Analyse et conception de recherches locales génériques pour l'optimisation combinatoire à un ou plusieurs objectifs

Matthieu Basseur

UNIVERSITÉ D'ANGERS

U.F.R. SCIENCES

Mémoire d'habilitation à diriger des recherches

DISCIPLINE : INFORMATIQUE

présentée et soutenue publiquement,

le 03/12/2014, par

MATTHIEU BASSEUR

Analyse et conception de recherches locales génériques pour l'optimisation combinatoire à un ou plusieurs objectifs

Composition du jury :

ALEXANDRE CAMINADA, Professeur, Université de Belfort-Montbéliard, rapporteur
JIN-KAO HAO, Professeur, Université d'Angers, garant
NOUREDINE MELAB, Professeur, Université de Lille 1, rapporteur
PATRICE PERNY, Professeur, Université de Paris 6, rapporteur
FRÉDÉRIC SAUBION, Professeur, Université d'Angers, examinateur
VINCENT T'KINDT, Professeur, Université de Tours, examinateur, président de Jury

Table des matières

Chapitre 1

Introduction générale

1.1 Parcours

De 2001 à 2005, j'ai préparé mon doctorat au sein du Laboratoire d'Informatique Fondamentale de Lille (LIFL) dans l'équipe Optimisation PArallèle Coopérative (OPAC) dirigée par le Professeur El-Ghazali Talbi. Cette équipe fait partie de l'INRIA Futurs, projet DOLPHIN. Mes travaux de recherche s'inscrivaient dans le cadre de l'optimisation combinatoire multi-objectif. Ma thèse a été soutenue le 21 Juin 2005 et s'intitulait "Conception d'algorithmes coopératifs pour l'optimisation multiobjectif : Application aux problèmes d'ordonnancement de type flow shop".

Fin 2005 j'ai effectué un séjour post-doctoral à l'école polytechnique fédérale de Zurich (Suisse), au sein de l'équipe de Lothar Thiele en collaboration avec Eckart Zitzler. Pendant cette période, j'ai travaillé sur la conception de métaheuristiques multiobjectif dans le cadre de l'optimisation sous incertitude.

De 2006 à 2007, j'ai réalisé un post-doctorat à l'université de Nottingham, dans l'équipe *Automated Scheduling, Optimisation and Planning*, dirigée à l'époque par Edmund K. Burke, en tant que *research associate*. Ce post-doctorat s'inscrivait dans le cadre du projet "Next Generation Decision Support : Automating the Heuristic Design Process", qui s'intégrait parfaitement dans les orientations que prenaient alors mes recherches.

Enfin, depuis septembre 2007, je suis maître de conférences dans l'équipe Métaheuristiques, Optimisation et Applications (MOA) du Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA). Dans les travaux réalisés depuis mon recrutement en tant que maître de conférences, je me suis intéressé principalement à la conception de méthodes d'optimisation génériques, en m'abstrayant au maximum des problèmes étudiés. L'objectif est, d'une part, la compréhension du comportement des métaheuristiques et d'autre part, la conception de métaheuristiques autonomes, peu sensibles au paramétrage et efficace sur un panel de problèmes d'optimisation combinatoire le plus large possible.

1.2 Activités de recherche

Depuis mes débuts dans la recherche, la majeure partie de mes travaux a toujours porté sur les métaheuristiques pour l'optimisation multiobjectif. Je me place dans un contexte a posteriori, c'est-à-dire que l'on cherche à fournir à un décideur un ensemble de solutions de compromis d'un problème combinatoire multiobjectif, sans avoir connaissance de préférences éventuelles du décideur.

Depuis mon recrutement, je me concentre essentiellement sur les métaheuristiques multiobjectif de type recherches locales itérées. Différentes méthodes de prise en compte des objectifs ont été envisagées. J'ai étudié dans un premier temps les approches Pareto qui sont très classiques depuis la fin des années 90. En particulier plusieurs algorithmes de type PLS (Pareto Local Search) ont été mis au point et éprouvés sur différents problèmes. Je me suis ensuite beaucoup intéressé aux approches de résolution basées sur les indicateurs de qualité. Cela consiste à définir un opérateur évaluant la qualité d'une configuration par rapport à un ensemble de configurations sous forme de scalaire. L'algorithme IBMOLS a été proposé, puis mes travaux se sont particulièrement orientés vers l'indicateur d'hypervolume donnant naissance en particulier à l'algorithme HBMOLS.

Je me suis également intéressé aux mécanismes d'initialisation des solutions dans un cadre multiobjectif. Plusieurs méthodes d'initialisations ont été envisagées : le classique bruit aléatoire, mais aussi la recombinaison de solutions de compromis ainsi que la génération de chemin entre solutions (*path-relinking*). Cette dernière méthode s'est révélée particulièrement efficace lorsqu'elle est combinée avec une recherche de type HBMOLS.

Ces activités dans le domaine de l'optimisation multiobjectif m'ont conduit naturellement à participer régulièrement au groupe de travail PM2O (Programmation Mathématique MultiObjectif) du GdR RO. Je suis ainsi devenu le coordonnateur de ce groupe avec Laëtitia Jourdan (INRIA Lille) et Nicolas Jozefowiez (LAAS Toulouse) de 2008 à 2013. En 2013, nous avons décidé de dissoudre ce groupe de recherche et de créer dans la continuité le groupe de travail ATOM (Applications et Théorie de l'optimisation Multiobjectif), également soutenu par de GdR RO. Je coordonne ce groupe en collaboration avec Laëtitia Jourdan et Thibaut Lust (LIP6, Paris). PM2O, puis maintenant ATOM, a permis de créer une émulation entre les chercheurs intéressés par l'optimisation multiobjectif en général.

Dans le cadre de mes activités en optimisation multiobjectif, j'ai été co-porteur du projet OCMRE (2011-2012) « Optimisation combinatoire multiobjectif par recherche d'ensembles ». Ce projet m'a permis de travailler avec Arnaud Liefooghe (INRIA Lille), Sébastien Verel (Université de Calais) et Adrien Goëffon (LERIA, Angers). J'ai pu également co-encadrer les thèses de Ron-Qiang Zeng de 2010 à 2013 et de Brahim Chabane depuis fin 2013 ainsi que les stages de master recherche d'Arnaud Liefooghe en 2006 et d'Arthur Chambon en 2013.

Depuis trois ans, mon envie de comprendre plus précisément le comportement des métaheuristiques m'a poussé à m'intéresser davantage à l'optimisation mono-objectif, qui est un cadre beaucoup plus accessible pour ce genre d'étude. Je collabore essentiellement avec Adrien Goëffon sur cet aspect qui a pris de plus en plus d'importance dans mes recherches. Ainsi, nous avons beaucoup de travaux en cours dans ce domaine. En particulier nous cherchons à déterminer les facteurs permettant de guider les métaheuristiques directement vers des optima locaux de bonne qualité, sans l'intervention de mécanismes de diversification de la recherche, très largement utilisés dans la littérature. Pour cela, nous nous intéressons à l'étude des paysages de recherche ainsi

qu'au développement de nouvelles approches pour les explorer.

Bien que mon intérêt pour l'optimisation mono-objectif soit très récent, cela m'a amené rapidement à encadrer plusieurs stages de master recherche sur ce domaine : Pierre Desport et Vincent Vigneron en 2013, puis Hugo Traverson en 2014. Depuis juillet 2014, je suis également porteur du projet PGMO (Programme Gaspard Monge pour l'Optimisation) intitulé "Vers la conception de recherches locales efficaces basées sur l'étude des paysage de recherche".

Mes activités de recherche s'articulent aujourd'hui autour de deux axes principaux. Premièrement, l'optimisation multiobjectif : c'est historiquement mon domaine de recherche de prédilection qui concerne la totalité des travaux réalisés jusqu'à mon recrutement fin 2007 (thèse et post-doctorats). J'essaie depuis de développer ce domaine au sein du LERIA, en partie par l'intermédiaire de collaborations avec d'anciens co-auteurs (par exemple pour le projet OCMRE), et par l'intermédiaire d'encadrements. Mon deuxième axe de recherche concerne l'analyse et la résolution de paysages de recherche : cet aspect qui a pris de plus en plus d'importance dans mes recherches. Le projet PGMO que je porte se concentre sur cet aspect et j'encadre régulièrement des étudiants sur ce sujet. Je souhaite à long terme approfondir au maximum ce sujet.

Ci-dessous se trouve la liste de mes co-auteurs :
- Enrique Alba (Université de Malaga, Espagne)
- Edmund K. Burke (Université de Nottingham, Angleterre / Université de Stirling, Écosse)
- Clarisse Dhaenens (Université de Lille 1, France)
- Adrien Goëffon (Université d'Angers, France)
- Jin-Kao Hao (Université d'Angers, France)
- Jérémie Humeau (Université de Lille 1, France / École Nationale Supérieure des Mines de Douai, France)
- Laëtitia Jourdan (Université de Lille 1, France)
- Frédéric Lardeux (Université d'Angers, France)
- Khoi Le (Université de Nottingham, Angleterre / Université de Hanoï, Vietnam)
- Julien Lemesre (Université de Lille 1, France)
- Arnaud Liefooghe (Université de Lille 1, France)
- Antonio Nebro (Université de Malaga, Espagne)
- Frédéric Saubion (Université d'Angers, France)
- Franck Seynhaeve (Université de Lille 1, France)
- El-Ghazali Talbi (Université de Lille 1, France)
- Sébastien Vérel (Université de Nice, France / Université de Calais, France)
- Vincent Vigneron (Université d'Angers, France)
- Ron-Qiang Zeng (Université d'Angers, France / Université de Chengdu, Chine)
- Eckart Zitzler (École Polytechnique Fédérale de Zurich, Suisse / Université de Bern, Suisse)

1.3 Introduction

Les métaheuristiques sont des méthodes génériques de résolution approchée des problèmes d'optimisation difficiles. Elles sont utilisées lorsque les méthodes de résolution exactes sont inapplicables, car nécessitant un coût calculatoire ou de stockage déraisonnable. Même si le terme métaheuristique est apparu récemment [Glover, 1986], les premières métaheuristiques ont été proposées antérieurement : [Rechenberg, 1965] pour les stratégies d'évolution, [Fogel *et al.*, 1966] pour

la programmation évolutionnaire et [Holland, 1975] pour les algorithmes génétiques. D'autres types de métaheuristiques ont été proposés ultérieurement, parmis lesquels on peut citer le recuit simulé [Kirkpatrick *et al.*, 1983], la recherche Tabu [Glover, 1986] ou les colonies de fourmis [Dorigo, 1992]. Les métaheuristiques ont ensuite connu un essor remarquable, en partie dû à la popularité des algorithmes génétiques, marquée par la publication de *Genetic algorithms in search, optimization, and machine learning* [Goldberg *et al.*, 1989], livre cité plus de 65000 fois (chiffres de 2014).

Depuis le milieu des années 90, le nombre de *nouvelles* métaheuristiques a explosé, proposant pour la plupart des métaphores de phénomènes naturels. En particulier, un grand nombre d'entre elles s'inspirent du comportement de diverses espèces d'insectes, tandis que d'autres se basent sur différents phénomènes naturels comme par exemple l'écoulement de l'eau, l'harmonie entre musiciens, la compétition entre empires ou l'évolution des galaxies. D'une manière générale, il parait souvent possible d'adapter un système évolutif réel et de s'en servir comme prétexte pour *inventer* une nouvelle classe de métaheuristiques. La conséquence de cette tendance est que ces nouvelles métaheuristiques n'apportent que très rarement de nouveaux résultats théoriques et se résument très souvent à réutiliser des concepts existants en introduisant de nouveaux termes en fonction de la métaphore considérée. Dans [Sörensen, 2013], cette expansion des métaheuristiques est soulignée, la contribution de tels travaux étant souvent questionnable. De manière générale, cet article de Sörensen porte un regard très critique sur une partie de ces travaux qui peuvent parfois nuire à la crédibilité du domaine. Il conclut que l'objectif doit être d'apporter des résultats concrets plutôt que de proposer systématiquement "une méthode A pour résoudre un problème P", sans pouvoir analyser précisément l'apport de chaque mécanisme, la généricité de la méthode ou son efficacité réelle. Depuis quelques années, je cherche de ainsi à obtenir des résultats généraux sur les métaheuristiques en essayant de m'abstraire de biais divers tels que la spécificité des problèmes étudiés ou le paramétrage, sans prétendre y être complétement parvenu.

Une des raisons du manque de résultats théoriques sur les métaheuristiques est qu'ils sont souvent difficiles à obtenir ou peu significatifs. Par exemple, on ne pourra jamais certifier qu'une métaheuristique est toujours meilleure qu'une autre, car le théorème *No Free Lunch* dit qu'il est toujours possible de construire une instance où ce ne sera pas le cas [Wolpert et Macready, 1997]. Parfois, des preuves de convergence de métaheuristiques sont données, mais l'intérêt d'une telle information est souvent limité. En effet, pour s'assurer qu'une métaheuristique converge, il suffit de générer de temps en temps une solution aléatoire, ce qui assure de trouver une solution optimale tôt ou tard. Il existe également des algorithmes à garantie de performance qui assurent de ne pas dépasser une certaine marge d'erreur, mais les résultats dans le cadre des métaheuristiques pour l'optimisation combinatoire sont assez limités et parfois contraignants pour la métaheuristique même. Enfin, il est possible de prévoir le comportement des métaheuristiques par une étude exhaustive de petits espaces de recherche, comme cela est présenté dans ce document. Néanmoins, il est nécessaire de modérer la portée des résultats obtenus, car le comportement des métaheuristiques est souvent différent lorsque l'on traite des problèmes de grandes tailles.

Ces raisons font que les résultats empiriques sont globalement préférés, ce qui peut donner lieu à toutes sortes de dérives quant aux conditions d'obtention des résultats. En effet, est-il raisonnable d'apporter des conclusions de la forme "*une métaheuristique A est meilleure qu'une métaheuristique B*" si cela est observé sur un (petit) ensemble d'instances d'un problème d'optimisation donné, considérant un paramétrage particulier et selon un temps d'exécution déterminé ? À cela peut s'ajouter des aspects de choix d'implémentation, ainsi que le choix du protocole de

comparaison qui peut influer sur les conclusions de l'analyse empirique.

Il est clairement difficile, voire impossible, de supprimer tous les biais qui rendent toute analyse empirique du comportement des métaheuristiques incertaine. Malgré tout, certains aspects permettent de réduire ce biais.

Premièrement, les métaheuristiques doivent être les plus autonomes possibles. Elles doivent être facilement adaptables à de nouveaux problèmes (pour justifier le préfixe *méta*). Pour cela, elles doivent nécessiter le moins de paramètres possible, ou tout au moins son comportement doit être le moins sensible possible au paramétrage. Il est en effet fastidieux de comparer deux méthodes si les résultats différent grandement selon le paramétrage établi ; cela peut donner lieu à des dérives consistant à *avantager* une métaheuristique vis à vis d'une autre. Dans le cas où certains paramètres cruciaux ne peuvent être supprimés, la proposition de méthodes de réglage automatique des paramètres peut alors être envisagé [Hamadi *et al.*, 2012]. Ce type d'étude est d'ailleurs encouragé dans [Sörensen, 2013].

Deuxièmement, il est nécessaire de comparer ce qui est comparable, c'est à dire qu'il faut être capable d'apporter des conclusions claires sur des mécanismes spécifiques des métaheuristiques. En effet, comparer un recuit simulé particulier avec un algorithme génétique particulier, même sur un ensemble complet de problèmes, ne permet pas de conclure sur la supériorité de l'une sur l'autre de manière générale (il faudrait pour cela comparer toutes les variantes de recuit simulé avec toute les variantes d'algorithmes génétiques). De telles conclusions générales sont évidemment très difficiles à obtenir de manière cohérente, c'est pourquoi il convient de se focaliser sur des aspects plus basiques des métaheuristiques, en s'intéressant généralement à un mécanisme spécifique pour lequel les résultats pourront être beaucoup plus parlants.

Enfin, l'analyse empirique réalisée doit être la plus complète et significative possible. Voici quelques exemples d'aspects à considérer :

- Étudier un panel de problèmes et d'instances le plus large et varié possible.
- Étudier la sensibilité des algorithmes aux différents paramétrages possibles.
- Minimiser le biais éventuel introduit par l'aspect stochastique des méthodes. Cela implique de réaliser une comparaison statistique des résultats des métaheuristique. De plus, il est préférable de réaliser les expériences dans les mêmes conditions initiales (mêmes solutions de départ, même graine aléatoire).
- Étudier le comportement des métaheuristiques au fil de l'exécution. La question est de savoir si la supériorité d'une méthode sur une autre est conditionnée au temps d'exécution alloué. Dans l'idéal, on s'intéressera aux métaheuristiques *anytime*, dont l'objectif est d'être le plus performant possible quel que soit le temps d'exécution accordé.
- Il faut pouvoir identifier la cause des résultats, c'est à dire qu'il doit être possible de mettre en exergue un composant qui rend une méthode meilleure qu'une autre.

