s'avère bénéfique car elle rend possible, grâce à une une procédure itérative, une évaluation plus rapide. Elle permet également de faciliter la proposition d'opérateurs évolutionnaires avec de bonnes qualités de localité et d'héritabilité phénotypique. Ainsi, nous avons proposé plusieurs opérateurs tels que le croisement k-transitions, la mutation de sous-chemins et la mutation par bifurcation.

De plus, cette représentation facilite la mise en place d'hybridations avec la programmation dynamique. Ces hybridations se font via des opérateurs évolutionnaires intelligents basés sur l'application de la programmation dynamique dans un graphe d'états restreint défini autour d'un ou plusieurs chemins issus de la population.

En outre, DYNAMOP est une méthode assez générique qui peut être appliquée à une large classe de problèmes.

Nous avons testé DYNAMOP sur deux problèmes d'application. D'une part, le problème d'affectation d'unités et d'autre part un problème de planification des débits d'eau dans un réseau hydro-électrique qui correspond à la modélisation d'un cas d'application réel. Pour chacun de ces deux problèmes, les résultats obtenus avec DYNAMOP sont très satisfaisants car nous avons observé que les capacités de DYNAMOP surpasse largement celles d'un algorithme génétique classique. Nous avons également étudié l'impact des opérateurs évolutionnaires intelligents proposés et ainsi montré leur intérêt. Pour le problème d'affectation d'unités, qui est un problème académique très étudié, nous avons pu comparer nos résultats aux solutions optimales qui sont connues, et constater que DYNAMOP parvient toujours à fournir une solution très proche de l'optimum. Il y a en effet, toujours moins de 0,025% de différence avec l'optimum global et pour les petits jeux de donnés l'optimum est atteint. Pour ce problème, nous avons également comparé DYNAMOP aux métaheuristiques les plus performantes que nous avons trouvées dans la littérature et nous avons pu constater que dans tous les cas DYNAMOP donne des résultats équivalents ou

meilleurs. Ainsi, nous pouvons dire que DYNAMOP constitue une méthode de résolution très efficace pour les problèmes étudiés.

MO-DYNAMOP : Nous avons ensuite proposé une généralisation de DYNAMOP pour la résolution de problèmes d'optimisation combinatoire multi-objectif. La méthode ainsi obtenue s'appelle MO-DYNAMOP, son principe et ses avantages sont similaires à ceux de DYNAMOP, mais les solutions sont représentées comme des chemins dans un graphe d'états multi-objectif et les opérateurs évolutionnaire intelligents utilisent la programmation dynamique multi-objectif.

Cette méthode a été testée avec succès sur le problème d'affectation d'unités multi-objectif. En effet, nous avons comparé, via une analyse statistique, MO-DYNAMOP à un algorithme évolutionnaire multi-objectif classique ainsi qu'à un algorithme plus sophistiqué, mCON [123], issu de la littérature. Cette étude nous a permis de montrer que MO-DYNAMOP est significativement plus performant que les deux autres approches. Nous avons également comparé les valeurs extrêmes trouvées par MO-DYNAMOP à celles données par un autre algorithme de la littérature, MOMAS [110], en utilisant les mêmes conditions en terme de taille de population et de critère de convergence, et, une fois de plus MO-DYNAMOP donne des résultats de meilleure qualité.

Représentation indirecte avec décodeur multi-objectif : Une autre contribution notable de ce manuscrit, réside dans la formulation et l'étude du problème posé par la représentation indirecte en optimisation multi-objectif dans le cas où le décodage des solutions partielles implique la résolution d'un sous-problème d'optimisation multi-objectif. L'idée est, comme en mono-objectif, de décrire le problème sous forme d'un problème hiérarchique à deux niveaux :

$$\begin{array}{c} \min\limits_{x,y}\{f_1(x,y), f_2(x,y)...f_m(x,y)\} \\ x \in X \\ y \in Y(x) \end{array} \tag{9.8}$$

tels que pour x fixé le sous problème Π_x :

$$\begin{array}{c} \min\limits_{y}\{f_1(x,y), f_2(x,y)...f_m(x,y)\} \\ y \in Y(x) \end{array} \tag{9.9}$$

soit solvable en un temps de calcul raisonnable par une méthode exacte ou une heuristique de qualité.

Puis il s'agit d'appliquer une métaheuristique manipulant des solutions partielles qui correspondent aux valeurs du vecteur x et d'utiliser un système de décodage basé sur une méthode exacte pour obtenir une solution complète (x, y). Cependant, en multi-objectif, une difficulté nouvelle se pose. En effet, pour une solution partielle x, on obtient un ensemble de solutions y Pareto équivalentes, car le problème de second niveau est également multi-objectif. Ainsi la problèmatique du décodage n'est pas évidente. Soit une unique solution y est associée à chaque solution partielle x, mais alors il faut réfléchir au critère de sélection de cette solution y parmi l'ensemble des solutions possibles qui sont Pareto optimales. Soit, pour chaque solution partielle x, l'ensemble des vecteurs y Pareto optimaux pour le problème de second niveau est associé à x, mais dans ce cas il est nécessaire d'adapter les processus d'attribution de valeur de fitness et/ou de diversité de la métaheuristique afin de prendre en compte cet ensemble.

Nous avons proposé et comparé, trois méthodes de décodage. La première est appelée méthode de scalarisation invariante. Elle consiste à transformer le problème de second

niveau, traité par le décodeur, en un problème mono-objectif en utilisant une somme pondérée des objectifs dont les coefficients de pondération sont fixés a priori par l'utilisateur. Ce traitement a priori est un inconvénient notable, car nous souhaitons proposer des approches dans lesquels le décideur n'intervient qu'en fin de processus, c'est-à-dire des méthodes a postériori. De plus, le choix des coefficients de pondération ne peut pas être arbitraire car l'ensemble des solutions Pareto optimales qui sont accessibles à la métaheuristique dépendent de ce choix.