Dans mes travaux en optimisation combinatoire multiobjectif, je cherche donc à concevoir des métaheuristiques génériques simples, ayant peu de paramètres ou peu sensibles au paramétrage, et efficaces sur le plus large panel de problèmes possibles. Ce n'est bien sûr pas un objectif facile à atteindre, donc sans vouloir prétendre réaliser cet objectif, mes travaux essaient tout au moins d'aller dans ce sens. Une autre partie de mes travaux, débutée plus récemment, porte sur les *climbers* pour l'optimisation combinatoire mono-objectif. L'objectif est de restreindre le champ des recherches (un seul objectif à optimiser, limitation de l'étude au cadre d'une amélioration itérative) qui, allié à une analyse des caractéristiques des problèmes, permet de pouvoir extraire des informations globales.

1.4 Organisation du document

Le document recense une partie des travaux que j'ai pu réaliser ces dernières années, souvent en collaboration avec des collègues d'Angers ou d'ailleurs, ou dans le cadre d'encadrement de thèses et de masters recherche. Le document est divisé en deux parties principales.

La **première partie** présente nos travaux réalisés dans le cadre de l'étude des recherches locales pour l'optimisation combinatoire mono-objectif. Il s'agit d'un ensemble de travaux récents datant d'après 2012, mais qui ont pris une importance grandissante dans mes activités de recherche.

Dans le **chapitre 2**, l'optimisation combinatoire mono-objectif est introduite ainsi que la notion de *paysage de recherche*. Cette notion va permettre de nous abstraire des problèmes étudiés afin de nous concentrer sur les méthodes permettant de grimper le plus haut possible dans ces paysages. Nous discutons des indicateurs permettant de caractériser la structure des paysages de recherche, puis nous donnons quelques résultats de caractérisation de paysages dérivés de problèmes combinatoires classiques, ainsi qu'une étude des propriétés des paysages de recherche. Dans ce chapitre sont également introduites les notions de recherche locale et de climber, et l'évaluation des métaheuristiques est discutée.

Le **chapitre 3** est consacré à la comparaison des climbers sur des paysages de recherches divers. Nous commençons par recenser les composants de base des climbers, puis nous nous concentrons sur deux composants essentiels définissant la politique d'acceptation des solutions durant la recherche (la *règle pivot* et la *politique de prise en compte de la neutralité*). L'analyse empirique des climbers combinées à l'analyse des paysages testés nous amènent à des conclusions intéressantes et qui contredisent parfois les choix d'implémentation effectués régulièrement dans la littérature.

Dans le **chapitre 4** nous nous intéressons à la conception de nouveaux climbers, capables d'atteindre de meilleurs optima locaux que les climbers classiques utilisés dans les métaheuristiques. L'objectif est de proposer des alternatives aux composants de base classiques. En particulier, nous proposons et analysons des règles pivot alternatives et montrons l'intérêt de modifier artificiellement un paysage de recherche afin de pouvoir l'explorer plus efficacement.

Le **chapitre 5** présente les conclusions résultant des différents travaux effectués dans la première partie et propose quelques voies de recherche à explorer.

La **deuxième partie** présente nos travaux réalisés sur la conception de métaheuristiques génériques pour l'optimisation combinatoire multiobjectif. Ce domaine est resté longtemps prioritaire dans mes recherches et à fait l'objet d'une grande partie de mes publications. Je présente une partie de ces travaux, qui sont globalement moins détaillés que pour la première partie.

Le **chapitre 6** introduit le contexte particulier de l'optimisation multiobjectif et des définitions sont introduites. Un survol des approches de résolution est proposé et l'évaluation d'approximations d'ensembles de Pareto est discutée. Enfin, nous discutons de l'étude des paysages de recherche multiobjectif.

Dans le **chapitre 7**, nous présentons l'algorithme IBMOLS, algorithme de recherche locale dont la sélection est basée sur un indicateur binaire de qualité. Des expérimentations sont réalisées, incluant une analyse de sensibilité au paramétrage. Des tests ont été réalisés sur différents problèmes montrent l'intérêt de ce type d'approche.

Les résultats obtenus par IBMOLS ont soulevé des questions qui nous ont poussé ensuite à

concevoir l'algorithme HBMOLS présenté dans le **chapitre 8**. Cet algorithme basé sur le même principe qu'IBMOLS, base la sélection sur l'indicateur unaire d'hypervolume de dominance. Les expérimentations menées montrent la capacité de HBMOLS à être efficace sur différents problèmes et à être peu sensible au paramétrage, qui se réduit principalement à définir une taille de population. Nous discutons également des limites de HBMOLS, qui s'avère très lent lorsque le nombre d'objectifs à optimiser augmente.

Dans le **chapitre 9**, nous nous intéressons à l'utilisation de l'algorithme du *path-relinking* pour initialiser les solutions de l'algorithme HBMOLS itéré. Nous discutons des différents choix d'implémentation possibles pour le *path-relinking*, et des expérimentations sont menées afin de déterminer l'impact de ces choix ainsi que l'utilité d'utiliser le *path-relinking* comme fonction d'initialisation de HBMOLS.

Dans le **chapitre 10**, nous proposons de reformuler les problèmes d'optimisation multiobjectif sous forme de problèmes d'optimisation mono-objectif où l'on cherche à maximiser la qualité d'un ensemble de solutions compromis, la qualité étant définie par un indicateur tel que l'hypervolume de dominance. Nous définissons des classes de voisinage dans un tel contexte, puis nous proposons une étude expérimentale afin d'évaluer la performance d'un climber selon la classe de voisinage considérée.

Le **chapitre 11** présente les conclusions et perspectives portant sur la deuxième partie de ce manuscrit, ainsi qu'un rapide aperçu de l'orientation générale de mes futures recherches. Enfin, la fin du document contient mon CV détaillé, ainsi qu'une liste de mes publications.

Première partie

Recherches locales pour l'optimisation combinatoire mono-objectif

Jusqu'à récemment, j'ai essentiellement travaillé dans le domaine de l'optimisation combinatoire multiobjectif en m'intéressant particulièrement à la conception de métaheuristiques génériques efficaces. Cependant, les travaux menés aux cours de ces années m'ont amené à penser que dans un cadre multiobjectif, il était difficile d'évaluer de manière objective l'efficacité générale de composants proposés dans le cadre de travaux de recherche. Il y a en général beaucoup d'aspects susceptibles de biaiser l'analyse et de surcroît il est difficile de caractériser de manière précise les paysages de recherche multiobjectif.

Depuis 3 ans, je me suis donc intéressé à l'optimisation mono-objectif, en me restreignant aux algorithmes de recherche locale de type *climber*. Il m'a semblé de plus en plus discutable de chercher à proposer des métaheuristiques avancées alors même que la conception d'un simple climber implique de se poser des questions pour lesquelles il n'existe pas de réponse claire dans la littérature. Afin de pouvoir étudier les climbers dans un cadre général, nous nous plaçons dans un contexte d'optimisation de paysages de recherche, qui permet de s'abstraire des problèmes d'optimisation et d'aspects liés tels que la représentation des solutions ou la structure de voisinage utilisée.

Les travaux présentés ont étés principalement réalisés avec Adrien Goëffon, maitre de conférences au LERIA. L'étude de l'atteignabilité de l'optimum global de paysages de recherche a été publiée dans [Vigneron *et al.*, 2014], la comparaison des climbers dans [Basseur et Goëffon, 2013] et l'étude des climbers de type *pire améliorant* dans [Basseur et Goëffon, 2014]. De plus, ces travaux ont également fait l'objet d'autres communications, et des articles sont en cours d'évaluation. En effet, une partie des résultats de recherche présentés ici ne sont pas encore publiés.

Chapitre 2

Optimisation combinatoire et analyse de paysage

Dans ce chapitre, nous introduisons quelques concepts généraux sur l'optimisation combinatoire et reformulons l'optimisation basée sur des voisinages sous forme de *paysages de recherche*. Ensuite, nous nous intéressons à l'analyse des paysages de recherche, en proposant des indicateurs pour les caractériser et nous proposons des résultats sur des paysages divers, dont certains sont dérivés de problèmes d'optimisation combinatoire classiques.

2.1 Contexte et définitions de base

2.1.1 Optimisation combinatoire

Un problème d'optimisation combinatoire (mono-objectif) peut être défini comme un ensemble discret de *solutions admissibles* X, ainsi que d'une fonction d'évaluation $f : X \rightarrow \mathbb{R}$ devant être maximisée ou minimisée. Dans la suite de ce document, sauf indication contraire, on se restreindra à un contexte de maximisation sans perte de généralité. Résoudre un problème (X,f) consistera alors à trouver $x^\star \in \text{argmax}_{x \in X} f(x)$ (ce qui est équivalent à trouver $x^\star \in \text{argmax}_{x \in X} -f(x)$ si l'on considère un problème de minimisation).

Dans les problèmes d'optimisation, les *variables de décision* sont des quantités numériques pour lesquelles des valeurs sont à choisir. Cet ensemble de variables est appelé *vecteur de décision*. Soit un problème d'optimisation avec n variables de décision, le vecteur de décision $\overrightarrow{x}$ est représenté par $[x_1,x_2,\ldots,x_n]$ et les différentes valeurs possibles prises par les variables de décision x_i constituent l'ensemble des solutions envisageables. Un problème d'optimisation est dit continu si les domaines de définition des différentes variables de décision sont continus (souvent dans $\mathbb{R}$). Un problème d'optimisation est dit combinatoire si les domaines de définition des variables de décision sont discrets (souvent les variables sont binaires ou dans $\mathbb{N}$).

L'ensemble des $n - uplets$ de valeurs composant le vecteur de décision est un espace de dimension n. L'ensemble des valeurs pouvant être prises par le vecteur de décision constitue l'*espace de recherche* de la méthode d'optimisation. On distingue l'espace *décisionnel* de dimension n, de l'espace *objectif* correspondant à l'ensemble de définition de la (des) fonction(s) objectif(s).

2.1.2 Méthodes de résolution

Les méthodes de résolution des problèmes d'optimisation sont très diverses et sont destinées soit à leur résolution de manière exacte (méthodes complètes), soit de manière approchée (ou heuristique). Les méthodes complètes ont l'avantage de fournir une garantie sur le résultat obtenu, qui est prouvé comme étant optimal, mais très souvent on ne peut garantir de trouver la solution optimale en un temps raisonnable. Les méthodes approchées, quant à elles, peuvent permettre de fournir très rapidement des solutions intéressantes, mais n'offrent pas la garantie de trouver la solution optimale ni même une solution proche de la solution optimale (sauf dans le cas d'algorithmes à garantie de performance).

Méthodes complètes

Afin d'avoir l'assurance d'obtenir une solution optimale, les méthodes complètes doivent parcourir l'ensemble de l'espace de recherche, ou au moins s'assurer de n'oublier aucune solution potentiellement meilleure que la meilleure solution trouvée par l'algorithme. Les méthodes de résolution les plus répandues sont certainement celles construisant un arbre de recherche afin d'explorer l'ensemble de l'espace de recherche. Ces méthodes sont appelées *Branch & ** ('*' pouvant être remplacé par *Bound*, *Cut*, *Price* ou *Cut & Price*). Elles utilisent des bornes pour éliminer de la recherche des ensembles de solutions ayant une certaine structure. D'autres méthodes sont spécifiques à un problème donné, ou moins générales, comme la programmation dynamique, le simplexe, et le problème de programmation linéaire en nombre entiers.

Les méthodes complètes sont peu étudiées dans mes travaux et ne sont pas présentées dans ce document, qui se focalise sur les méthodes approchées. Les méthodes complètes sont généralement limitées à la résolution de petites instances de problèmes et sont souvent incapables de fournir un résultat convenable lorsque l'espace de recherche à explorer est trop grand. Bien que préférables aux méthodes approchées grâce à la garantie d'optimalité, les méthodes complètes ne sont pas toujours *praticables*. Dans mes travaux, je m'intéresse aux méthodes approchées pour la résolution d'instances trop larges pour être résolues de manière exacte.

Méthodes approchées

Les méthodes de résolution approchée deviennent utiles lorsque les méthodes exactes échouent (espace de recherche trop grand, par exemple). Dans ce cas, une exécution partielle de l'algorithme exact permet rarement d'obtenir une solution de bonne qualité et dans ce cas les méthodes de résolution approchées offrent une alternative intéressante.

On trouve de nombreuses heuristiques qui utilisent les spécificités du problème traité afin de générer de manière partiellement stochastique ou non, de bonnes solutions. Ces approches sont appelées *heuristiques constructives*. Parmi ce type d'approche, les algorithmes *gloutons* consistent à créer une (des) solution(s) pas à pas, en exploitant les particularités du problème. Pour un problème de voyageur de commerce, par exemple, un algorithme glouton basique serait de visiter à chaque itération la ville non visitée la plus proche.

Les méthodes de résolution approchées de type *métaheuristiques* cherchent au contraire à s'abstraire au maximum des problèmes considérés et de proposer un cadre générique de résolution

des problèmes d'optimisation en général. Parmi les métaheuristiques les plus connues on peut citer les stratégies d'évolution, les algorithmes génétiques, les algorithmes à évolution différentielle, les algorithmes à estimation de distribution, les systèmes immunitaires artificiels, la recomposition de chemin (*path-relinking*), le recuit simulé, les algorithmes de colonies de fourmis, les essaims particulaires, les algorithmes de descente (itérée ou non), la recherche tabou, la méthode GRASP (*Greedy Randomized Adaptive Search Procedure*). L'objectif ici n'est pas de détailler le fonctionnement de ces différentes métaheuristiques ; on se référera à [Dréo *et al.*, 2006, Talbi, 2009] pour une description complète de ces méthodes d'optimisation.

Le point commun de la plupart des métaheuristiques est l'utilisation d'une structure de *voisinage*, qui est un sous-ensemble de solutions qu'il est possible d'atteindre par une modification d'une solution courante. Le voisinage a un impact important sur le comportement des métaheuristiques, car il influe sur la facilité d'accès des différentes solutions de l'espace de recherche et donc sur la difficulté même de l'optimisation du problème considéré.

Dans la suite de cette partie, on s'intéresse avant tout à comprendre ce qui rend les recherches efficaces ou non. Pour cela, nous nous sommes concentrés sur les climbers, car ils constituent la métaheuristique la plus basique tout en étant impliqués, directement ou non, dans la plupart des métaheuristiques efficaces de la littérature. Cette étude des climbers est couplée à une étude des paysages de recherche afin de déterminer les facteurs influant sur la qualité des méthodes d'optimisation.

2.1.3 Paysage de recherche

Le terme de *paysage de recherche* (traduit de *fitness landscape* [1]) prend ses origines en biologie où il a été introduit pour illustrer le processus de l'évolution. Le papier fondateur [Wright, 1932] visualise sous forme d'espaces phénotypique/génotypique les mécanismes d'évolution des espèces. De nos jours, la notion de paysage de recherche est largement utilisée dans divers domaines de recherche, mais plus particulièrement en algorithmique évolutionnaire, car cela permet d'étudier de manière théorique et/ou empirique le comportement des algorithmes de recherche [Mitchell *et al.*, 1992, Merz et Freisleben, 1999, Ratle, 2001].

Formellement, un *paysage de recherche* est un triplet $(\mathcal{X},\mathcal{N},\mathfrak{f})$, où $\mathcal{X}$ est un *espace de recherche* contenant un ensemble de *configurations* (ou *solutions candidates*), $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ est une *relation de voisinage*, et $\mathfrak{f}$ est une *fonction objectif* (où *fonction d'évaluation*). $\mathfrak{f}(x)$ représente le *fitness* (ou *hauteur*) de $x \in \mathcal{X}$.

Généralement, on définit $\mathcal{X}$ et $\mathfrak{f}$ de telle sorte que $X = \mathcal{X}$ et $f = \mathfrak{f}$, afin de résoudre exactement le problème d'optimisation souhaité. Cependant ce n'est pas toujours le cas. En effet, dans certains cas où la fonction d'évaluation est particulièrement coûteuse en ressources, il peut être souhaitable de définir $\mathfrak{f}$ comme étant une approximation de f, mais beaucoup plus rapide à calculer. De plus, utiliser une version modifiée de f peut également rendre un algorithme de recherche plus performant, celle-ci évitant plus facilement d'aboutir sur des optima locaux de mauvaise qualité. Ce point sera discuté plus tard dans ce document. La remarque est valable également pour $\mathcal{X}$, qui peut considérer des solutions non-réalisables de X, afin de rendre la recherche plus efficace. De même, des solutions de X peuvent être ignorées dans $\mathcal{X}$, par exemple lorsque l'on est capable de montrer

1. Une traduction littérale donnerait plutôt *paysages de fitness*, mais ce terme est plus en lien avec la définition originale donnée par les biologistes, qui est déconnectée des algorithmes de recherche que nous utilisons pour les explorer. C'est pourquoi nous préférerons ici le terme *paysage de recherche*, qui fait référence à l'aspect optimisation.

qu'une partie des solutions réalisables sont nécessairement moins bonnes que d'autres solutions de l'espace de recherche. Dans ce document, nous utiliserons indifféremment les termes *solution* (candidate) et *configuration* ; d'une manière générale, le terme configuration sera préféré dans un contexte de paysage de recherche.

Dans la suite, on appellera *graphe de transition* le graphe $TG(\mathcal{X},\mathcal{N}[,\mathfrak{f}]) = (\mathcal{X},\{(a,b),b \in \mathcal{N}(a)\})$ dont les sommets sont les configurations de $\mathcal{X}$ et les arcs les mouvements autorisés par la relation de voisinage $\mathcal{N}$. La distance $d(x,y)$ entre deux configurations x et y correspond a la longueur du plus court chemin entre x et y dans le graphe de transition associé à l'espace de recherche et le voisinage considéré. On appellera *graphe de transition de recherche itérée* le graphe de transition contraint $TG_{\triangleright}(\mathcal{X},\mathcal{N},\mathfrak{f})$, où les arcs ne satisfaisant pas un critère de transition $\triangleright$ ont été supprimés. Par exemple, on notera $TG_{>}(\mathcal{X},\mathcal{N},\mathfrak{f}) = (\mathcal{X},\{(a,b),b \in \mathcal{N}(a) \text{ et } \mathfrak{f}(b) > \mathfrak{f}(a)\})$, le graphe de transition induit par $(\mathcal{X},\mathcal{N},\mathfrak{f})$ et l'amélioration stricte comme critère de transition.

Enfin, on appellera *optimum local* une configuration x telle que $\forall x' \in \mathcal{N}(x), \mathfrak{f}(x') \leqslant \mathfrak{f}(x)$.

Un *optimum local scrict* est une configuration x telle que $\forall x' \in \mathcal{N}(x), \mathfrak{f}(x') < \mathfrak{f}(x)$.

Un *optimum global* est une configuration $x^\star \in \operatorname{argmin}_{x\in\mathcal{X}} \mathfrak{f}(x)$ (il peut y avoir plusieurs optima globaux).

La figure 2.1 illustre sous forme d'exemple les termes qui viennent d'être introduits. De plus, cet exemple montre l'intérêt que l'on peut avoir à définir $X \neq \mathcal{X}$ et $f \neq \mathfrak{f}$. Ceci est l'un des aspects étudiés dans mes recherches.

Remarque : les paysages de recherche ont été introduits dans un contexte de maximisation du fitness de configurations. Pour cette raison, les différents aspects présentés dans ce manuscrit se placent systématiquement dans ce contexte. Évidemment, de nombreux problèmes d'optimisation consistent à minimiser un ou plusieurs objectifs. Dans ce cas, le fitness $\mathfrak{f}$ des configurations est défini comme étant l'opposé de la fonction objectif f. Pourtant, les tableaux de résultats présentés pour ces paysages reporteront les valeurs de la fonction objectif afin de pouvoir se rapporter facilement au problème combinatoire sous-jacent et de pouvoir évaluer les résultats par rapport au reste de la littérature.

2.1.4 Algorithme de recherche locale

Un algorithme de recherche locale consiste à explorer un sous-ensemble de $\mathcal{X}$ par l'intermédiaire du graphe de transition de recherche itérée $TG_{\triangleright}(\mathcal{X},\mathcal{N},\mathfrak{f})$, dans le but de trouver la meilleure solution de $\mathcal{X}$ (par rapport à $\mathfrak{f}$). Globalement, l'objectif de ces méthodes est de trouver des solutions de qualité acceptable dans un temps raisonnable. Résoudre une instance $(\mathcal{X},\mathfrak{f})$ avec un algorithme de recherche locale se limite ici à définir une relation de voisinage $\mathcal{N}$ et une stratégie de transition de manière à ce que l'exploration du paysage de recherche $(\mathcal{X},\mathcal{N},\mathfrak{f})$ soit la plus efficace possible.

L'algorithme 2.1 décrit de manière très générale le fonctionnement d'un algorithme de recherche locale. Le critère d'arrêt peut être par exemple un temps d'exécution ou une condition sur la solution atteinte (optimum local, qualité recherchée atteinte, plus d'amélioration récente, etc.). Le critère de sélection d'une configuration dans le voisinage doit faire en sorte que la qualité de la configuration courante tende à augmenter durant la recherche, sans pour autant forcer une amélioration systématique du fitness. Par conséquent, il convient naturellement de sauvegarder la meilleure configuration atteinte afin de pouvoir retourner le meilleur résultat possible. Cela n'est

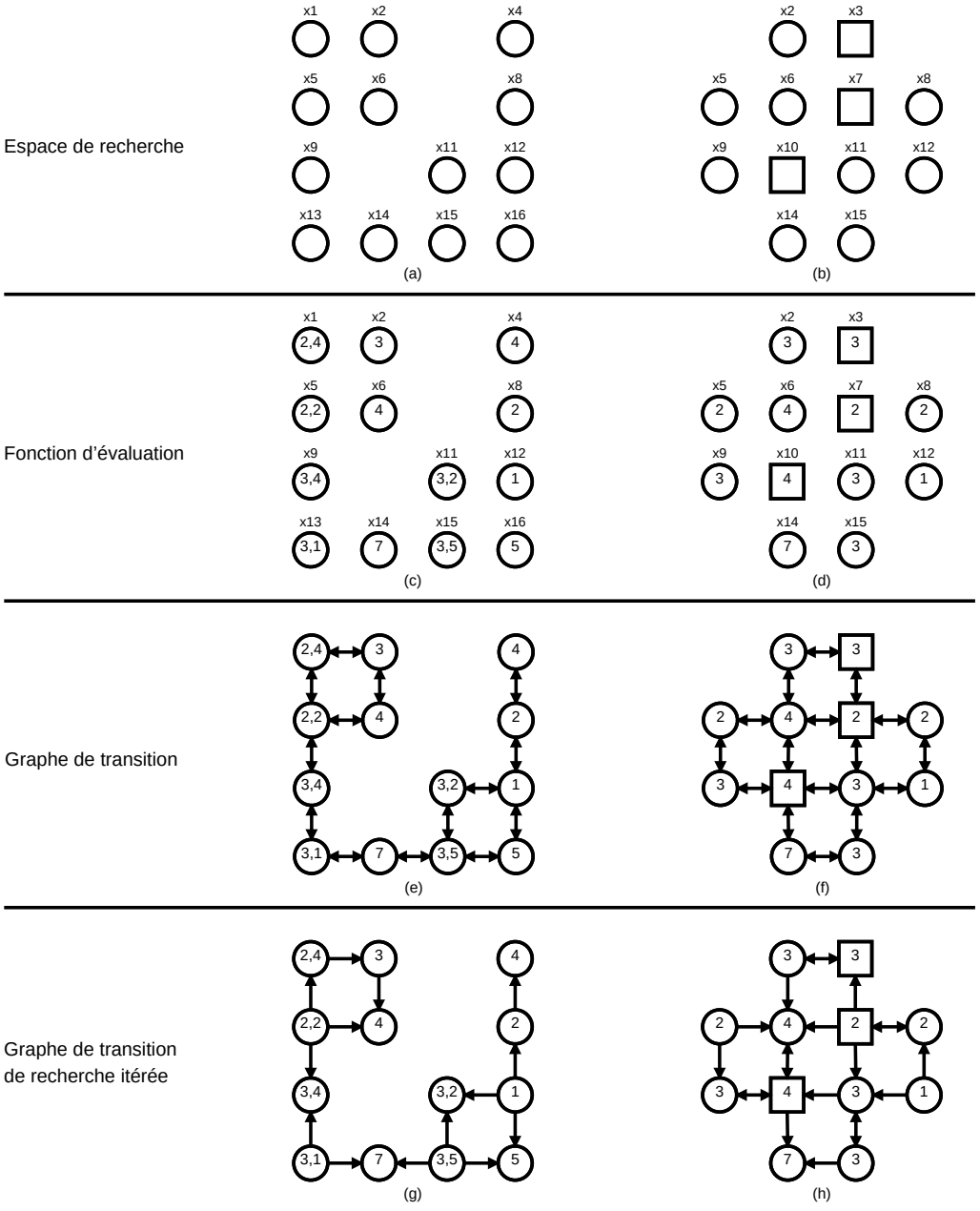


FIGURE 2.1 – D'un problème d'optimisation à un graphe de recherche locale itérée (exemple) :
(a) Ensemble X des solutions admissibles du problème d'optimisation.
(b) Espace de recherche $\mathcal{X}$ du problème d'optimisation (les solutions $x1$, $x4$, $x13$ et $x16$ de X ne sont pas considérées dans $\mathcal{X}$; les solutions $x3$, $x7$ et $x10$ de $\mathcal{X}$ sont non réalisables).
(c) Fonction objectif des solutions de X.
(d) Fitness des solutions de $\mathcal{X}$ (fonction objectif aux décimales tronquées).
(e) Graphe de transition $TG(X,\mathcal{N})$ (les solutions adjacentes sur la figure forment $\mathcal{N}$). Muni de la fonction objectif, on obtient le paysage de recherche $(X,\mathcal{N},f)$.
(f) Graphe de transition $TG(\mathcal{X},\mathcal{N},\mathfrak{f})$. Muni de la fonction d'évaluation, on obtient le paysage de recherche $(\mathcal{X},\mathcal{N},\mathfrak{f})$.
(g) Graphe de transition de recherche itérée $TG_{\geqslant}(X,\mathcal{N},\mathfrak{f})$. Seules les solutions initiales $x13$, $x14$ et $x15$ peuvent permettre d'atteindre l'optimum global ($x15$).
(h) Graphe de transition de recherche itérée $TG_{\geqslant}(\mathcal{X},\mathcal{N},\mathfrak{f})$. Les recherches aboutissent toutes tôt ou tard à l'optimum global, quelle que soit la configuration initiale.

Algorithme 2.1 Recherche locale générique

Entrée : une configuration initiale x
$x* \leftarrow x$
tant que critère d'arrêt non atteint **faire**
  sélectionner $x' \in \mathcal{N}$
  $x \leftarrow x'$
  si $f(x) > f(x*)$ **alors** $x* \leftarrow x$
Retourner $x*$

pas nécessaire lorsqu'on interdit la détérioration du fitness, comme dans le cas des climbers (voir ci-dessous).

2.1.5 Climber

Un *climber*, ou algorithme de *descente*, est une recherche locale basique qui consiste à se déplacer dans l'espace de recherche (ici un paysage de recherche) en autorisant uniquement les mouvements non-détériorant [2]. Étant donné une *configuration initiale*, des mouvements successifs sont appliqués afin d'atteindre systématiquement un voisin de meilleure qualité, jusqu'à ce qu'un optimum local strict soit atteint ou que tout autre critère d'arrêt soit satisfait. Cette méthode de recherche permet de distinguer quelques variantes qui font l'objet de discussions dans le chapitre suivant. Un climber est un algorithme de recherche locale, mais la meilleure configuration rencontrée ne doit pas être sauvegardée puisque il s'agit nécessairement de la configuration courante, qui est alors retournée à la fin de l'exécution.

2.2 Analyse de paysage

Nous définissons dans un premier temps les propriétés que nous avons utilisées pour caractériser les paysages de recherche. Puis, nous présentons quelques problèmes classiques d'optimisation combinatoire, que nous définirons en tant que paysages de recherche afin de pouvoir fournir une analyse de ces paysages.

2.2.1 Propriétés des paysages de recherche

La caractérisation des paysages en fonctions d'indicateurs a fait l'objet de nombreux travaux, présentés récemment dans un état de l'art [Malan et Engelbrecht, 2013]. Nous proposons ici un ensemble de mesures nous permettant d'établir un parallèle entre caractéristiques des paysages de recherche et comportement des climbers. Nous présentons cdans un premeir temps ces mesures (taille, connectivité, rugosité, neutralité), puis nous les utiliserons ensuite pour analyser le comportement des algorithmes de descente selon les caractéristiques de divers paysages de recherche.

2. Dans la suite de ce document, on préférera malgré l'anglicisme le terme de *climber*, le terme étant souvent usité dans la communauté francophone. La raison principale est que les paysages de recherche sont définis dans un contexte de maximisation de fitness, par conséquent le terme d'algorithme de *descente* est moins intuitif et peut prêter à confusion.

Dimension

La dimension d'un paysage de recherche sera déterminée ici par deux mesures :
- la **taille** de l'espace de recherche, c'est-à-dire le nombre de solutions candidates. Cela affecte clairement la difficulté que l'on aura à trouver une configuration particulière, par exemple l'optimum global.
- la **connectivité** du paysage induit par la relation de voisinage, c'est à dire le nombre de voisins des solutions de l'espace de recherche.

Notons qu'il existe des méthodes avancées pour évaluer la connectivité d'un graphe et donc d'un paysage de recherche : diamètre, densité, distance moyenne entre les solutions, etc. Cependant, les structures de voisinages utilisées pour des problèmes d'optimisation combinatoire classiques induisent souvent des graphes de transitions symétriques ou quasi-symétriques, ce qui est le cas des problèmes étudiés dans nos travaux. C'est pourquoi nous considérons ici que la taille du voisinage est un indicateur pertinent pour évaluer la connectivité d'un paysage de recherche.

Neutralité

Dans de nombreux paysages de recherche, plus particulièrement si la fonction de fitness est faiblement discrétisée, une part non négligeable de solutions voisines peut avoir des fitness égaux. Dans ce cas, le fitness ne permet pas de déterminer si une solution voisine est préférable ou non à la solution courante. Ce phénomène, appelé couramment *neutralité*, peut avoir une grande influence sur le comportement des méthodes de recherche [Smith *et al.*, 2002]. Par conséquent, la quantification de cette neutralité est un aspect majeur de la caractérisation des paysages de recherche.

On appellera *voisin neutre* d'une configuration x, une configuration x' telle que $x' \in \mathcal{N}(x)$ et $\mathfrak{f}(x') = \mathfrak{f}(x)$. Un *plateau* est défini comme étant un graphe (V,E) où $V \in \mathcal{X}$, $E = \{(v_i,v_j)$ tel que $v_j \in \mathcal{N}(v_i)$ et $\mathfrak{f}(x_i) = \mathfrak{f}(x_j),\forall x_i,x_j \in V\}$, c'est-à-dire un ensemble de configurations de même fitness.

Les indicateurs de neutralité proposés dans la littérature ont été répertoriés récemment dans [Malan et Engelbrecht, 2013] et se focalisent en général sur la notion de plateau [Reidys *et al.*, 1998, Vanneschi *et al.*, 2006]. Dans notre cas, nous avons choisi de considérer la proportion d'arcs *neutres* dans les graphes de transition, ce qui correspond au taux moyen de voisins neutres d'une configuration (voir la définition 1). Cela nous a semblé suffisant pour caractériser la neutralité d'un espace de recherche, même si la répartition de la neutralité dans l'espace de recherche n'est pas prise en compte. La simplicité de la mesure permet d'approximer facilement sa valeur en échantillonnant l'espace de recherche.

Définition 1 (taux de neutralité) *Soit* $(\mathcal{X},\mathcal{N},\mathfrak{f})$ *un paysage de recherche, son* taux de neutralité *est défini par :*

$$\nu(\mathcal{X},\mathcal{N},\mathfrak{f}) = \frac{\#\{(x,y) \mid x \in \mathcal{N}(y) \wedge \mathfrak{f}(x) = \mathfrak{f}(y)\}}{\#\{(x,y) \mid x \in \mathcal{N}(y)\}} \tag{2.1}$$

Rugosité

Pour les biologistes, un paysage de recherche est une cartographie du rapport génotype / phénotype qui met en lumière l'effet des mutations (modification du génotype) sur le fitness des indi-

vidus (chances de survie à long terme). Dans ce contexte, les interactions entre les gènes tendent à augmenter la complexité du paysage de recherche, puisque cela empêche d'établir le caractère bénéfique ou non de chaque gène pris individuellement. Ce phénomène, appelé *épistasie* chez les biologistes, existe lorsque l'effet d'une mutation sur le fitness dépend de la présence ou non d'autres mutations [Bateson, 1909].