La seconde méthode proposée est appelée méthode de scalarisation embarquée. Dans cette méthode, les coefficients de pondération utilisés lors du décodage sont inclus dans la représentation des solutions et peuvent être modifiés par les opérateurs de parcours de l'espace de recherche de la métaheuristique. Cependant, si le front Pareto du problème de second niveau n'est pas convexe, il reste possible que certaines solutions Pareto optimales ne soient pas accessibles à la métaheuristique.

La troisième méthode consiste à utiliser un décodeur multi-objectif, c'est à dire que nous ne cherchons plus à associer une unique solution complète à chaque solution partielle, mais, à l'inverse, le front des solutions Pareto optimales pour le problème de second niveau est associé à la solution partielle correspondante. Pour calculer ce front n'importe quelle méthode peut être utilisée. Cependant, nous avons dû mettre en place une méthode d'assignation de valeur de fitness et de diversité aux solutions partielles qui prenne en compte ce décodage multiple. Les techniques d'assignation que nous avons proposées sont génériques et elles peuvent s'adapter à n'importe quel type de métaheuristique. Nous les avons testées sur IBEA, NSGAII et SMS-EMOA.

Une analyse basée sur des tests de comparaison statistique, nous a permis de montrer la supériorité de la troisième approche sur les deux autres, c'est à dire de l'utilisation d'un décodeur multi-objectif.

La méthode de scalarisation embarquée présente un avantage significatif par rapport à celle utilisant la scalarisation invariante. Mais les valeurs des coefficients de pondération ont tendance à converger trop fortement ce qui rend cette méthode bien moins efficace que celle basée sur l'utilisation d'un décodage multiple.

Perspectives

Dans cette section nous présentons les améliorations qui peuvent être apportées à notre travail ainsi que les perspectives de recherche pouvant faire suite à cette thèse.

Expérimentations : Pour compléter ce travail, certaines expérimentations pourraient être faites. Tout d'abord, il serait intéressant de tester l'influence des opérateurs évolutionnaires intelligents utilisés par MO-DYNAMOP pour résoudre la version multi-objectif du problème d'affectation d'unités, comme cela a été fait pour la version mono-objectif du problème.

Ensuite, afin de mieux comprendre l'impact de la représentation et des opérateurs évolutionnaires utilisés par DYNAMOP, il serait intéressant de faire une analyse de comparaison statistique entre DYNAMOP et une version hybride de l'algorithme génétique basique BGA. Dans cette version hybride, les opérateurs de mutations basés sur la programmation dynamique seraient utilisés en plus des opérateurs évolutionnaires classiques.

Enfin, dans le chapitre 3, plusieurs types d'opérateurs de mutation et de croisement hybrides ou non ont été proposés. Ces opérateurs n'ont pas tous été testés lors des applications et il pourrait donc être intéressant de les implémenter et d'effectuer une étude de comparaison.

Amélioration du traitement des cas d'application : Concernant le problème d'affectation d'unités, une perspective de travail prometteuse serait la prise en compte de l'aspect stochastique, tout aussi bien en ce qui concerne la demande, qu'en ce qui concerne les capacités de production des unités. En effet, dans beaucoup de cas pratiques ces paramètres doivent être modélisés de façon stochastique en fonction de paramètres externes au problème. Par exemple, en cas de variation extrême de la température, la demande en électricité peut être plus forte que prévue pour répondre au besoin de chauffage ou de climatisation. De plus, pour certains type de système de production, les capacités de production peuvent varier de façon non déterministe, c'est par exemple le cas pour les systèmes de production d'énergie hydraulique ou d'énergie éolienne. Nous avons d'ores et déjà effectué quelques travaux en ce sens.

En ce qui concerne le problème de planification des débits d'eau dans un réseau hydro-électrique, une première amélioration pourrait être de mettre en oeuvre une seconde mutation hybride impactant sur les débits de plusieurs réservoirs car la mutation hybride actuelle n'est pas très efficace pour les données associées à des réseaux hydro-électrique de grande taille.

Pour ce problème, il serait également utile d'affiner la modélisation pour la rendre plus réaliste et plus générique. Pour ce faire, il faudrait d'une part, étendre la méthode de résolution pour les réseaux hydro-électriques non-arborescents et ajouter la possibilité d'inclure des pompes dans le réseau. D'autre part, il faudrait ajouter certaines contraintes au modèle, permettant, par exemple, la prise en compte de variations maximales possibles des débits d'une période de temps à l'autre, de paliers de production intermédiaire, de durées minimales de fonctionnement et d'arrêt ainsi que de plages de fonctionnement constant.

Perspectives de recherche concernant DYNAMOP et MO-DYNAMOP : Dans cette thèse la partie hybride de DYNAMOP résidait dans l'utilisation d'une mutation basée sur l'application de la programmation dynamique dans un graphe restreint autour du chemin représentant la solution à muter. Néanmoins, d'autres idées d'hybridations pourraient être exploitées, notamment en proposant des croisements intelligents consistant à appliquer la programmation dynamique dans un graphe d'états restreint défini à partir d'une sous-population d'individus.

Nous pensons également qu'il serait intéressant d'appliquer DYNAMOP sur d'autres types de problèmes, afin de voir si cette approche peut donner de bons résultats sur des problèmes différents. Nous voudrions en particulier appliquer DYNAMOP à un problème de type *Just in Time Job Shop Scheduling* puis MO-DYNAMOP à une version multi-objectif de ce problème. Cela nous permettrait également de tester MO-DYNAMOP sur un cas d'application où le problème de la représentation indirecte ne se pose pas.

Enfin, nous pourrions facilement appliquer le principe de DYNAMOP aux autres paradigmes de métaheuristiques mono et multi-objectif. En effet le principe de la mutation proposé pour DYNAMOP peut être réutilisé pour définir un voisinage de solutions et ainsi étendre DYNAMOP à toutes les métaheuristiques basées sur des recherches locales.

Perspectives de recherche concernant le décodeur multi-objectif : Dans cette thèse nous avons étudié la problématique de la représentation indirecte en multi-objectif dans le cas où le décodage implique la résolution d'un problème multi-objectif. Nous pensons que cette problématique permettra de définir une ligne de recherche nouvelle et prometteuse.