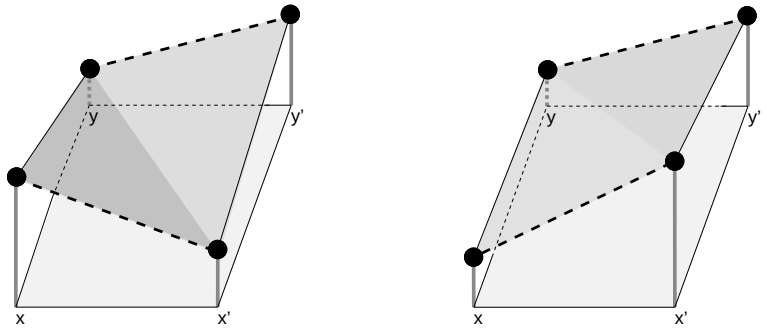


FIGURE 2.2 – Il y a épistasie lorsque l'effet d'une mutation ($x \rightarrow y$) sur le fitness dépend de la présence ou non d'une autre mutation ($x \rightarrow x'$, $y \rightarrow y'$). Une épistasie de signe est observée quand une mutation détériorante devient bénéfique si elle a lieu après une autre mutation (à gauche). À droite : pas d'épistasie de signe.

Bien que la rugosité des paysage soit un facteur extrêmement déterminant pour évaluer leur difficulté d'exploration, la définition même de la rugosité n'est pas clairement établie, plusieurs indicateurs tentent de l'évaluer en utilisant diverses approches (voir exemples de paysages de recherche de rugosité différentes dans la figure 2.3). Dans de nombreux travaux, le calcul de rugosité se résume à une mesure d'autocorrélation [Barnett, 1998]. Malan et Engelbrecht font une distinction entre l'interdépendance des variables (épistasie) et rugosité, référant plutôt au nombre et à la distribution des optima locaux [Malan et Engelbrecht, 2013]. Quoi qu'il en soit, il est clair qu'un grand nombre d'optima locaux implique naturellement un haut degré de dépendance entre les variables.

Étant donné que l'épistasie reflète clairement une rugosité locale, associée à l'opérateur de voisinage considéré, il semble naturel d'évaluer la rugosité des paysages de recherche par le taux d'interactions épistatiques de celui-ci. Nous nous intéressons ici au concept d'*épistasie de signe* [Poelwijk *et al.*, 2007], qui indique la présence d'une mutation bénéfique dans certains contextes génétiques, et détériorante dans d'autres contextes (voir la figure 2.2). L'aspect bénéfique ou détériorant des mouvements (mutations) réalisés sur des configurations (individus) est justement ce qui nous intéresse essentiellement dans un contexte d'exécution de climbers.

Dans un contexte de paysage de recherche combinatoire, cette mesure doit exprimer la corrélation entre le voisinage et le signe de l'évolution du fitness. Par analogie, les relations de voisinage représentent les mutations génétiques, tandis que le signe de la variation du fitness exprime l'aspect bénéfique ou détériorant d'une mutation. Cela nous conduit à définir une mesure de rugosité basée sur le concept d'*échelle* (voir la définition 2 et la figure 2.4) et sur la notion d'indicateur d'épistasie de signe.

Définition 2 (*k*-échelle) *Dans un graphe de transition $TG(\mathcal{X},\mathcal{N})$, on appelera k-*échelle *de sommets (x_0,x'_0,x_k,x'_k), une bipartition de solutions $\{(x_0,\ldots,x_k),(x'_0,\ldots,x'_k)\}$ telle que $\forall i,j \in$*

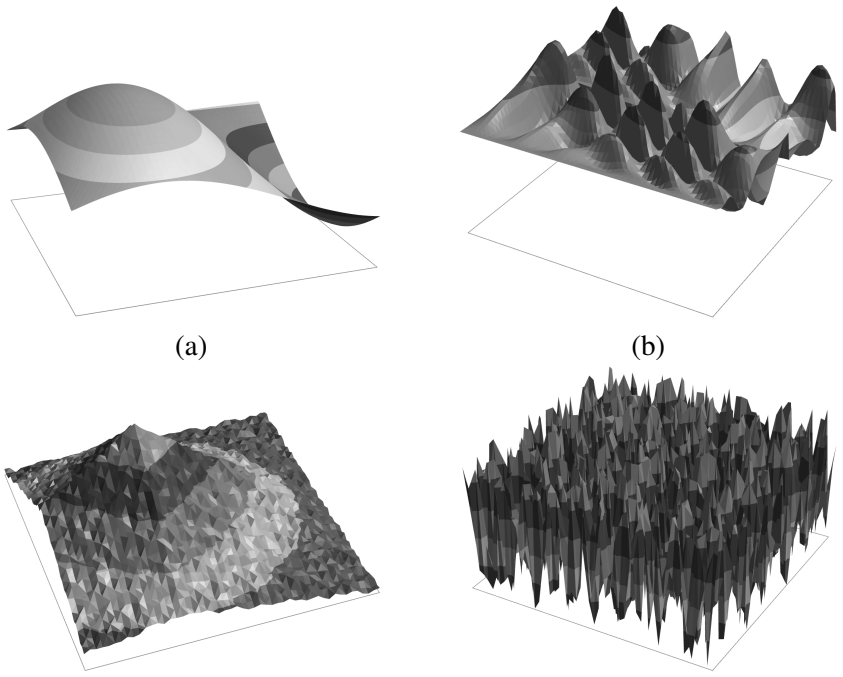


FIGURE 2.3 – Quatre paysages de recherche ayant des rugosités différentes : (a) paysage lisse, (b) paysage lisse localement, mais globalement rugueux, (c) paysage localement rugueux mais globalement lisse, (d) paysage extrêmement rugueux (hauteurs aléatoires). Ces paysages sont construits à partir d'un graphe de transition extrêmement simplifié, sous forme de grille à deux dimensions.

$\{0,\ldots,k\}$,$d(x_i,x_j) = d(x'_i,x'_j) = |i-j|$ *et* $d(x_i,x'_j) = |i-j|+1$. *On notera* $\mathcal{L}_k$ *l'ensemble de tous les quadruplets* (x,x',y,y') *étant les sommets d'une* k*-échelle, et* $\mathcal{L} = \cup_k \mathcal{L}_k$.

Définition 3 (indicateur d'épistasie de signe) *Soit* $(x,x',y,y') \in \mathcal{L}$ *une* k*-échelle, l'indicateur d'épistasie de signe* ξ *est défini comme suit :*

$$\xi^{\triangleright}(x,x',y,y') = \begin{cases} 1 \text{ if } (\mathrm{x}' \triangleright \mathrm{x}) \oplus (\mathrm{y}' \triangleright \mathrm{y}) & \textit{(épistasie de signe)} \\ 0 \text{ sinon} & \textit{(pas d'épistasie de signe)} \end{cases} \tag{2.2}$$

Le symbole $\oplus$ représente ici l'opérateur de disjonction et $\triangleright$ est un critère de comparaison. En effet, on considère en général deux interprétations de l'épistasie de signe, selon que l'on considère un mouvement vers un voisin de même fitness comme améliorant ou non ($\triangleright \in \{>\ ,\ \geqslant\}$). Un climber acceptant uniquement les mouvements strictement améliorants ($>$) considérera les mouvements neutres comme étant détériorants. Par contre, un climber acceptant les mouvements améliorants ou neutres ($\geqslant$) assimilera les mouvements neutres à des mouvements améliorants. Dans la table 2.1, la présence ou non d'épistasie de signe est indiquée, en fonction du critère de comparaison utilisé et la qualité relative de x par rapport à un de ses voisins x'.

En utilisant l'indicateur d'épistasie de signe ξ, nous pouvons définir un indicateur de k*-rugosité*, qui correspond au taux d'épistasie de signe des éléments des échelles $\mathcal{L}_k$ d'un paysage de recherche.

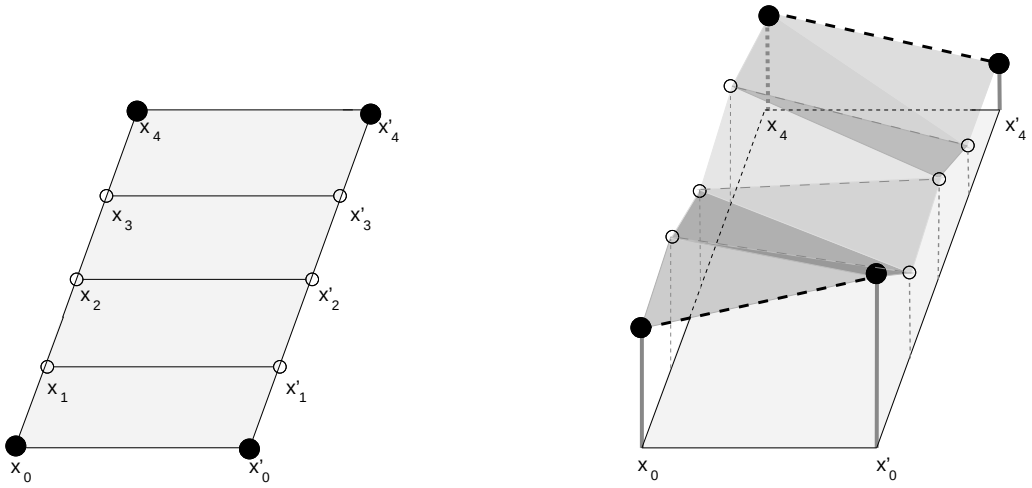


FIGURE 2.4 – Exemple d'une 4-échelle d'un point de vue "graphe de transition" (à gauche), et d'un point de vue "paysage de recherche" (à droite).

$\Delta\mathfrak{f}$	$\mathfrak{f}(x') - \mathfrak{f}(x)$	$-$	$-$	$-$	$=$	$=$	$=$	$+$	$+$	$+$
	$\mathfrak{f}(y') - \mathfrak{f}(y)$	$-$	$=$	$+$	$-$	$=$	$+$	$-$	$=$	$+$
Épistasie ($>$)		non	oui	oui	oui	non	non	oui	non	non
Épistasie ($\geqslant$)		non	non	oui	non	non	non	oui	non	non

TABLE 2.1 – Présence ou non d'épistasie de signe en fonction de la variation du fitness et du critère de comparaison ($>$,$\geqslant$).

Définition 4 (*k*-rugosité) *La* k-rugosité *d'un paysage de recherche* $(\mathcal{X},\mathcal{N},\mathfrak{f})$ *est définie par :*

$$\rho_k^{\triangleright}(\mathcal{X},\mathcal{N},\mathfrak{f}) = \frac{1}{\#\mathcal{L}_k} \sum_{q \in \mathcal{L}_k} \xi^{\triangleright}(q) \tag{2.3}$$

L'indicateur de k-rugosité peut être estimé par échantillonnage de l'espace de recherche. En faisant varier k, on obtient donc un ensemble de valeurs. La 1-rugosité, que l'on appellera rugosité locale, correspond au taux d'épistasie observé à partir de paires de mutations. Les k-rugosités correspondent à l'épistasie observée lorsqu'une même mutation est appliquée à des solutions situées à une distance k l'une de l'autre. Cela correspond donc aux rugosités observées à une échelle plus large. Pour illustrer cela, l'évolution des k-rugosités des paysages de la figure 2.3 en fonction de k est reportée sur la figure 2.5. Sur cette figure, on observe bien l'aspect local (k petit) ou global (k grand) de la rugosité.

Notons que pour les indicateurs de neutralité et de rugosité proposés ici, un seul indicateur est calculé pour caractériser l'ensemble de l'espace de recherche. Cela suppose donc que la rugosité et la neutralité des paysages de recherche est relativement uniforme sur l'ensemble des paysages. Or, les algorithmes de recherche locale tendent à explorer des solutions qui se trouvent sur les meilleures zones de l'espace de recherche, pour lesquelles ces propriétés peuvent être différentes de l'estimation générale effectuée lors de l'analyse du paysage. Cette particularité a notamment

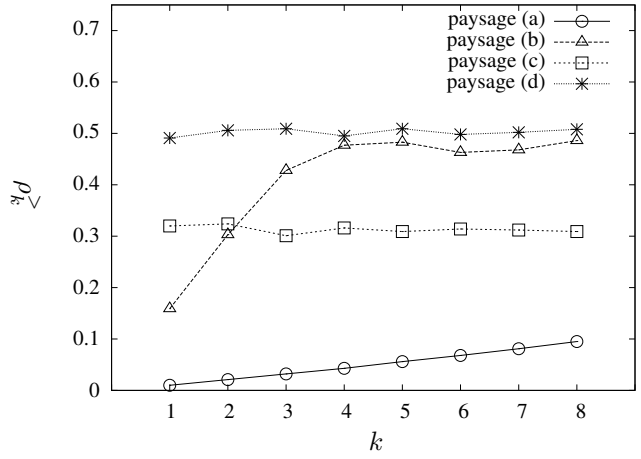


FIGURE 2.5 – k-rugosité des paysages de recherche de la figure 2.3. Lorsque le paysage est lisse (a), les rugosités calculées sont petites et augmentent faiblement avec k. Dans le paysage lisse aux multiples pics (b), la rugosité locale ($k = 1$) est faible mais augmente rapidement lorsque k augmente. Inversement, le paysage localement rugueux mais constitué essentiellement d'un seul pic (c) présente une 1-rugosité forte, mais cette rugosité n'augmente pas ou peu lorsque k augmente. Naturellement, un paysage de recherche aléatoire présente des valeurs de k-rugosité toutes autour de $0{,}5$.

été discutée par Smith et al. [Smith *et al.*, 2002] pour la neutralité ; nous reviendrons sur cet aspect plus tard dans le document.

Dans la section suivante, nous allons estimer les rugosités et neutralités de différents paysages de recherches issus de plusieurs problèmes d'optimisation combinatoire. Dans le chapitre suivant, nous essaierons de déterminer une possible corrélation entre les caractéristiques des paysages de recherche et la capacité des différents climbers à explorer ceux-ci efficacement.

2.2.2 Analyse de paysages de recherche

Dans cette section, différents problèmes d'optimisation combinatoire sont présentés, ainsi que les paysages de recherches engendrés par les opérateurs de voisinages ad hoc pour ces problèmes. Nous avons réalisé une analyse des caractéristiques des paysages de différentes instances de ces problèmes, afin d'éventuellement mettre en correspondance ces caractéristiques avec l'efficacité des différentes méthodes de recherches éprouvées par la suite. On s'intéresse ici aux paysages de grande taille, dont les solutions ne peuvent pas être énumérées : les propriétés de neutralité et de rugosité ont donc été systématiquement évaluées par échantillonnage de l'espace de recherche (1 million de configurations). Contrairement à ce que pourrait laisser penser la figure 2.5, l'interprétation de la k-rugosité n'est pas aisée et nous n'avons pas su encore comment extraire des conclusions pertinentes d'un ensemble de k-rugosités. Dans ce qui suit, nous nous concentrons sur la 1-rugosité qui est facilement interprétable. Aussi, le terme *rugosité* fera référence à la 1-rugosité.

NK-landscapes

Les paysages NK [Kauffman, 1993] ont été proposés dans le but de générer des paysages combinatoires artificiels ayant des tailles et rugosités facilement réglables. Ces paysages de recherches ne sont donc pas associés à un problème d'optimisation existant mais sont justement intéressants pour comprendre le comportement des méthodes d'optimisation selon les caractéristiques des paysages étudiés.

L'espace de recherche d'un paysage NK est constitué de configurations formées de chaînes de bits, et l'opérateur *bit-flip* est utilisé comme opérateur de voisinage. Les caractéristiques d'un paysage NK sont déterminées par deux paramètres N et K. N est la longueur de la chaîne de bits, et $K \in \{0,\ldots,N-1\}$ spécifie le degré d'interdépendance des variables, qui affecte directement la rugosité du paysage de recherche. En faisant varier K de 0 à $N-1$, on arrive à générer des paysages pouvant être de très lisses à très rugueux. Lorsque $K = 0$, le paysage ne possède qu'un seul optimum local (l'optimum global) ; lorsque $K = N-1$, les fitness des solutions du paysage engendré sont complètement aléatoires.

On cherche à maximiser la fonction $F_{\mathrm{NK}} : \{0,1\}^N \to [0,1)$, définie comme suit :

$$F_{\mathrm{NK}}(x) = \frac{1}{N}\sum_{i=1}^{N} C_i(x_i,\Pi_i(x)) \tag{2.4}$$

x_i est le i-ème bit d'une configuration x ($1 \leqslant i \leqslant N$). La fonction Π_i définit les dépendances du bit i, avec $\Pi_i(x) = \{x_{\pi_1(i)},\ldots,x_{\pi_K(i)}\}$ tel que $\pi_j(i) \in \{1,\ldots,N\} \setminus \{i\}$ et $|\bigcup_{j=1}^{K} \pi_j(i)| = K$. La fonction $C_i : \{0,1\}^{K+1} \to [0,1)$ définit la contribution de x_i par rapport à ses dépendances $\Pi_i(x)$.

Une instance NK est déterminée par les $(K+1)$-uplets $(x_i,x_{\pi_1(i)},\ldots,x_{\pi_K(i)})$ et une matrice de contributions C donnant les $2^N \times (K+1)$ valeurs de contribution possibles. Dans les instances NK standard, les dépendances et les valeurs de contributions sont générées aléatoirement de manière uniforme.

Nous avons évalué les différentes caractéristiques de 16 paysages NK (combinaisons de $N = \{128,256,512,1024\}$ et $K = \{1,2,4,8\}$). Les résultats sont reportés dans la table 2.2.

L'indicateur de rugosité donne à peu près les résultats attendus : lorsque l'on double la valeur de K, la rugosité évaluée est un peu plus que doublée. De même, pour un K fixé, lorsque l'on double la valeur de N, la rugosité est à peu près divisée par deux. Cela ne veut pas dire que le problème est plus simple, puisque l'espace de recherche engendré est aussi beaucoup plus grand.

La neutralité des paysages NK est globalement nulle. En effet, comme les contributions de C sont générés aléatoirement dans l'intervalle $[0,1)$, il y a peu de chances que deux solutions voisines aient le même fitness et donc les taux de neutralité calculés sont toujours nuls ou négligeables.

Paysages NK{pqr}

Plusieurs approches ont été proposées afin de pouvoir contrôler la neutralité dans les paysages NK : les paysages NKp [Barnett, 1998], les paysages NKq [Newman et Engelhardt, 1998] et les paysages NKr que nous avons proposé dans [Basseur et Goëffon, 2013].

N_K	Taille	Connectivité	Neutralité	Rugosité
128_1	2^{128}	128	0%	0,52%
128_2	2^{128}	128	0%	1,24%
128_4	2^{128}	128	0%	3,12%
128_8	2^{128}	128	0%	7,40%
256_1	2^{256}	256	0%	0,26%
256_2	2^{256}	256	0%	0,61%
256_4	2^{256}	256	0%	1,57%
256_8	2^{256}	256	0%	3,98%

N_K	Taille	Connectivité	Neutralité	Rugosité
512_1	2^{512}	512	0%	0,12%
512_2	2^{512}	512	0%	0,32%
512_4	2^{512}	512	0%	0,80%
512_8	2^{512}	512	0%	2,06%
1024_1	2^{1024}	1024	0%	0,07%
1024_2	2^{1024}	1024	0%	0,14%
1024_4	2^{1024}	1024	0%	0,39%
1024_8	2^{1024}	1024	0%	1,05%

TABLE 2.2 – Caractéristiques des paysages NK.

Paysages NKp Les paysages NK *probabilistes* [Barnett, 1998] sont des instances NK dont la matrice de contributions contient de nombreux zéros. Lors de la génération d'une instance NKp, chaque valeur de C a une probabilité p d'être définie à 0, sinon la valeur reste définie aléatoirement sur l'intervalle $[0,1)$. Un paysage NKp est alors déterminé par ses trois paramètres N, K et p. Lorsque $p = 0$, on retrouve les paysages NK classiques ; lorsque $p = 1$, le paysage de recherche est complètement plat.

Paysages NKq Newman et Engelhardt ont introduit les paysages NK *quantiques*, qui consistent à discrétiser plus ou moins les contributions c_i [Newman et Engelhardt, 1998]. Lorsque le nombre de valeurs possibles pour chaque c_i est faible, cela augmente naturellement la probabilité d'avoir des voisins neutres. Cette discrétisation est contrôlée par l'intermédiaire d'un paramètre $q \geqslant 2$ qui spécifie la taille du codomaine des fonctions c_i. Le taux de neutralité maximal est obtenu lorsque $q = 2$, C est alors une matrice binaire (on pourrait définir également les paysages NKq pour $q = 1$, mais les paysages engendrés seraient complètement plats et ne présenteraient donc pas d'intérêt réel). Le taux de neutralité diminue lorsque q augmente.

Paysages NKr Nous avons introduit les paysages NKr après avoir observé les spécificités particulières des paysages NKp et NKq. Ces aspects sont discutés dans la section suivante, lors de l'analyse des paysages NKp, NKq et NKr.

Dans les paysages NKr, la neutralité est créée artificiellement en arrondissant la fonction de fitness du modèle NK classique. Cela induit que les paysages NKr correspondent aux paysages NK, mais où les fitness possibles ont été divisés en un nombre prédéfini de niveaux. Contrairement aux paysages NKp et NKq, les paysages NKr ne sont pas une restriction des paysages NK, où la matrice de contribution est construite à partir d'un paramètre de neutralité. Dans le cas des paysages NKr, le fitness est modifié *a posteriori*, sans que la matrice de contribution soit modifiée :

$$\mathfrak{f}(x,r) = \frac{\lfloor r.\mathfrak{f}(x) \rfloor}{r} \tag{2.5}$$

Chaque paysage NKr est par conséquent une version lissée du paysage NK correspondant. En particulier, $\mathfrak{f}(x_1) < \mathfrak{f}(x_2) \Rightarrow \mathfrak{f}(x_1,r) \leqslant \mathfrak{f}(x_2,r)$, ce qui implique que chaque optimum global d'une instance NK est aussi un optimum global du paysage NKr correspondant (l'inverse n'est pas vrai).

N_K_p	Neutralité	Rugosité
128_1_p50	11,23%	0,38%
128_1_p80	44,55%	0,17%
128_1_p95	81,75%	0,03%
128_1_p99	94,92%	0,00%
128_2_p50	5,57%	1,04%
128_2_p80	30,85%	0,47%
128_2_p95	73,62%	0,08%
128_2_p99	94,40%	0,01%
128_4_p50	1,17%	2,98%
128_4_p80	14,52%	1,79%
128_4_p95	60,50%	0,29%
128_4_p99	89,66%	0,01%
128_8_p50	0,05%	7,31%
128_8_p80	3,42%	5,78%
128_8_p95	41,32%	1,51%
128_8_p99	83,39%	0,09%
256_1_p50	11,91%	0,20%
256_1_p80	45,09%	0,08%
256_1_p95	81,46%	0,01%
256_1_p99	96,11%	0,00%
256_2_p50	5,40%	0,53%
256_2_p80	30,82%	0,28%
256_2_p95	74,86%	0,03%
256_2_p99	94,22%	0,00%
256_4_p50	1,18%	1,53%
256_4_p80	15,19%	0,93%
256_4_p95	60,85%	0,15%
256_4_p99	90,77%	0,01%
256_8_p50	0,05%	3,85%
256_8_p80	3,55%	3,01%
256_8_p95	41,25%	0,74%
256_8_p99	83,70%	0,05%
512_1_p50	12,17%	0,10%
512_1_p80	45,27%	0,04%
512_1_p95	81,69%	0,00%
512_1_p99	96,29%	0,00%
512_2_p50	5,39%	0,27%
512_2_p80	31,18%	0,12%
512_2_p95	74,31%	0,02%
512_2_p99	94,57%	0,00%
512_4_p50	1,23%	0,75%
512_4_p80	15,24%	0,47%
512_4_p95	61,17%	0,07%
512_4_p99	90,19%	0,01%
512_8_p50	0,05%	2,00%
512_8_p80	3,51%	1,54%
512_8_p95	41,40%	0,38%
512_8_p99	83,64%	0,02%
1024_1_p50	12,00%	0,05%
1024_1_p80	44,70%	0,02%
1024_1_p95	81,67%	0,00%
1024_1_p99	96,10%	0,00%
1024_2_p50	5,62%	0,14%
1024_2_p80	31,46%	0,07%
1024_2_p95	74,31%	0,01%
1024_2_p99	94,15%	0,00%
1024_4_p50	1,23%	0,37%
1024_4_p80	15,11%	0,24%
1024_4_p95	61,24%	0,04%
1024_4_p99	90,72%	0,00%
1024_8_p50	0,07%	1,01%
1024_8_p80	3,56%	0,78%
1024_8_p95	41,27%	0,19%
1024_8_p99	83,62%	0,01%

N_K_q	Neutralité	Rugosité
128_1_q10	6,35%	0,43%
128_1_q5	15,21%	0,29%
128_1_q3	24,38%	0,23%
128_1_q2	41,12%	0,06%
128_2_q10	4,76%	1,07%
128_2_q5	12,42%	0,75%
128_2_q3	21,32%	0,47%
128_2_q2	33,33%	0,25%
128_4_q10	2,91%	2,77%
128_4_q5	9,37%	1,98%
128_4_q3	16,21%	1,33%
128_4_q2	26,16%	0,57%
128_8_q10	1,75%	6,79%
128_8_q5	6,85%	4,88%
128_8_q3	11,80%	3,31%
128_8_q2	19,23%	1,68%
256_1_q10	6,62%	0,24%
256_1_q5	15,26%	0,17%
256_1_q3	25,03%	0,12%
256_1_q2	39,88%	0,04%
256_2_q10	4,73%	0,56%
256_2_q5	12,21%	0,39%
256_2_q3	21,29%	0,25%
256_2_q2	33,26%	0,12%
256_4_q10	2,99%	1,38%
256_4_q5	9,44%	0,97%
256_4_q3	16,18%	0,65%
256_4_q2	26,15%	0,27%
256_8_q10	1,78%	3,60%
256_8_q5	6,85%	2,52%
256_8_q3	11,84%	1,69%
256_8_q2	19,26%	0,77%
512_1_q10	6,70%	0,11%
512_1_q5	15,19%	0,08%
512_1_q3	26,07%	0,05%
512_1_q2	40,16%	0,02%
512_2_q10	4,65%	0,28%
512_2_q5	12,35%	0,19%
512_2_q3	21,03%	0,12%
512_2_q2	33,47%	0,06%
512_4_q10	3,02%	0,71%
512_4_q5	9,36%	0,48%
512_4_q3	16,14%	0,32%
512_4_q2	26,15%	0,14%
512_8_q10	1,80%	1,87%
512_8_q5	6,84%	1,28%
512_8_q3	11,84%	0,85%
512_8_q2	19,25%	0,37%
1024_1_q10	6,55%	0,06%
1024_1_q5	15,23%	0,04%
1024_1_q3	25,87%	0,03%
1024_1_q2	40,51%	0,01%
1024_2_q10	4,66%	0,15%
1024_2_q5	12,33%	0,09%
1024_2_q3	21,22%	0,06%
1024_2_q2	33,65%	0,03%
1024_4_q10	3,02%	0,36%
1024_4_q5	9,40%	0,25%
1024_4_q3	16,18%	0,16%
1024_4_q2	26,18%	0,07%
1024_8_q10	1,78%	0,95%
1024_8_q5	6,85%	0,65%
1024_8_q3	11,85%	0,43%
1024_8_q2	19,27%	0,18%

N_K_r	Neutralité	Rugosité
128_1_r10000	0,80%	0,52%
128_1_r1000	9,38%	0,38%
128_1_r100	65,57%	0,03%
128_1_r10	93,68%	0,00%
128_2_r10000	0,66%	1,20%
128_2_r1000	7,72%	0,90%
128_2_r100	58,47%	0,08%
128_2_r10	93,07%	0,01%
128_4_r10000	0,59%	3,05%
128_4_r1000	5,85%	2,40%
128_4_r100	49,50%	0,23%
128_4_r10	91,13%	0,02%
128_8_r10000	0,43%	7,24%
128_8_r1000	4,27%	5,82%
128_8_r100	39,01%	0,67%
128_8_r10	88,16%	0,04%
256_1_r10000	1,82%	0,25%
256_1_r1000	17,86%	0,15%
256_1_r100	82,23%	0,01%
256_1_r10	95,85%	0,00%
256_2_r10000	1,59%	0,59%
256_2_r1000	15,09%	0,35%
256_2_r100	78,39%	0,01%
256_2_r10	95,20%	0,00%
256_4_r10000	1,18%	1,48%
256_4_r1000	11,73%	0,88%
256_4_r100	72,10%	0,03%
256_4_r10	93,68%	0,01%
256_8_r10000	0,86%	3,76%
256_8_r1000	8,57%	2,30%
256_8_r100	62,97%	0,08%
256_8_r10	91,56%	0,01%
512_1_r10000	3,97%	0,11%
512_1_r1000	34,48%	0,04%
512_1_r100	91,18%	0,00%
512_1_r10	97,09%	0,00%
512_2_r10000	3,08%	0,28%
512_2_r1000	28,95%	0,10%
512_2_r100	89,20%	0,00%
512_2_r10	96,49%	0,00%
512_4_r10000	2,35%	0,70%
512_4_r1000	22,73%	0,24%
512_4_r100	86,00%	0,01%
512_4_r10	95,59%	0,00%
512_8_r10000	1,72%	1,84%
512_8_r1000	16,90%	0,63%
512_8_r100	81,05%	0,01%
512_8_r10	94,02%	0,01%
1024_1_r10000	7,53%	0,05%
1024_1_r1000	58,10%	0,01%
1024_1_r100	95,57%	0,00%
1024_1_r10	97,88%	0,00%
1024_2_r10000	6,17%	0,12%
1024_2_r1000	50,82%	0,02%
1024_2_r100	94,65%	0,00%
1024_2_r10	97,54%	0,00%
1024_4_r10000	4,71%	0,32%
1024_4_r1000	41,83%	0,04%
1024_4_r100	92,95%	0,00%
1024_4_r10	96,93%	0,00%
1024_8_r10000	3,44%	0,84%
1024_8_r1000	32,29%	0,11%
1024_8_r100	90,48%	0,00%
1024_8_r10	95,80%	0,00%

TABLE 2.3 – Neutralité et rugosité des paysages NKp, NKq et NKr.

Analyse des paysages NK{pqr} Nous avons généré 192 paysages définis de la manière suivante :

- NKp : 64 paysages NKp, dérivés des 16 paysages NK basiques étudiés précédemment, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de p (0,50, 0,80, 0,95 et 0,99).
- NKq : 64 paysages NKq, dérivés des mêmes 16 paysages NK basiques, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de q (2, 3, 5 et 10).
- NKr : 64 paysages NKr, dérivés des mêmes 16 paysages NK basiques, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de r (10, 100, 1000 et 10000).

Les tailles de ces paysages ainsi que leurs connectivités sont naturellement identiques aux paysages NK correspondants. Les taux de rugosité et de neutralité estimés des paysages NKp, NKq et NKr sont reportés dans la table 2.3.

Générer des paysages NKp à fort taux de neutralité implique d'utiliser une valeur de p proche de 1. Cela est plutôt problématique puisque la matrice C contient alors essentiellement des 0. Optimiser un tel paysage devient alors beaucoup plus facile, puisque cela revient alors principalement à trouver les valeurs différentes de 0 dans C. Les paysages NKq n'impliquent pas ce genre de problème, cependant il est impossible de générer des paysages à fort taux de neutralité (la valeur maximale est de 41,12% pour l'instance 128_1_q2). Le modèle NKr permet de générer des paysages à fort taux de neutralité, tout en préservant la complexité du problème sous-jacent. Enfin, pour tous ces modèles, ajouter de la neutralité implique naturellement une baisse de la rugosité. En effet, il n'y a pas d'épistasie de signe si deux solutions voisines d'une 1-échelle sont de même fitness. Cela pousse à penser qu'ajouter artificiellement de la neutralité pourrait rendre les problèmes plus faciles à résoudre.

Flow shop de permutation

Les problèmes d'ordonnancement de type flow shop sont très étudiés dans la littérature. Ce problème est naturellement multiobjectif, mais est souvent étudié sous la forme mono-objectif, le critère le plus couramment étudié étant la date de fin d'ordonnancement. Il existe de nombreuses variantes de ce problème, nous allons nous restreindre ici à une version classique.