En effet, nous sommes persuadés que cette problématique pourrait être utile à la résolution de problèmes autres que le problème d'affectation d'unité. Une de nos perspective de

travail est donc de l'appliquer à un autre problème, comme par exemple le problème de *Job Shop Scheduling* sur une machine avec pour objectif de trouver les solutions minimisant simultanément le makespan et l'écart aux délais.

Nous arrivons à la conclusion que les solutions que nous avons proposées peuvent faire l'objet d'améliorations importantes. Tout d'abord, concernant la méthode de décodage par scalarisation embarquée, il faudrait chercher des moyens de forcer la diversification dans la population des coefficients de pondérations attachés aux individus.

Ensuite, concernant le décodeur multi-objectif, nous avons, dans cette thèse, cherché à proposer des stratégies d'adaptation des processus d'assignation de valeurs de fitness et de diversité qui soient génériques, ceci afin de pouvoir réaliser une implémentation d'une solution directement utilisable avec tout type de métaheuristique. Ce travail nous permettra d'implémenter un paquetage pour Paradiseo donnant la possibilité à l'utilisateur de facilement mettre en place une stratégie de représentation indirecte en multi-objectif. Cependant nous pensons que ces stratégies sont améliorables, en particulier en cherchant à prendre en considération la qualité des solutions phénotypiques associées à une solution génotypique plutôt en tant qu'ensemble que simplement individuellement. En effet, nous avons pu voir l'intérêt potentiel d'une telle prise en compte lorsque nous avons comparé les deux versions de décodages pour IBEA : IBEAV1 et IBEAV2.

Bibliographie

[1] E.H.L. Aarts and J. K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] F. Abbattista, N. Abbattista, and L. Caponetti. An evolutionary and cooperative agents model for optimization. In *Evolutionary Computation, IEEE International Conference on*, volume 2, pages 668–671. IEEE, 1995.

[3] F. B. Abdelaziz, S. Krichen, and J. Chaouachi. A hybrid heuristic for multiobjective knapsack problems. In *Meta-heuristics*, pages 205–212. Springer, 1999.

[4] M. Abido. Environmental/economic power dispatch using multiobjective evolutionary algorithms. *Power Systems, IEEE Transactions on*, 18(4) :1529–1537, 2003.

[5] M. Abido. Multiobjective particle swarm optimization for environmental/economic dispatch problem. *Electric Power Systems Research*, 79(7) :1105–1113, 2009.

[6] C. W. Ahn and R. Ramakrishna. A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE Transactions on Evolutionary Computation*, 6, 2002.

[7] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1) :75–102, 2002.

[8] E. Alba. *Parallel metaheuristics : a new class of algorithms*, volume 47. John Wiley & Sons, 2005.

[9] E. Angel, E. Bampis, and L. Gourves. A dynasearch neighborhood for the bicriteria traveling salesman problem. In *Metaheuristics for Multiobjective Optimisation*, pages 153–176. Springer, 2004.

[10] D. Applegate, R. Bixby, W. Cook, and V. Chvátal. *On the solution of traveling salesman problems*. Rheinische Friedrich-Wilhelms-Universität Bonn, 1998.

[11] V. Armentano and J. Claudio. An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, 10(5) :463–481, 2004.

[12] P. Augerat, J. M. Belenguer, E. Benavent, A. Corbéran, and D. Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2) :546–557, 1998.

[13] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1 : Basic algorithms and operators*, volume 1. CRC Press, 2000.

[14] J. Bader and E. Zitzler. Hype : An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1) :45–76, 2011.

[15] H. Balci and J. Valenzuela. Scheduling electric power generators using particle swarm optimization combined with the lagrangian relaxation method. *International Journal of Applied Mathematics and Computer Science*, 14(3) :411–422, 2004.

[16] V. Barichard and J. Hao. Un algorithme hybride pour le problème de sac à dos multi-objectifs. *Huitiemes Journées Nationales sur la Résolution Pratique de Problemes NP-Complets JNPC'2002 Proceedings*, 2002.

[17] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W.P. Savelsbergh, and P. H. Vance. Branch-and-price : Column generation for solving huge integer programs. *Operations research*, 46(3) :316–329, 1998.

[18] M. Basseur, J. Lemesre, C. Dhaenens, and E. Talbi. Cooperation between branch and bound and evolutionary approaches to solve a bi-objective flow shop problem. In *Experimental and Efficient Algorithms*, pages 72–86. Springer, 2004.

[19] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[20] H.P. Benson. Existence of efficient solutions for vector maximization problems. *Journal of Optimization Theory and Applications*, 26(4) :569–580, 1978.

[21] R. Bent and P. Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4) :515–530, 2004.

[22] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA : Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3) :1653–1669, 2007.

[23] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa : A platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 494–508, 2003.

[24] S. Boivin. *Résolution d'un problème de satisfaction de contraintes pour l'ordonnancement d'une chaîne d'assemblage automobile*. PhD thesis, Université du Québec à Montréal Montréal, 2005.

[25] R.M. Burns and C.A. Gibson. Optimization of priority lists for a unit commitment program. *IEEE Transactions on power apparatus and systems*, 94(6) :1917–1917, 1975.

[26] J.P.S. Catalao, S.J.P.S. Mariano, V.M.F. Mendes, and L.A.F.M. Ferreira. A practical approach for profit-based unit commitment with emission limitations. *International journal of electrical power & energy systems*, 32(3) :218–224, 2010.

[27] K. Chandrasekaran, S. Hemamalini, S. Simon, and N.P. Padhy. Thermal unit commitment using binary/real coded artificial bee colony algorithm. *Electric Power Systems Research*, 84(1) :109–119, 2012.

[28] G. Chang, M. Aganagic, J. Waight, J. Medina, T. Burton, S. Reeves, and M. Christoforidis. Experiences with mixed integer linear programming based approaches on short-term hydro scheduling. *Power Systems, IEEE Transactions on*, 16(4) :743–749, 2001.