Une instance de flow shop sera représentée par :

- Un nombre n de jobs J_1 à J_n à ordonnancer.
- Un nombre m de machines sur lesquelles les jobs doivent être traités, dans l'ordre imposé de m_1 à m_m.
- Un ensemble de $n * m$ taches t_{ij} à réaliser, t_{ij} correspondant au temps nécessaire au traitement du job J_i sur la machine m_j.

Les machines sont des ressources critiques, c'est-à-dire que deux jobs ne peuvent être affectés simultanément à une machine. Comme dans la plupart des travaux, nous nous restreindrons aux solutions où les jobs sont traités dans le même ordre sur toutes les machines (flow shop *de permutation*).

Le flow shop est majoritairement traité comme un problème mono-objectif et l'on cherche principalement à minimiser C_{max}, souvent appelé *makespan*, c'est-à-dire la date d'achèvement de l'ordonnancement. Chaque tâche étant ordonnancée à la date s_{ij} définie par rapport à la permuta-

tion des jobs considérée, le *makespan* se calcule comme suit :

$$C_{\max}(\pi) = \max_{i \in [1...n]} \{s_{im} + t_{im}\} \tag{2.6}$$

Notons que pour m machines (m>2), le problème de flow shop présenté ci-dessus est fortement NP-difficile, d'après les résultats de [Lenstra *et al.*, 1977].

L'espace de recherche pour ce problème est l'ensemble des permutations de $\{1, \ldots, n\}$ que nous appellerons S_n. La taille de l'espace de recherche vaut $|S_n| = n!$. L'opérateur de voisinage le plus efficace pour résoudre ce problème est l'opérateur d'*insertion* ($\mathcal{N}_{\text{ins}}$), qui consiste à déplacer un job de la permutation à une position différente. La taille du voisinage engendré ne dépend pas de la permutation et vaut $|\mathcal{N}_{\text{ins}}(\pi)| = (n-1)^2, \forall \pi \in S_n$. Ceci nous permet de définir ce problème en tant que paysage de recherche $\text{FSP}_{\text{insert}}$ qui correspond au triplet $(S_n, \mathcal{N}_{\text{ins}}, C_{\max})$.

Instance	Neut.	Rug.
20_5_01	21,54%	10,10%
20_10_01	9,85%	11,94%
20_15_01	8,89%	12,31%
20_20_01	5,65%	13,22%
30_5_01	24,68%	7,43%
30_10_01	11,79%	9,99%
30_15_01	8,72%	10,59%

Instance	Neut.	Rug.
30_20_01	5,25%	11,47%
50_5_01	33,00%	4,89%
50_10_01	13,77%	7,63%
50_15_01	8,88%	8,37%
50_20_01	7,09%	9,06%
70_5_01	33,84%	4,14%
70_10_01	15,55%	6,22%

Instance	Neut.	Rug.
70_15_01	10,23%	7,22%
70_20_01	7,96%	7,73%
100_5_01	32,08%	4,00%
100_10_01	17,65%	5,26%
100_15_01	11,18%	6,01%
100_20_01	9,04%	6,55%
150_5_01	35,59%	2,54%

Instance	Neut.	Rug.
150_10_01	17,82%	4,30%
150_15_01	12,63%	4,84%
150_20_01	9,57%	5,47%
200_10_01	20,45%	3,56%
200_15_01	13,17%	4,27%
200_20_01	10,71%	4,73%

TABLE 2.4 – Neutralité et rugosité de paysages $\text{FSP}_{\text{insert}}$ dérivés d'instances de flow shop. Rappel : le premier nombre figurant dans le nom de l'instance est n, qui définit la taille de l'espace de recherche ($n!$) et la connectivité du graphe de transition ($(n-1)^2$).

Les jeux d'instances les plus utilisés pour le flow shop mono-objectif sont certainement ceux proposés par Taillard [Taillard, 1993], que nous avons utilisés pour nos expérimentations ($n = \{20,30,50,70,100,150,200\}$, $m = \{5,10,15,20\}$).

La table 2.4 contient les caractéristiques des paysages associés aux instances de flow shop étudiées dans ce document. Ces instances sont caractérisées par un taux de rugosité relativement élevé et la présence significative de neutralité. Globalement, la rugosité est plus importante pour un nombre de machines élevé, tandis que la neutralité diminue. Par contre, lorsque le nombre de jobs est plus important, la rugosité diminue et la neutralité augmente.

QAP

Le problème d'affectation quadratique (QAP) est considéré, parmi les problèmes NP-difficiles, comme étant très difficile à résoudre [Sahni et Gonzalez, 1976, Pardalos et Wolkowicz, 1994] et est utilisé dans la modélisation de nombreux problèmes de la vie réelle. Le QAP consiste à placer n unités sur n sites, étant donné une matrice D de distances entre les sites et une matrice F de flux entre les unités. Soient d_{ij} la distance entre les sites i et j et f_{rs} le flux entre les unités r et s, une solution de ce problème est une permutation π décrivant les affectations des unités aux sites. La fonction objectif correspond alors au produit des flux par les distances :

$$W_{\text{QAP}}(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f_{\pi_i \pi_j} \tag{2.7}$$

π_i correspond à l'emplacement de l'unité i dans la configuration $\pi \in S_n$. Nous cherchons à minimiser W_{QAP}.

L'opérateur de voisinage couramment utilisé pour optimiser le QAP est l'*échange* ($\mathcal{N}_{\mathrm{swap}}$), qui consiste à échanger l'affectation des sites de deux unités. La taille du voisinage est constante pour toute configuration et vaut $|\mathcal{N}_{\mathrm{swap}}| = \binom{2}{n}, \forall \pi \in S_n$. Le paysage de recherche $\mathrm{QAP}_{\mathrm{swap}}$ associé est donc défini par le triplet $(S_n, \mathcal{N}_{\mathrm{swap}}, W_{\mathrm{QAP}})$.

Instance	Neut.	Rug.
bur26a	1,03%	9,65%
bur26b	2,04%	9,65%
bur26c	1,07%	9,85%
bur26d	2,13%	9,82%
bur26e	1,12%	9,65%
bur26f	2,27%	9,65%
bur26g	1,01%	9,59%
bur26h	1,99%	9,59%
chr12a	0,06%	19,86%
chr12b	0,16%	19,40%
chr12c	0,04%	19,96%
chr15a	0,05%	16,40%
chr15b	0,08%	15,91%
chr15c	0,03%	16,56%
chr18a	0,03%	14,16%
chr18b	2,51%	13,50%
chr20a	0,64%	13,11%
chr20b	0,35%	13,30%
chr20c	0,14%	12,91%
chr22a	0,27%	11,91%
chr22b	0,21%	12,01%
chr25a	0,32%	10,63%
els19	0,40%	12,77%
esc16a	30,06%	14,75%
esc16b	38,94%	12,35%
esc16c	16,66%	16,10%
esc16d	23,99%	15,00%
esc16e	39,10%	12,83%
esc16g	38,76%	12,73%
esc16h	29,29%	14,41%
esc16i	28,95%	13,53%
esc16j	45,27%	11,08%
esc32a	12,04%	9,53%

Instance	Neut.	Rug.
esc32b	17,67%	10,26%
esc32c	27,67%	8,87%
esc32d	32,56%	8,31%
esc32e	62,15%	3,55%
esc32f	62,18%	3,53%
esc32g	67,70%	3,32%
esc32h	25,74%	8,91%
esc64a	54,71%	2,81%
had12	3,94%	18,93%
had14	3,44%	17,38%
had16	3,13%	16,67%
had18	2,64%	15,27%
had20	2,43%	14,16%
kra30a	2,99%	10,60%
kra30b	2,95%	10,55%
kra32	3,11%	9,95%
lipa20a	2,68%	15,00%
lipa20b	0,10%	16,47%
lipa30a	1,98%	10,85%
lipa30b	0,04%	12,98%
lipa40a	1,31%	8,52%
lipa40b	0,02%	10,87%
lipa50a	1,05%	7,01%
lipa50b	0,01%	9,51%
lipa60a	0,84%	5,91%
lipa60b	0,01%	8,61%
lipa70a	0,72%	5,16%
lipa70b	0,00%	7,82%
lipa80a	0,62%	4,54%
lipa80b	0,00%	7,26%
lipa90a	0,56%	4,09%
lipa90b	0,00%	6,77%
nug12	3,25%	21,24%

Instance	Neut.	Rug.
nug14	2,69%	18,79%
nug15	2,57%	18,25%
nug16a	2,23%	17,37%
nug16b	2,33%	17,25%
nug17	2,21%	16,42%
nug18	2,12%	15,93%
nug20	1,85%	14,86%
nug21	1,90%	14,26%
nug22	1,39%	13,42%
nug24	1,71%	13,21%
nug25	1,81%	12,94%
nug27	1,37%	12,14%
nug28	1,46%	12,00%
nug30	1,39%	11,55%
rou12	0,01%	22,52%
rou15	0,01%	19,75%
rou20	0,01%	16,36%
scr12	0,26%	19,32%
scr15	0,17%	16,16%
scr20	0,09%	13,38%
sko100a	0,44%	5,63%
sko100b	0,43%	5,63%
sko100c	0,45%	5,72%
sko100d	0,44%	5,73%
sko100e	0,43%	5,67%
sko100f	0,45%	5,71%
sko42	0,99%	9,61%
sko49	0,91%	8,78%
sko56	0,72%	8,04%
sko64	0,69%	7,50%
sko72	0,59%	6,90%
sko81	0,53%	6,40%
sko90	0,49%	6,03%

Instance	Neut.	Rug.
ste36a	0,58%	8,06%
ste36b	0,28%	7,91%
ste36c	0,17%	7,94%
tai100a	0,00%	6,39%
tai100b	0,00%	4,42%
tai10a	0,01%	24,63%
tai10b	0,00%	22,67%
tai12a	0,01%	21,65%
tai12b	0,00%	19,02%
tai15a	0,01%	19,89%
tai15b	0,00%	15,91%
tai17a	0,01%	18,42%
tai20a	0,01%	16,58%
tai20b	0,00%	12,91%
tai25a	0,00%	14,64%
tai25b	0,00%	11,15%
tai30a	0,01%	12,97%
tai30b	0,00%	10,29%
tai35a	0,00%	11,89%
tai35b	0,00%	9,42%
tai40a	0,00%	11,03%
tai40b	0,00%	8,87%
tai50a	0,00%	9,63%
tai50b	0,00%	7,53%
tai60a	0,00%	8,63%
tai60b	0,00%	6,25%
tai64c	67,12%	2,27%
tai80a	0,00%	7,27%
tai80b	0,00%	5,25%
tho30	0,04%	11,59%
tho40	0,04%	9,44%
wil100	0,43%	5,74%
wil50	0,79%	8,75%

TABLE 2.5 – Neutralité et rugosité de paysages $\mathrm{QAP}_{\mathrm{swap}}$ dérivés d'instances de QAP. Le nombre n figurant dans le nom de l'instance définit la taille de l'espace de recherche ($n!$) et la connectivité du graphe de transition ($n(n-1)/2$).

Nous avons étudié le paysage de 132 instances de QAP, les résultats étant reportés dans la table 2.5. Comme le laissent penser les études mettant en exergue la difficulté du QAP, les taux de rugosité constatés sur les différentes instances sont relativement élevés. En ce qui concerne les taux de neutralité, ils varient de très faible à très important.

MAXSAT

Le problème de satisfiabilité (SAT) consiste à déterminer une instanciation de n variables booléennes ($x_i \in \mathcal{B} = \{\bot, \top\}$) qui satisfasse une expression booléenne formulée sous forme normale conjonctive. Ce problème de décision a été le premier prouvé comme étant NP-difficile

[Cook, 1971] ; depuis de nombreux problèmes sont prouvés comme étant NP-difficiles par réduction polynomiale en problème SAT. Le problème d'optimisation MAXSAT consiste à satisfaire un maximum de clauses disjonctives d'une formule booléenne. Une *clause* $c_j = \bigvee_{k=1}^{n_j} l_{jk}$ est une disjonction de littéraux l_{jk}, un *littéral* étant une variable booléenne (x_i) ou sa négation ($\neg x_i$).

Étant donné une instance MAXSAT $\{c_1, \ldots, c_m\}$ avec n variables et m clauses, la fonction objectif d'une instanciation $x \in \mathcal{B}^n$ est donnée par :

$$V(x) = \sum_{j=1}^{m} v(x^*(c_j)), \text{ avec } v(x^*(c_j)) = \begin{cases} 1, \text{si } c_j = \top \text{ relativement à } x \\ 0, \text{sinon} \end{cases} \tag{2.8}$$

La plupart du temps, MAXSAT est modélisé comme un problème de minimisation où l'on cherche à réduire au maximum le nombre $m - V(x)$ de clauses non satisfaites. Par conséquent, lors de la présentation des résultats sur ce problème, les valeurs correspondront à cette version du problème, ce qui est pratique car la solution optimale est toujours 0 pour les instances satisfiables (ce qui est le cas de celles présentées dans ce document, sauf pour les instances de type uuf*).

La représentation usuelle des configurations pour le MAXSAT est généralement sous forme de chaine de bits ($0 \equiv \bot, 1 \equiv \top$). Le voisinage naturel pour cette représentation est le bit-flip ($\mathcal{N}_{\text{flip}}$), comme pour les paysages NK. Les paysages NK et MAXSAT ont des graphes de transitions équivalents, par conséquent un paysage MAXSAT sera défini par le triplet $(\{0,1\}^n, \mathcal{N}_{\text{flip}}, V)$, la taille de l'espace de recherche étant de 2^n et la taille du voisinage n.

Nous avons utilisé 129 instances SAT de SATLIB et avons reporté les résultats dans la table 2.6. Les taux de rugosité observés pour MAXSAT sont globalement assez faibles. C'est une des raisons pour laquelle on est généralement capables de résoudre des instances de grandes tailles en comparaison au QAP ou même au flow shop. Les taux de neutralités varient entre 22% et 25% pour les instances MAXSAT aléatoires (de la première à CBS_...m449_b90_0), hormis pour l'instance BMS_...m429_0 qui affiche un taux de neutralité de 33,46%. Pour les instances structurées, issue d'instanciations SAT de problèmes combinatoires classiques, la neutralité est parfois à priori inexistante (même si ce n'est pas forcément le cas, ce que nous verrons plus loin).

2.2.3 Atteignabilité de l'optimum global

Les algorithmes de recherche locale, et plus particulièrement les climbers, sont souvent considérés comme des méthodes d'intensification de la recherche qui se focalisent sur une petite portion de l'espace de recherche, leur efficacité étant alors étroitement liée au choix de la configuration initiale. Il est souvent admis que l'optimum global est souvent trop éloigné de la zone de recherche et qu'il faut réinitialiser régulièrement la recherche ou faire intervenir un mécanisme de diversification afin d'avoir une chance de pouvoir l'atteindre. Pourtant, dans la plupart des paysages de recherche, le voisinage induit un graphe de transition au diamètre restreint. Par exemple, en considérant un paysage constitué de chaines de bits de longueur N, l'opérateur de voisinage classique 1-flip relie l'ensemble des configurations via un hypercube de dimension N, N étant donc le diamètre du graphe de transition. La plupart des solutions dans un tel paysage de recherche se trouvent à une distance proche de $N/2$ de l'optimum global (une seule solution se trouve à une distance N de l'optimum global). Ce nombre est relativement petit en comparaison au nombre de pas réalisés lors d'une recherche locale ou même lors de l'exécution d'un climber.