[29] P. Chen. Two-level hierarchical approach to unit commitment using expert system and elite PSO. *IEEE Transactions on Power Systems*, 27(2) :780–789, 2012.

[30] C.P. Cheng, C-W. Liu, and C-C. Liu. Unit commitment by annealing-genetic algorithm. *International Journal of Electrical Power & Energy Systems*, 24(2) :149–158, 2002.

[31] C. Coello Coello and M. Lechuga. MOPSO : A proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1051–1056. IEEE, 2002.

[32] A. Cohen and M. Yoshimura. A branch-and-bound algorithm for unit commitment. *IEEE Trans. Power Appar. Syst. ;(United States)*, 102(2), 1983.

[33] R. K. Congram, C. N. Potts, and S. L. Van De Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14 :52–67, 1998.

[34] C. Cotta and J. M. Troya. Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18(2) :137–153, 2003.

[35] P. Czyżżak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1) :34–47, 1998.

[36] I. G. Damousis, A. G. Bakirtzis, and P. S. Dokopoulos. A solution to the unit-commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power Systems*, 19(2) :1165–1172, 2004.

[37] C. Darwin. On the origin of species by means of natural selection london. *J. Murray*, 1859.

[38] S. Dasgupta and C. Papadimitriou. *Algorithms*. McGraw-Hill Science/Engineering/Math, 2006.

[39] L. N. De Castro and J. Timmis. *Artificial immune systems : a new computational intelligence approach*. Springer Science & Business Media, 2002.

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, 2002.

[41] K. Doerner, W. Gutjahr, R. Hartl, C. Strauss, and C. Stummer. Ant colony optimization in multiobjective portfolio selection. In *Proc. 4th Metaheuristics International Conference, Porto, Portugal*, pages 243–248, 2001.

[42] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B : IEEE Transactions on Cybernetics*, 26(1) :29–41, 1996.

[43] T. Dunker, G. Radons, and E. Westkämper. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research*, 165(1) :55–69, 2005.

[44] M. Ehrgott. *Multicriteria optimization*, volume 2. Springer, 2005.

[45] M. Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[46] P. Engrand. A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management. Technical report, Electricite de France, 92-Clamart (France), 1998.

[47] R.W. Ferrero, J.F. Rivera, and S.M. Shahidehpour. A dynamic programming two-stage algorithm for long-term hydrothermal scheduling of multireservoir systems. *IEEE Transaction on Power System*, 13(4) :1534–1540, 1998.

[48] C. Fleurent and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. *Quadratic assignment and related problems*, 16 :173–187, 1994.

[49] C. M. Fonseca and P. J. Fleming. Multiobjective genetic algorithms. In *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pages 6–1. IET, 1993.

[50] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *2012 IEEE Symposium on Security and Privacy*, pages 202–214. IEEE Computer Society, 1994.

[51] P.E.C. Franco, M.F. Carvalho, and S. Soares. A network flow model for short-term hydro-dominated hydrothermal scheduling problems. *IEEE Transactions on Power Systems*, 9(2) :1016–1022, 1994.

[52] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3) :491–511, 2006.

[53] Z. Gaing. Discrete particle swarm optimization algorithm for unit commitment. In *Power Engineering Society General Meeting, 2003, IEEE*, volume 1. IEEE, 2003.

[54] X. Gandibleux and A. Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem : the two objectives case. *Journal of Heuristics*, 6(3) :361–383, 2000.

[55] X. Gandibleux, N. Mezdaoui, and A. Fréville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In *Advances in multiple objective and goal programming*, pages 291–300. Springer, 1997.

[56] X. Gandibleux, H. Morita, and N. Katoh. The supported solutions used as a genetic information in a population heuristic. In *Evolutionary Multi-Criterion Optimization*, pages 429–442. Springer, 2001.

[57] J. Garcia-Gonzalez and G. Castro. Short-term hydro scheduling with cascaded and head-dependent reservoirs based on mixed-integer linear programming. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 3, pages 6–pp. IEEE, 2001.

[58] T. Gjengedal. Emission constrained unit-commitment. *IEEE Transactions on Energy Conversion*, 11(1) :132–138, 1996.

[59] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5) :533–549, 1986.

[60] D. E. Goldberg. Genetic algorithm in search. *Optimization and Machine Learning*, 1989.

[61] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications : Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ : Lawrence Erlbaum, 1987.

[62] M. Gravel, W. Price, and C. Gagné. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, 143(1) :218–229, 2002.

[63] X. Guan, Q. Zhai, and A. Papalexopoulos. Optimization based methods for unit commitment : Lagrangian relaxation versus general mixed integer programming. In *Power Engineering Society General Meeting, 2003, IEEE*, volume 2. IEEE, 2003.

[64] Y.Y. Haimes, L.S. Lasdon, and D.A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems Man and Cybernetics*, 1 :296–297, 1971.

[65] W.A. Hall and Buras. The dynamic programming approach to water resources development. *Journal of Geophysical Research*, 66 :517–520, 1961.

[66] M. Hammami and K. Ghédira. Cosats, X-COSATS : Two multi-agent systems cooperating simulated annealing, tabu search and x-over operator for the k-graph partitioning problem. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 647–653. Springer, 2005.

[67] M. P. Hansen. Tabu search for multiobjective optimization : MOTS. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, pages 574–586. Citeseer, 1997.

[68] K. Hara, M. Kimura, and N. Honda. A method for planning economic unit commitment and maintenance of thermal power systems. *IEEE Transactions on Power Apparatus and Systems*, 5 :427–436, 1966.

[69] M. Heidari, V. Chow, P. Kokotović, and D. Meredith. Discrete differential dynamic programing approach to water resources systems optimization. *Water Resources Research*, 7(2) :273–282, 1971.

[70] M. Heidari, V. T. Chow, P. V. Kokotovifa, and Da. D. Meredith. Discrete differential dynamic programing approach to water resources systems optimization. *Water Resources Research*, 7(2) :273–282, 1971.