Instance	Neut.	Rug.
uf20-01	20,93%	1,81%
uf50-01	22,53%	0,31%
uuf50-01	24,08%	0,31%
uf75-01	23,67%	0,18%
uuf75-01	23,69%	0,15%
uf100-01	23,23%	0,09%
uuf100-01	23,84%	0,08%
uf125-01	23,46%	0,05%
uuf125-01	23,77%	0,05%
uf150-01	23,12%	0,03%
uuf150-01	23,47%	0,03%
uf175-01	22,91%	0,03%
uuf175-01	23,27%	0,03%
uf200-01	23,15%	0,02%
uuf200-01	23,67%	0,03%
uf225-01	23,11%	0,02%
uuf225-01	23,37%	0,02%
uf250-01	22,92%	0,01%
uuf250-01	23,46%	0,01%
RTI_...m429_0	23,32%	0,08%
BMS_...m429_0	33,46%	0,03%
CBS_...m403_b10_0	24,17%	0,07%
CBS_...m403_b30_0	24,18%	0,08%
CBS_...m403_b50_0	24,20%	0,07%
CBS_...m403_b70_0	24,27%	0,07%
CBS_...m403_b90_0	25,10%	0,09%
CBS_...m411_b10_0	23,48%	0,08%
CBS_...m411_b30_0	24,42%	0,07%
CBS_...m411_b50_0	24,71%	0,05%
CBS_...m411_b70_0	24,71%	0,06%
CBS_...m411_b90_0	24,32%	0,06%
CBS_...m418_b10_0	23,41%	0,08%
CBS_...m418_b30_0	23,25%	0,09%
CBS_...m418_b50_0	23,37%	0,08%
CBS_...m418_b70_0	23,37%	0,11%
CBS_...m418_b90_0	23,71%	0,08%
CBS_...m423_b10_0	23,05%	0,09%
CBS_...m423_b30_0	22,94%	0,09%
CBS_...m423_b50_0	23,32%	0,09%
CBS_...m423_b70_0	22,73%	0,09%
CBS_...m423_b90_0	23,32%	0,10%
CBS_...m429_b10_0	22,84%	0,08%
CBS_...m429_b30_0	22,71%	0,08%

Instance	Neut.	Rug.
CBS_...m429_b50_0	23,20%	0,11%
CBS_...m429_b70_0	23,16%	0,09%
CBS_...m429_b90_0	23,16%	0,10%
CBS_...m435_b10_0	22,90%	0,09%
CBS_...m435_b30_0	23,05%	0,09%
CBS_...m435_b50_0	22,51%	0,08%
CBS_...m435_b70_0	23,46%	0,09%
CBS_...m435_b90_0	22,90%	0,09%
CBS_...m441_b10_0	22,33%	0,07%
CBS_...m441_b30_0	22,62%	0,09%
CBS_...m441_b50_0	22,71%	0,09%
CBS_...m441_b70_0	22,07%	0,08%
CBS_...m441_b90_0	23,27%	0,07%
CBS_...m449_b10_0	22,11%	0,08%
CBS_...m449_b30_0	22,68%	0,10%
CBS_...m449_b50_0	22,09%	0,06%
CBS_...m449_b70_0	22,65%	0,09%
CBS_...m449_b90_0	23,16%	0,07%
flat30-1	6,68%	0,09%
flat50-1	5,55%	0,06%
flat75-1	5,47%	0,03%
flat100-1	5,37%	0,02%
flat125-1	5,05%	0,02%
flat150-1	5,11%	0,02%
flat175-1	5,34%	0,01%
flat200-1	5,23%	0,01%
sw100-8-lp0-c5-1	0,46%	0,00%
sw100-8-lp1-c5-1	0,35%	0,00%
sw100-8-lp2-c5-1	0,31%	0,00%
sw100-8-lp3-c5-1	0,25%	0,00%
sw100-8-lp4-c5-1	0,22%	0,00%
sw100-8-lp5-c5-1	0,21%	0,00%
sw100-8-lp6-c5-1	0,20%	0,00%
sw100-8-lp7-c5-1	0,20%	0,00%
sw100-8-lp8-c5-1	0,20%	0,00%
sw100-8-p0-c5-1	0,19%	0,00%
bw_anomaly	9,59%	0,67%
bw_medium	5,18%	0,15%
bw_huge	1,48%	0,01%
bw_large.a	2,44%	0,01%
bw_large.b	1,19%	0,00%
bw_large.c	0,41%	0,00%
bw_large.d	0,17%	0,00%

Instance	Neut.	Rug.
logistics.a	1,94%	0,00%
logistics.b	2,55%	0,01%
logistics.c	1,71%	0,00%
logistics.d	9,04%	0,00%
ais6	1,57%	0,15%
ais8	0,57%	0,04%
ais10	0,24%	0,01%
ais12	0,10%	0,00%
qg1-07	0,00%	0,00%
qg1-08	0,00%	0,00%
qg2-07	0,00%	0,00%
qg2-08	0,00%	0,00%
qg3-08	0,01%	0,00%
qg3-09	0,00%	0,00%
qg4-08	0,02%	0,00%
qg4-09	0,01%	0,00%
qg5-09	0,01%	0,00%
qg5-10	0,00%	0,00%
qg5-11	0,00%	0,00%
qg5-12	0,00%	0,00%
qg5-13	0,00%	0,00%
qg6-09	0,05%	0,00%
qg6-10	0,02%	0,00%
qg6-11	0,01%	0,00%
qg6-12	0,00%	0,00%
qg7-09	0,00%	0,00%
qg7-10	0,00%	0,00%
qg7-11	0,00%	0,00%
qg7-12	0,00%	0,00%
qg7-13	0,00%	0,00%
bmc-ibm-1	10,22%	0,00%
bmc-ibm-2	12,84%	0,02%
bmc-ibm-3	12,64%	0,00%
bmc-ibm-4	11,77%	0,00%
bmc-ibm-5	14,08%	0,00%
bmc-ibm-6	10,35%	0,00%
bmc-ibm-7	11,17%	0,00%
bmc-galileo-8	13,68%	0,00%
bmc-galileo-9	13,62%	0,00%
bmc-ibm-10	10,69%	0,00%
bmc-ibm-11	13,90%	0,00%
bmc-ibm-12	12,70%	0,00%
bmc-ibm-13	14,69%	0,00%

TABLE 2.6 – Neutralité et rugosité de paysages MAXSAT dérivés d'instances de MAXSAT. Le nombre de variables v définit la taille du paysage (2^v) et sa connectivité (v). Les informations sur le nombre de clauses et variables des instances se trouvent sur : `www.cs.ubc.ca/~hoos/SATLIB/benchm.html`. Les instances jusqu'à CBS_...m449_b90_0 sont aléatoires, les suivantes sont construites à partir d'une formulation MAXSAT de problèmes combinatoires.

Nous nous sommes intéressés à l'influence de la configuration initiale sur le comportement général des climbers. Plus précisément, nous nous sommes efforcés de savoir si le choix de cette configuration est déterminant pour savoir si l'optimum global peut être atteint, ou si les choix de conception du climber influencent majoritairement les résultats de la recherche.

Pour cela nous définissons le *taux de couverture* d'une configuration comme étant la proportion de l'espace de recherche pour laquelle cette configuration est atteignable par un climber avec une probabilité non nulle. On s'intéressera plus particulièrement au taux de couverture des optima globaux des paysages afin de savoir dans quelle mesure ils sont atteignables dans l'espace de recherche.

Bassins d'attraction et taux de couverture

Étant donnés un paysage de recherche $(\mathcal{X},\mathcal{N},f)$ et un climber, le *bassin d'attraction* d'un optimum local x_{opt} est l'ensemble des configurations $B_{x_{\text{opt}}} = \{x \in \mathcal{X}, p_{x_{\text{opt}}}(x) > 0\}$, où $p_{x_{\text{opt}}}(x)$ représente la probabilité d'atteindre x_{opt} à partir d'une configuration de départ x et en utilisant la stratégie de transition du climber [Ochoa *et al.*, 2010]. Cette définition est restreinte aux optima locaux mais est bien sûr généralisable à toute configuration de l'espace de recherche.

Dans [Ochoa *et al.*, 2010], la taille d'un bassin d'attraction $B_{x_{\text{opt}}}$ est défini comme la somme des probabilités $\sum_{x\in\mathcal{X}} p_{x_{\text{opt}}}(x)$. Cette définition des bassins d'attraction n'est pas basée uniquement sur sa taille mais aussi sur la probabilité d'atteindre un optimum local par un climber partant d'une configuration aléatoire de l'espace de recherche. Les bassins d'attraction sont donc étroitement liés aux paramètres du climber considéré. La probabilité d'atteindre un optima local vaut alors $\frac{size(B_{x_{\text{opt}}})}{\#\mathcal{X}}$.

Nous nous sommes intéressés ici à la cardinalité des bassins d'attraction dans le but d'évaluer la *possibilité* d'atteindre un optimum local donné. Cela permet de s'abstraire du climber considéré, les aspects de conception des climbers serviront ensuite à faire varier la *probabilité* d'atteindre un certain optimum local. Le *taux de couverture* $\gamma(x)$ d'une configuration $x \in \mathcal{X}$ correspond donc à la proportion de l'espace de recherche qui peut atteindre x via un climber :

$$\gamma(x) = \frac{\#B_x}{\#\mathcal{X}}$$

Taux de couverture des optima globaux des paysages NK

Nous avons cherché à calculer le taux de couverture des optima globaux sur les paysages NK basiques. Ces paysages présentent d'une part l'avantage d'être facilement paramétrables et d'autre part la représentation simple des solutions et de leur fitness permet de mettre au point des méthodes moins complexes que pour d'autres paysages issus de problèmes combinatoires. Un haut taux de couverture indique que si l'on choisit une configuration aléatoirement, il y a de grande chance qu'il soit *possible* d'atteindre l'optimum global en une simple amélioration itérative (climber).

Méthodologie Dans un premier temps il est nécessaire de calculer un optimum global pour chaque paysage NK. Comme nous nous sommes focalisés sur les petits paysages, cela a pu être réalisé par énumération des configurations possibles. À partir de là, le calcul de la couverture des

optima globaux est relativement plus complexe. Cela consiste à descendre le paysage à partir de l'optimum global considéré, en empruntant tous les chemins possibles. La génération d'un chemin s'arrête lorsqu'un minimum local est atteint. Il reste alors à compter le nombre de configurations distinctes rencontrées lors de la génération des chemins possibles. Cela s'avère extrêmement coûteux, en temps et en espace de stockage, puisque la plupart des configurations se trouvent dans un grand nombre de chemins possibles.

Afin de rendre l'algorithme d'évaluation de la couverture praticable, nous avons proposé une méthode utilisant une table de hachage basée sur le fitness des solutions, qui permet d'éviter l'exploration multiple d'une même configuration. De plus, l'exploration des chemins dans l'ordre décroissant des fitness permet de pouvoir libérer l'espace mémoire durant la recherche. L'algorithme est décrit beaucoup plus précisément dans un article que nous avons publié à la conférence SOCS [Vigneron *et al.*, 2014].

Résultats La table 2.7 présente les taux de couverture moyens obtenus sur des petits paysages NK. Pour chaque paramétrage (N,K), les valeurs de la table correspondent à la moyenne et à l'écart type observé pour 10 paysages. Les taux de couverture calculés dépassent tous 70% quel que soit le paramétrage (N,K) testé. La plus petite couverture est obtenue sur l'instance la plus petite et la plus rugueuse, mais ce taux reste relativement élevé.

Paysage	$N = 16$			$N = 24$		
	$K = 2$	$K = 4$	$K = 8$	$K = 2$	$K = 4$	$K = 8$
Moyenne	**75.69%**	**88.85%**	**70.17%**	**78.08%**	**95.61%**	**89.41%**
Écart type	14.86%	2.19%	2.82%	12.57%	1.28%	1.57%

TABLE 2.7 – Taux de couverture pour des paysages NK générés aléatoirement ($N \in \{16,24\}$ et $K \in \{2,4,8\}$).

Il est intéressant de noter que le taux de couverture de l'optimum global est toujours élevé et qu'il ne varie pas énormément selon le type de paysage considéré. De plus, ce taux tend à augmenter avec la dimension du paysage. En effet, les taux moyens calculés pour $N = 24$ sont toujours plus élevés que pour $N = 16$, quel que soit K. Cela nous encourage à croire que le taux de couverture de l'optimum global est également élevé pour des paysages NK plus grands, que les méthodes exactes ne sont pas capables de résoudre.

Pendant les expérimentations, nous avons également observé la distribution des solutions selon leur fitness et leur appartenance ou non à la couverture de l'optimum global. Grâce à cela, nous pouvons visualiser la corrélation entre le taux de couverture et la qualité des configurations (voir figure 2.6). La figure montre que l'optimum global est quasiment toujours atteignable à partir des plus mauvaises configurations, qui permettent il est vrai d'emprunter beaucoup de chemins améliorants possibles. Par contre, le taux de couverture tend à diminuer si l'on considère des configurations initiales de meilleur fitness.

Rappelons qu'un taux de couverture élevé n'indique en aucun cas que l'optimum global est facilement atteignable, mais nous permet d'évaluer la *possibilité* de l'atteindre. Dans ce qui suit, nous nous intéressons à la probabilité d'atteindre l'optimum global sachant une sélection aléatoire d'un chemin améliorant.

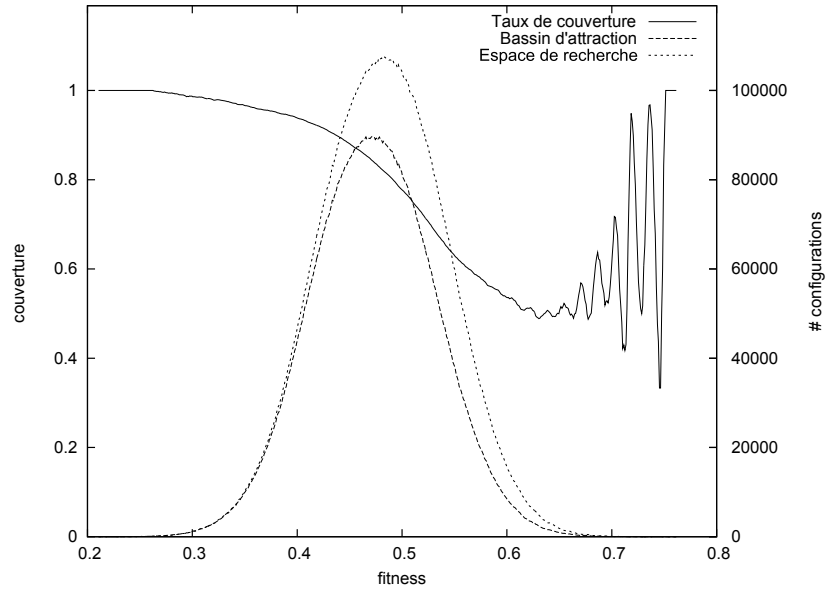


FIGURE 2.6 – Taux de couverture de l'optimum global d'un paysage NK en fonction du fitness des configurations ($N = 24$, $K = 2$, instance #9, taux de couverture global : $80{,}10\%$). Les deux autres courbes indiquent un nombre de configurations, de l'espace de recherche ou de la couverture de l'optimum global, pour chaque intervalle de fitness (de taille 10^{-3}).

Probabilité d'atteindre un optimum global/local

Afin de calculer la probabilité d'atteindre un optimum global à partir d'une configuration aléatoire, nous nous plaçons dans un contexte où tous les voisins améliorants d'une configuration donnée peuvent être choisis de manière équiprobable. Cela correspond à un climber adaptant la stratégie du *premier améliorant*. La méthodologie du calcul de cette probabilité est présentée brièvement ci-dessous.

Méthodologie Pour calculer la probabilité qu'une configuration x_0 atteigne un optimum x_{opt}, il est nécessaire de déterminer la probabilité de sélection de tout chemin améliorant conduisant à x_{opt}. Précisément, si une configuration x a une probabilité p d'être atteinte et dispose de α voisins améliorants, alors la probabilité de choisir chacun des voisins améliorants est de p/α. Lorsqu'une configuration x peut être atteinte à partir de deux configurations qui ont elles-mêmes des probabilités d'être atteintes respectivement de p_1 et p_2, alors la probabilité d'atteindre x est de $p_1 + p_2$. Afin de pouvoir réaliser ces calculs de manière exhaustive, le principe est le même que pour le calcul de la couverture. De plus, étant donné qu'il est fastidieux de calculer les probabilités en empruntant les chemins descendants, on emprunte ici des chemins ascendants, à partir d'un échantillonnage de l'espace des configurations. Plus de détails concernant cet algorithme se trouvent dans [Vigneron *et al.*, 2014].