[71] S. Helbig and D. Pateva. On several concepts for epsilon-efficiency. *Operations-Research-Spektrum*, 16(3) :179–186, 1994.

[72] W. Hobbs, G. Hermon, S. Warner, and G.B. Shelbe. An enhanced dynamic programming approach for unit commitment. *IEEE Transactions on Power Systems*, 3(3) :1201–1205, 1988.

[73] J. H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* U. Michigan Press, 1975.

[74] C. Huang, H.r Yang, and C. Huang. Bi-objective power dispatch using fuzzy satisfaction-maximizing decision approach. *IEEE Transactions on Power Systems*, 12(4) :1715–1721, 1997.

[75] K. Huang, H. Yang, and C. Huang. A new thermal unit commitment approach using constraint logic programming. In *20th International Conference on Power Industry Computer Applications.*, pages 176–185. IEEE, 1997.

[76] T. Ikeguchi, M. Hasegawa, T. Kimura, T. Matsuura, and K. Aihara. Theory and applications of chaotic optimization methods. In *Innovative Computing Methods and Their Applications to Engineering Problems*, pages 131–161. Springer, 2011.

[77] S. Jacquin, L. Jourdan, and E. Talbi. Dynamic programming based metaheuristic for energy planning problems. In *Applications of Evolutionary Computation*, volume LNCS 8602, pages 165–176. Springer, 2014.

[78] S. Jacquin, L. Jourdan, and E. Talbi. Dynamic programming based metaheuristic for the unit commitment problem. In *The 5th international conference on metaheuristic ant nature inspired computing, META'2014*, 2014.

[79] S. Jacquin, L. Jourdan, and E. Talbi. Dynamop applied to the unit commitment problem. In *Learning and Intelligent OptimizatioN Conference LION 9*, volume LNCS 8994, 2015.

[80] S. Jacquin, L. Jourdan, and E. Talbi. A multi-objective dynamic programming based metaheuristic. *The XI metaheuristic International Conference*, 11, 2015.

[81] S. Jacquin, L. Mousin, I. Machado, E. Talbi, and L. Jourdan. A comparison of decoding strategies for the 0/1 multi-objective unit commitment problem. In *Evolutionary Multi-Criterion Optimization*, volume LNCS 9018, pages 381–395. Springer, 2015.

[82] A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131(1-4) :135–158, 2004.

[83] Y. Jeong, J. Park, S. Jang, and K. Y. Lee. A new quantum-inspired binary pso : application to unit commitment problems for power systems. *IEEE Transactions on Power Systems*, 25(3) :1486–1495, 2010.

[84] Y. Jeong, J. Park, J. Shin, and K. Lee. A thermal unit commitment approach using an improved quantum evolutionary algorithm. *Electric Power Components and Systems*, 37(7) :770–786, 2009.

[85] D. R. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In *ICGA*, pages 442–449, 1991.

[86] L. Jourdan, M. Basseur, and E-G Talbi. Hybridizing exact methods and metaheuristics : A taxonomy. *European Journal of Operational Research*, 199(3) :620–629, 2009.

[87] N. Jozefowiez. *Modélisation et résolution approchée de problèmes de tournées multi-objectif*. PhD thesis, Lille 1, 2004.

[88] N. Jozefowiez, F. Glover, and M. Laguna. A hybrid meta-heuristic for the traveling salesman problem with profits. Technical report, Technical report, Leeds School of Business, University of Colorado at Boulder, 2006.

[89] S. A. Kazarlis, A.G. Bakirtzis, and V. Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 11(1) :83–92, 1996.

[90] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[91] J.O. Kephart et al. A biologically inspired immune system for computers. In *Artificial Life IV : proceedings of the fourth international workshop on the synthesis and simulation of living systems*, pages 130–139, 1994.

[92] R.H. Kerr, J.L. Scheidt, A.J. Fontanna, and J.K. Wiley. Unit commitment. *IEEE Transactions on Power Apparatus and Systems*, 5 :417–421, 1966.

[93] S. Khan. A self-adaptive weighted sum technique for the joint optimization of performance and power consumption in data centers. In *ISCA PDCCS*, pages 13–18, 2009.

[94] I.Y. Kim and O.L. De Weck. Adaptive weighted-sum method for bi-objective optimization : Pareto front generation. *Structural and multidisciplinary optimization*, 29(2) :149–158, 2005.

[95] S. Kirkpatrick. Optimization by simulated annealing : Quantitative studies. *Journal of statistical physics*, 34(5-6) :975–986, 1984.

[96] R. Klötzler. Multiobjective dynamic programming. *Optimization*, 9(3) :423–426, 1978.

[97] J. Knowles and D. Corne. The pareto archived evolution strategy : A new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE, 1999.

[98] M. Krueger. *Méthode d'analyse d'algorithmes d'optimisation stochastiques à l'aide d'algorithmes génétiques, Paris, école nationale supérieure des télécommunications*. PhD thesis, 1994.

[99] S. Kuloor, G.S. Hope, and O.P. Malik. Environmentally constrained unit commitment. In *Generation, Transmission and Distribution, IEE Proceedings C*, volume 139, pages 122–128. IET, 1992.

[100] S. Kumar and R. Naresh. Efficient real coded genetic algorithm to solve the non-convex hydrothermal scheduling problem. *International Journal of Electrical Power & Energy Systems*, 29 :738–747, 2007.

[101] T.W. Lau, C.Y. Chung, K.P. Wong, T.S. Chung, and S.L. Ho. Quantum-inspired evolutionary algorithm approach for unit commitment. *IEEE Transactions on Power Systems*, 24(3) :1503–1512, 2009.

[102] G.S. Lauer, N.R. Sandell, D.P. Bertsekas, and T.A. Posbergh. Solution of large-scale optimal unit commitment problems. *IEEE Transactions on Power Apparatus and Systems*, (1) :79–86, 1982.

[103] M. Laumanns, L. Thiele, and E. Zitzler. An adaptive scheme to generate the pareto front based on the epsilon-constraint method. *Practical Approaches to Multi-Objective Optimization*, 4461, 2005.

[104] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3) :932–942, 2006.

[105] E. L. Lawler and D. E. Wood. Branch-and-bound methods : A survey. *Operations research*, 14(4) :699–719, 1966.

[106] F.N. Lee. Short-term thermal unit commitment-a new method. *IEEE Transactions on Power Systems*, 3(2) :421–428, 1988.

[107] A. Lew and H. Mauch. *Dynamic programming : A computational tool*, volume 38. Springer Science & Business Media, 2006.

[108] G. Li and R. Matthew. New approach for optimization of urban drainage systems. *Journal of Environmental Engineering*, 116(5) :927–944, 1990.

[109] Y. Li, J. Li, and D. Zhao. Dynasearch algorithms for solving time dependent traveling salesman problem. *Journal of Southwest Jiaotong University*, 2 :009, 2008.

[110] Y. Li, N. Pedroni, and E. Zio. A memetic evolutionary multi-objective optimization method for environmental power unit commitment. *IEEE Transactions on Power Systems*, 28(3) :2660–2669, 2013.

[111] A. Liefooghe, L. Jourdan, and E-G. Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization : Paradiseo-moeo. *European Journal of Operational Research*, 209(2) :104–112, 2011.

[112] F. Lin, C. Kao, and C. Hsu. Incorporating genetic algorithms into simulated annealing. In *Proceedings. International Symposium on Artificial Intelligence*, pages 290–297, 1991.

[113] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical report, IRIDIA, 2011.

[114] H. R. Lourenço, O. C. Martin, and T. Stutzle. Iterated local search. 2001.

[115] M. Madrigal and V. Quintana. An interior-point/cutting-plane method to solve unit commitment problems. In *Power Industry Computer Applications, 1999. PICA'99. Proceedings of the 21st 1999 IEEE International Conference*, pages 203–209. IEEE, 1999.

[116] A.H. Mantawy, Y. Abdel-Magid, and S. Selim. A simulated annealing algorithm for unit commitment. *IEEE Transactions on Power Systems*, 13(1) :197–204, 1998.

[117] A.H. Mantawy, Y. L. Abdel-Magid, and S. Selim. Unit commitment by tabu search. *IEE Proceedings-Generation, Transmission and Distribution*, 145(1) :56–64, 1998.

[118] A.H. Mantawy, S.A. Soliman, and M.E. El-Hawary. A new tabu search algorithm for the long-term hydro scheduling problem. In *Power Engineering 2002 Large Engineering Systems Conference on, LESCOPE 02*, pages 29–34. IEEE, 2002.

[119] A.H. Mantawy, S.A. Soliman, and M.E. El-Hawary. The long-term hydro-scheduling problem—a new algorithm. *Electric Power Systems Research*, 64(1) :67–72, 2003.

[120] F. Manzanedo, M.P. Donsion, and J.L. Castro. An evolution based algorithm for environmentally constrained thermal scheduling problems. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 2, pages 6–pp. IEEE, 2001.

[121] C. Mariano and E. Morales. MOAQ : An ant-q algorithm for multiple objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 894–901, 1999.

[122] C. Mariano and E. Morales. A multiple objective ant-q algorithm for the design of water distribution irrigation networks. *Instituto Mexicano de Tecnología del Agua, Technical Report HC-9904*, 1999.

[123] A. M. Marques de Moura Gomes Viana. *Metaheuristics for the Unit Commitment Problem The Constraint Oriented Neighbourhoods Search Strategy*. PhD thesis, Faculty of Engineering, University of Porto, 2004.

[124] O. C. Martin and S. W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63(1) :57–75, 1996.

[125] M. Masaharu, T. Yoshiaki, and S. Yoshiharu. A migration scheme for the genetic adaptative routing algorithm. *IEEE Intelligent Systems*, pages 2774–2779, 1998.

[126] V. Miranda, D. Srinivasan, and L.M. Proenca. Evolutionary computation in power systems. *International Journal of Electrical Power & Energy Systems*, 20 :89–98, 1998.

[127] J. Moore and R. Chapman. Application of particle swarm to multiobjective optimization. *Department of Computer Science and Software Engineering, Auburn University*, 1999.

[128] H. Morita, X. Gandibleux, and N. Katoh. Experimental feedback on biobjective permutation scheduling problems solved with a population heuristic. *Foundations of Computing and Decision Sciences*, 26(1) :23–50, 2001.

[129] T. Murata and H. Ishibuchi. MOGA : multi-objective genetic algorithms. In *IEEE International Conference on Evolutionary Computation*, volume 1, page 289. IEEE, 1995.

[130] Y. Mutsunori and I. Toshihide. The use of dynamic programming in genetic algorithms for permutation problems. *European Journal of Operational Research*, 92 :387–401, 1996.

[131] A. Nakib, E. Talbi, and A. Fuser. Hybrid metaheuristic for annual hydropower generation optimization. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 412–419. IEEE, 2014.

[132] R. Naresh and J. Sharma. Short term hydro scheduling using two-phase neural network. *International journal of electrical power & energy systems*, 24(7) :583–590, 2002.

[133] V. Nwana, K. Darby-Dowman, and G. Mitra. A co-operative parallel heuristic for mixed zero–one linear programming : Combining simulated annealing with branch and bound. *European journal of operational research*, 164(1) :12–23, 2005.

[134] W. Ongsakul and N. Petcharaks. Unit commitment by enhanced adaptive lagrangian relaxation. *IEEE Transactions on Power Systems*, 19(1) :620–628, 2004.

[135] S.O. Orero and M.R. Irving. A genetic algorithm modelling framework and solution technique for short term optimal hydrothermal scheduling. *Power Systems, IEEE Transactions on*, 13 :501–518, 1998.

[136] Z. Ouyang and S.M. Shahidehpour. A hybrid artificial neural network-dynamic programming approach to unit commitment. *Power Systems, IEEE Transactions on*, 7(1) :236–242, 1992.

[137] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1) :60–100, 1991.

[138] C.K. Pang and H.C. Chen. Optimal short-term thermal unit commitment. *IEEE Transactions on Power Apparatus and Systems*, 95(4) :1336–1346, 1976.

[139] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1998.

[140] Y. Park, J. Park, and J. Won. A hybrid genetic algorithm/ dynamic programming approach to optimal long-term generation expansion planning. *Elservier Science*, 20 :295–303, 1998.

[141] K. Parsopoulos and M. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607. ACM, 2002.

[142] V. T. Paschos. *Applications of combinatorial optimization*. John Wiley & Sons, 2013.

[143] M.R. Piekutowski, T. Litwinowicz, and R. Frowd. Optimal short-term scheduling for a large-scale cascaded hydro system. In *Power Industry Computer Application Conference, 1993. Conference Proceedings*, pages 292–298. IEEE, 1993.

[144] G.K. Purushothama and L. Jenkins. Simulated annealing with local search-a hybrid algorithm for unit commitment. *IEEE Transactions on Power Systems*, 18(1) :273–278, 2003.

[145] D. F. Rahman, A. Viana, and J. P. Pedroso. Metaheuristic search based methods for unit commitment. *International Journal of Electrical Power & Energy Systems*, 59 :14–22, 2014.

[146] L.A.C. Roque, D.B.M.M. Fontes, and F.A.C.C. Fontes. A hybrid biased random key genetic algorithm approach for the unit commitment problem. *Journal of Combinatorial Optimization*, 28(1) :140–166, 2014.

[147] Provas Kumar Roy. Solution of unit commitment problem using gravitational search algorithm. *International Journal of Electrical Power & Energy Systems*, 53 :85–94, 2013.

[148] M. Rui. A novel bi-objective fuzzy optimal model of short-term trade planning considering environmental protection and economic profit in deregulated power system. *Proceeding-chinese society of electrical engineering*, 22(4) :104–108, 2002.

[149] S. Salcedo-Sanz, Y. Xu, and X. Yao. Hybrid meta-heuristics algorithms for task assignment in heterogeneous computing systems. *Computers & operations research*, 33(3) :820–835, 2006.

[150] A Saramourtsis, J Damousis, A Bakirtzis, and P Dokopoulos. Genetic algorithm solution to the economic dispatch problem—application to the electrical power grid of crete island. In *Proc. Workshop Machine Learning Applications to Power Systems (ACAI)*, pages 308–317, 2001.

[151] H. Sasaki, M. Watanabe, J. Kubokawa, N. Yorino, and R. Yokoyama. A solution method of unit commitment by artificial neural networks. *IEEE Transactions on Power Systems*, 7(3) :974–981, 1992.

[152] T. Senjyu, H. Yamashiro, K. Uezato, and T. Funabashi. A unit commitment problem by using genetic algorithm based on unit characteristic classification. In *Power Engineering Society Winter Meeting, 2002. IEEE*, volume 1, pages 58–63. IEEE, 2002.

[153] P. Serafini. Simulated annealing for multi objective optimization problems. In *Multiple criteria decision making*, pages 283–292. Springer, 1994.

[154] K. Shahookar and P. Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(5) :500–511, 1990.

[155] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*, pages 417–431. Springer, 1998.

[156] Z.K. Shawwash, T. K. Siu, and S. Russell. The bc hydro short term hydro scheduling optimization model. *IEEE Transactions on Power Systems*, 15(3) :1125–1131, 2000.

[157] R. Shoults, S. K. Chang, S. Helmick, and W. Grady. A practical approach to unit commitment, economic dispatch and savings allocation for multiple-area pool operation with import/export constraints. *IEEE Transactions on Power Apparatus and Systems*, (2) :625–635, 1980.

[158] S. P. Simon, N. P. Padhy, and R.S. Anand. An ant colony system approach for unit commitment problem. *International Journal of Electrical Power & Energy Systems*, 28(5) :315–323, 2006.

[159] D.N. Simopoulos, S.D. Kavatza, and C. D. Vournas. Unit commitment by an enhanced simulated annealing algorithm. *IEEE Transactions on Power Systems*, 21(1) :68–76, 2006.

[160] N.S. Sisworahardjo and A.A. El-Keib. Unit commitment using the ant colony search algorithm. In *Power Engineering 2002 Large Engineering Systems Conference on, LESCOPE 02*, pages 2–6. IEEE, 2002.

[161] M. Sniedoviech and S. Voss. The corridor method : a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35 :551–578, 2006.

[162] F. Sourd. Dynasearch for the earliness–tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, 34(5) :591–598, 2006.

[163] F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework : Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, 20(3) :472–484, 2008.

[164] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3) :221–248, 1994.

[165] D. Srinivasan and A. Tettamanzi. An evolutionary algorithm for evaluation of emission compliance options in view of the clean air act amendments. *IEEE Transactions on Power Systems*, 12(1) :336–341, 1997.

[166] C. Su and Y. Hsu. Fuzzy dynamic programming : an application to unit commitment. *IEEE Transactions on Power Systems*, 6(3) :1231–1237, 1991.

[167] T. Sum-Im and W. Ongsakul. Ant colony search algorithm for unit commitment. In *IEEE International Conference on Industrial Technology*, volume 1, pages 72–77. IEEE, 2003.

[168] K.S. Swarup and S. Yamashiro. Unit commitment solution methodology using genetic algorithm. *IEEE Transactions on Power Systems*, 17(1) :87–91, 2002.

[169] É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8) :661–673, 1993.

[170] E. Taillard and S. Voss. Popmusic—partial optimization metaheuristic under special intensification conditions. In *Essays and surveys in metaheuristics*, pages 613–629. Springer, 2002.

[171] S. Takriti and J. Birge. Using integer programming to refine lagrangian-based unit commitment solutions. *IEEE Transactions on Power Systems*, 15(1) :151–156, 2000.

[172] S. Takriti, J. R. Birge, and E. Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3) :1497–1508, 1996.

[173] J.H. Talaq, F. El-Hawary, and M.E. El-Hawary. A summary of environmental/economic dispatch algorithms. *IEEE Transactions on Power Systems*, 9(3) :1508–1516, 1994.

[174] E. Talbi. *Hybrid metaheuristics*. Springer, 2013.

[175] E-G. Talbi. *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons, 2009.

[176] K. C. Tan, Y.H. Chew, and L. H. Lee. A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172(3) :855–885, 2006.

[177] K.C. Tan, Y.H. Chew, and L.H. Lee. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34(1) :115–151, 2006.

[178] R. Tanese. Parallel genetic algorithm for a hypercube. In *Genetic algorithms and their applications : proceedings of the second International Conference on Genetic Algorithms : July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, 1987.

[179] W. H. Tang, B. C. Yen, and L. W. Mays. Optimal risk-based design of storm sewer networks. *Journal of the Environmental Engineering Division*, 101(3) :381–398, 1975.

[180] R Core Team. R language definition, 2000.

[181] V. T'kindt, N. Monmarché, F. Tercinet, and D. Laügt. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, 142(2) :250–257, 2002.

[182] J. Tospornsampan, I. Kita, M. Ishii, and Y. Kitamura. Optimization of a multiple reservoir system operation using a combination of genetic algorithm and discrete differential dynamic programming : a case study in mae klong system, thailand. *Paddy and Water Environment*, 3(1) :29–38, 2005.

[183] A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl. Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, 2015.

[184] E.L. Ulungu, J. Teghem, P.H. Fortemps, and D. Tuyttens. Mosa method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4) :221–236, 1999.

[185] A. Uyar and B. Turkay. Evolutionnaary algorithms for the unit commitment problem. *Turkish Journal of Electrical Enginering*, 2008.

[186] J. Valenzuela and A. Smith. A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, 8(2) :173–195, 2002.

[187] J. Valenzuela and A. Smith. A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, 8(2) :173–195, 2002.

[188] G. Venturini. Algorithmes génétiques et apprentissage. *Revue d'intelligence artificielle*, 10(2-3) :345–387, 1996.

[189] S. Verdu and H.V. Poor. Backward, forward and backward-forward dynamic programming models under commutativity conditions. In *23rd Conference on Decision and Control, Las Vegas, December*, 1984.

[190] A. Viana and J. P. Pedroso. A new milp-based approach for unit commitment in power production planning. *International Journal of Electrical Power & Energy Systems*, 44(1) :997–1005, 2013.

[191] S. Virmani, E. C. Adrian, K. Imhof, and S. Mukherjee. Implementation of a lagrangian relaxation based unit commitment problem. *IEEE Transactions on Power Systems*, 4(4) :1373–1380, 1989.

[192] M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12(2) :139–155, 1998.

[193] S.J. Wang, S.M. Shahidehpour, D.S. Kirschen, S. Mokhtari, and G.D. Irisarri. Short-term generation scheduling with transmission and environmental constraints using an augmented lagrangian relaxation. *IEEE Transactions on Power Systems*, 10(3) :1294–1301, 1995.

[194] R. Wardlaw and M. Sharif. Evaluation of genetic algorithms for optimal reservoir system operation. *Journal of Water Resources Planning and Management*, 125 :25–33, 1999.

[195] R. Wardlaw and M. Sharif. Multireservoir systems optimization using genetic algorithms : Case study. *Journal of Computing in Civil Engineering*, 14 :255–263, 2000.

[196] L. D. Whitley. The genitor algorithm and selection pressure : Why rank-based allocation of reproductive trials is best. In *ICGA*, volume 89, pages 116–123, 1989.

[197] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[198] S. Yakowitz. Dynamic programming applications in water resources. *Water Resources Research*, 18 :673–696, 1983.

[199] L. Yang, E. S. Fraga, and L. G. Papageorgiou. Mathematical programming formulations for non-smooth and non-convex electricity dispatch problems. *Electric Power Systems Research*, 95 :302–308, 2013.

[200] B. Yu, X. Yuan, and J. Wang. Short-term hydro-thermal scheduling using particle swarm optimization method. *Energy Conversion and Management*, 48(7) :1902–1908, 2007.

[201] X. Yuan, H. Nie, A. Su, L. Wang, and Y. Yuan. An improved binary particle swarm optimization for unit commitment problem. *Expert Systems with applications*, 36(4) :8049–8055, 2009.

[202] X. Yuan, Y. Yuan, and Y. Zhang. A hybrid chaotic genetic algorithm for short-term hydro system scheduling. *Mathematics and Computers in Simulation*, 59(4) :319–327, 2002.

[203] X. Zhang, J. Zhao, and X. Chen. Multi-objective unit commitment fuzzy modeling and optimization for energy-saving and emission reduction [j]. *Proceedings of the CSEE*, 22 :71–76, 2010.

[204] B. Zhao, C.X. Guo, B.R. Bai, and Y.J. Cao. An improved particle swarm optimization algorithm for unit commitment. *International Journal of Electrical Power & Energy Systems*, 28(7) :482–490, 2006.

[205] F. Zhuang and F.D. Galiana. Unit commitment by simulated annealing. *IEEE Transactions on Power Systems*, 5(1) :311–318, 1990.

[206] E. Zio, P. Baraldi, and N. Pedroni. Optimal power system generation scheduling by multi-objective genetic algorithms with preferences. *Reliability Engineering & System Safety*, 94(2) :432–444, 2009.

[207] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer, 2004.

[208] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation*, pages 3–37. Springer, 2004.

[209] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 : Improving the strength pareto evolutionary algorithm, 2001.

[210] E. Zitzler and L. Thiele. *An evolutionary algorithm for multiobjective optimization : The strength pareto approach*, volume 43. 1998.

[211] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE transactions on evolutionary computation*, 3(4) :257–271, 1999.

[212] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers : an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2) :117–132, 2003.

[213] C.E. Zoumas, A.G. Bakirtzis, J.B. Theocharis, and V. Petridis. A genetic algorithm solution approach to the hydrothermal coordination problem. *IEEE Transactions on Power Systems*, 19 :1356–1364, 2004.