Résultats La table 2.8 donne les probabilités d'atteindre les meilleurs optima locaux sur les paysages NK, pour $N = 16$. Comme indiqué précédemment, les résultats de la table montrent

16_2	Optimum	10.00 %	30.00 %	50.00 %	Top 5	#LO	Cov.
Moyenne	**14.84%**	**21.84%**	**53.99%**	**75.91%**	**65.70%**	**35.7**	**75.69**
Écart type	13.93 %	11.41 %	12.35 %	8.96 %	22.55 %	21.93	14.86
16_4	Optimum	10.00 %	30.00 %	50.00 %	Top 5	#LO	Cov.
Moyenne	**4.57%**	**32.23%**	**65.30%**	**82.72%**	**18.40%**	**120.5**	**88.85**
Écart type	1.84 %	4.98 %	4.79 %	2.17 %	4.83 %	18.03	2.19
16_8	Optimum	10.00 %	30.00 %	50.00 %	Top 5	#LO	Cov.
Moyenne	**0.94%**	**29.57%**	**59.74%**	**78.07%**	**3.24%**	**755.4**	**70.17**
Écart type	0.74 %	2.45 %	1.63 %	1.02 %	1.19 %	21.42	2.82

TABLE 2.8 – Probabilité d'atteindre différents optima locaux sur les paysages ($N = 16$, $K = \{2,4,8\}$). Les probabilités sont calculées à partir d'un échantillon de 100 configurations aléatoires initiales, sur lequel on a calculé la probabilité d'atteindre chacun des optima locaux. La colonne $Best$ indique la probabilité d'atteindre l'optimum global. Les colonnes 10%, 30% et 50% indiquent les probabilités d'atteindre un optimum local parmi respectivement les 10%, 30% et 50% meilleurs optima locaux des paysages. La colonne Top 5 correspond à la probabilité d'atteindre l'un des 5 meilleurs optima locaux. La colonne #LO indique le nombre d'optima locaux moyen des paysages. La dernière colonne rappelle le taux de couverture moyen de ces paysages.

que malgré un taux de couverture élevé, la probabilité d'atteindre l'optimum global est faible, en particulier lorsque $K = 4$ ou $K = 8$ (voir la deuxième colonne de la table). D'une manière générale, la probabilité d'atteindre l'optimum global décroît lorsque l'on accroît la rugosité des paysages. On observe également la corrélation entre l'indicateur de 1-rugosité que nous avons défini et le nombre d'optima locaux des paysages, qui sert en général d'indicateur de rugosité dans de nombreux travaux. En définitive, ce tableau nous permet de bien visualiser la forte corrélation entre le nombre d'optima locaux, la 1-rugosité et la probabilité d'atteindre l'optimum global.

La table nous apporte une autre information intéressante lorsque l'on observe les autres colonnes. En effet, on remarque que globalement, les meilleurs optima locaux ont une probabilité plus forte d'être atteints que les optima locaux de moins bonne qualité. Par exemple, la probabilité d'atteindre un optimum local parmi les 50% meilleurs est supérieure à 75%. Cela indique qu'un climber choisissant aléatoirement des voisins améliorants tend à converger naturellement vers les meilleurs optima locaux. Dans [Ochoa *et al.*, 2010], un résultat similaire a été obtenu, mais en considérant un climber choisissant à chaque pas le meilleur mouvement possible. D'une manière générale, les meilleurs optima locaux ont un bassin d'attraction plus étendu que les pires optima locaux [Ochoa *et al.*, 2010], ce qui permet de comprendre intuitivement les résultats que nous avons obtenu.

Malgré le fait que les bons optima locaux sont plus enclins à être atteints que les autres optima locaux, la probabilité d'atteindre le meilleur, voire les meilleurs optima locaux tend à être assez faible. Cela devrait être encore plus flagrant si l'on considère des instances plus grandes et plus rugueuses. Pourtant, la couverture de l'optimum global semble rester élevée. Cela nous pousse à penser qu'il doit être possible de mettre au point des climbers ayant des règles de transitions influant sur les probabilités de sélection des mouvements de telle sorte que l'on atteigne plus fréquemment l'optimum global. Cet aspect fait l'objet de nos recherches récentes et sera présenté dans les chapitres suivants.

Corrélation entre taux de couverture et distance de voisinage

Nous avons cherché à calculer le taux de couverture en fonction de l'éloignement des solutions par rapport à la solution optimale dans le graphe de transition des paysages NK. Pour calculer cela, nous avons repris l'algorithme du calcul du taux de couverture en le modifiant de telle sorte qu'un taux soit calculé pour chaque unité de distance entre les configurations du paysage et la configuration de fitness optimal. L'idée est d'évaluer si le taux de couverture baisse sensiblement lorsque l'on considère des solutions éloignées dans le paysage de recherche.

Dans le cadre des paysages NK, on considère donc la distance de Hamming entre les configurations, celle-ci pouvant varier entre 1 et N. Le nombre de configurations se situant à une distance d d'une autre configuration est de $\binom{N}{d}$. Le taux de couverture d'une solution optimale x_{opt} à une distance d vaut donc :

$$\frac{\#\{x \in B_{x_{\text{opt}}}, h(x, x_{\text{opt}}) = d\}}{\binom{N}{d}}$$

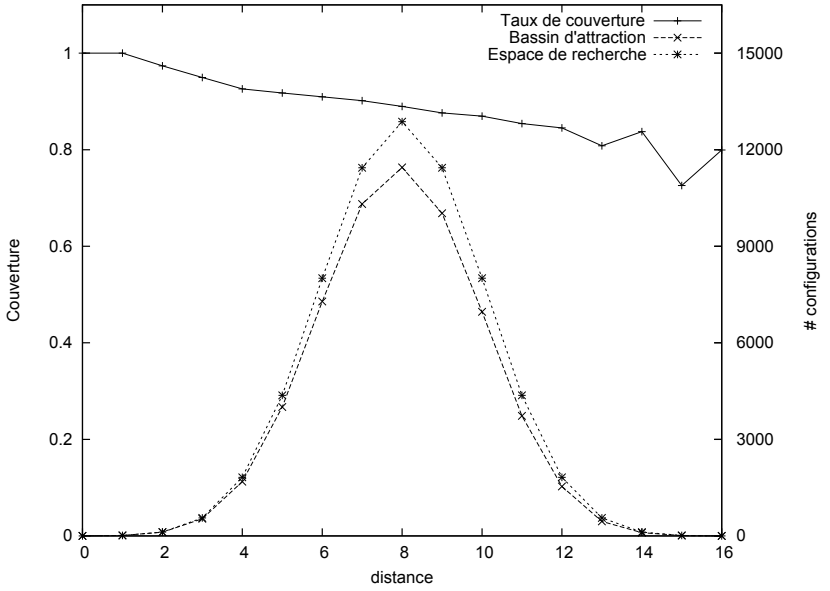


FIGURE 2.7 – Taux de couverture moyens des optima globaux de 10 paysages NK ($N = 16$, $K = 4$, taux de couverture global : $88{,}85\%$). Pour chaque valeur de distance, le nombre de solutions se trouvant dans le bassin d'attraction de l'optimum global apparaît en ligne discontinue, tandis que le nombre total de configurations du paysage se trouvant à une distance d de la configuration optimale apparaît en pointillés.

La figure 2.7 représente une vue du taux de couverture de l'optimum global en fonction de la distance des configurations du paysage de recherche, moyennée sur 10 instances ($N = 16$, $K = 4$). On peut observer que bien que la couverture basée sur les distances tend à décroître de manière linéaire en fonction de la distance à l'optimum global, celle-ci reste relativement élevée, même pour les configurations les plus éloignées du paysage. Cela permet de visualiser l'idée que les climbers n'ont en aucun cas ou presque le défaut de rechercher trop localement, c'est à dire dans une zone très restreinte de l'espace de recherche. Ceci remet en partie en cause l'importance

de la solution initiale pour atteindre de bonnes solutions en fin de recherche.

Corrélation distance de voisinage et longueur de la recherche

Nous avons également étudié s'il existait des chemins améliorants courts permettant de relier les configurations de l'espace de recherche à l'optimum global. Pour cela, nous nous sommes intéressés au calcul du chemin améliorant de longueur minimale reliant les solutions de la couverture de l'optimum global à celui-ci. En d'autres termes, existe-t-il un n-uplet de transitions de telle sorte que chaque bit ne soit modifié qu'une seule fois au cours de la recherche, ou bien est-il absolument nécessaire de modifier plusieurs fois un ou plusieurs bits ? Encore une fois, la méthode de calcul est similaire à celle du calcul du taux de couverture. Néanmoins la quantité de calculs réalisés est limitée dès lors que l'on limite le nombre de pas de la recherche : si pendant la génération d'un chemin, la distance courante à la solution optimale augmentée du nombre de mouvements déjà réalisés dépasse le nombre de mouvements autorisés, on peut stopper la génération de ce chemin.

N	K	Distance	d	$d+2$	$d+4$	$d+6$	$d+8$	Corrélation
16	2	Moyenne	**91.73%**	**6.27%**	**1.39%**	**0.40%**	**0.19%**	**93.38%**
		Écart type	0,042	0,029	0,008	0,004	0,002	0,041
	4	Moyenne	**89,96%**	**7,40%**	**1,81%**	**0,57%**	**0,18%**	**90,87%**
		Écart type	0,029	0,016	0,008	0,004	0,001	0,037
	8	Moyenne	**81,99%**	**12,76%**	**3,63%**	**1,15%**	**0,35%**	**85,34%**
		Écart type	0,023	0,011	0,007	0,004	0,002	0,027
24	2	Moyenne	**88,16%**	**7,63%**	**2,38%**	**0,99%**	**0,46%**	**90,43%**
		Écart type	0,059	0,033	0,015	0,007	0,004	0,050
	4	Moyenne	**91,53%**	**6,26%**	**1,41%**	**0,49%**	**0,19%**	**93,58%**
		Écart type	0,015	0,010	0,003	0,001	0,001	0,000
	8	Moyenne	**87,73%**	**8,21%**	**2,46%**	**0,95%**	**0,40%**	**89,53%**
		Écart type	0,013	0,007	0,003	0,002	0,001	0,001
Moyenne cumulative			**88,52%**	**96,61%**	**98,79%**	**99,55%**	**99,85%**	

TABLE 2.9 – Chemin améliorant de longueur minimale, en fonction de la distance de Hamming initiale d par rapport à la solution optimale. On ne considère ici que les solutions du bassin d'attraction de l'optimum global. Les taux correspondent à des moyennes sur 10 paysages, pour chaque paramétrage (N,K). La dernière colonne correspond à la corrélation entre la longueur des chemins minimaux et la distance de Hamming initiale.

Les résultats pour les instances de tailles $N = 16$ et $N = 24$ apparaissent dans la table 2.9. Les résultats indiquent que pour une grande partie des configurations des paysages de recherche, il existe non seulement un chemin menant vers l'optimum global, mais en plus il existe souvent un chemin direct, c'est à dire que chaque bit n'est modifié qu'une seule fois au maximum durant la recherche. Ces observations indiquent qu'il paraît possible de limiter le nombre de pas des recherches locales sans pour autant anéantir les possibilités d'atteindre l'optimum global. Il pourrait être possible, par exemple, de limiter la recherche de voisinage aux bits qui n'ont pas encore été modifiés. Une telle approche est potentiellement possible, en supposant que la règle pivot utilisée soit également performante malgré cette limitation de voisinage.

Discussion

Nous nous sommes intéressés à l'atteignabilité de l'optimum global dans les paysages NK. Pour cela, nous avons réalisé plusieurs séries d'expérimentations qui nous ont permis d'obtenir quelques conclusions intéressantes. Premièrement, nous avons vu que l'optimum global était généralement atteignable par un simple climber à partir d'une grande proportion des configurations des paysages testés. Ensuite, nous avons vu que même si la probabilité d'atteindre l'optimum global par une stratégie du premier améliorant est faible, cette stratégie permet d'atteindre statistiquement plus souvent les meilleurs optima locaux. Enfin, nous nous sommes intéressés à la relation entre atteignabilité et distance entre configurations. Sur les paysages testés, nous avons remarqué que même lorsque les configurations initiales étaient éloignées de l'optimum global, celui-ci restait très souvent atteignable par un climber. De plus, les expérimentations ont montré que lorsque l'optimum global est atteignable à partir d'une configuration, il existait très souvent un chemin améliorant de longueur minimale, égale à la distance entre les deux configurations. Notons que ces résultats ont été obtenus sur des paysages de recherche sans neutralité, or la neutralité augmente l'atteignabilité de l'optimum global dès lors que les mouvements neutres sont considérés comme acceptables par le climber.

Parmi les extensions possibles de ce travail, il serait intéressant de pouvoir réaliser des études similaires pour de plus grands paysages de recherche : il semble que les résultats ne soient pas nécessairement dépendants de la taille du paysage de recherche, mais cela reste bien entendu à prouver au travers d'une étude expérimentale. Pour cela, il faudrait proposer un protocole qui permettrait une étude approchée du calcul du taux de couverture de paysages de recherches. De la même manière, il faudrait bien entendu réaliser des études similaires sur des paysages issus de problèmes combinatoires classiques afin de généraliser éventuellement les résultats de cette section.

D'une manière générale, cette étude nous amène à penser qu'il est possible de mettre au point des climbers performants par l'élaboration de stratégies avancées de sélection des mouvements, dans le but de maximiser les chances d'atteindre de bons optima locaux. Idéalement, le climber pourrait atteindre l'optimum global à chaque fois que la configuration initiale se trouve dans sa couverture. De manière plus réaliste, il s'agit de contraindre le graphe de transition de telle sorte que les transitions menant vers les bons optima locaux soient conservées et les autres supprimées, ou tout au moins augmenter la probabilité d'emprunter les bonnes transitions.

2.3 Comparaison des algorithmes d'approximation stochastiques

Dans le reste du document, nous nous focalisons sur les algorithmes d'approximation de problèmes d'optimisation combinatoire, et plus particulièrement sur les métaheuristiques. Les métaheuristiques étant généralement stochastiques, il convient de définir comment évaluer leur capactié à atteindre de bonnes configurations. Il convient d'être le plus équitable possible entre les différentes métaheuristiques mises à l'épreuve. De plus, étant donné que nos travaux se concentrent avant tout sur les aspects génériques des métaheuristiques, les méthodes doivent être testées de préférence sur un ensemble varié d'instances. Dans l'ensemble des expérimentations évoquées dans ce document, nous avons essayé de garder une même ligne directrice pour le protocole d'évaluation des métaheuristiques. Voici, dans les grandes lignes, les choix réalisés afin de permettre une évaluation correcte des méthodes d'optimisation :

- **Instances test** – Dans la mesure du possible, les méthodes ont été testées sur des problèmes d'optimisation de natures différentes. Ainsi, ces dernières années, j'ai pu travailler sur les problèmes suivants : Flow-shop (1 à 4 objectifs), assignement quadratique (1 à 4 objectifs), MAXSAT (mono-objectif), paysages NK (1 et 2 objectifs), *ring-star* (bi-objectif), emploi du temps d'infirmières (3 objectifs), sac à dos (2 à 6 objectifs), programmation quadratique binaire (bi-objectif). Dans mes travaux les plus récents, les méthodes tendent à être éprouvées sur plusieurs problèmes. Enfin, pour chaque problème donné, un large panel d'instances test est étudié, toujours dans le but d'éprouver l'adaptabilité des méthodes proposées.
- **Impact de l'aspect stochastique** – Dans un soucis de réduire l'impact de l'aspect stochastique de la plupart des métaheuristiques, de nombreux tests sont effectués, et une étude statistique est réalisée sur les résultats. De plus, toujours dans un soucis d'équité entre méthodes, pour chaque test effectué, les solutions initiales aléatoires sont les mêmes pour chaque méthode, ainsi que la graine aléatoire utilisée pour démarrer le processus d'optimisation.
- **Critère d'arrêt** – Globalement, les tests sont effectués de manière à laisser converger les algorithmes au maximum. L'objectif est que les améliorations probables en augmentant de manière conséquente le temps de calcul soient minimes. Le critère d'arrêt des métaheuristiques est généralement soit basé sur le temps d'exécution, soit sur le nombre d'appels à la fonction d'évaluation. Le temps de calcul à été souvent utilisé, pour prendre en compte la complexité des différents algorithmes proposés, qui est assez variable dans mes travaux. Néanmoins, baser le critère d'arrêt sur un nombre fixé d'évaluation n'est pas dénué de sens, car lorsqu'un problème réel est considéré, la fonction d'évaluation est souvent très coûteuse, et le coût en temps de la méthode d'optimisation peut devenir négligeable comparativement. Dans ce cas, la métaheuristique la plus performante est alors celle capable d'atteindre les meilleures solutions en un nombre minimum de solutions évaluées. Lorsque les méthodes d'optimisation comparées sont de complexité similaire, et que celle-ci est suffisamment faible par rapport au coût de la fonction d'évaluation, un nombre fixé d'évaluations a été choisi comme critère d'arrêt.

Pour évaluer la performance des algorithmes présentés dans ce document, nous avons d'abord cherché à les appliquer sur plusieurs problèmes d'optimisation combinatoire, en étudiant un large panel d'instances pour chaque problème. Cela permet de visualiser l'adaptabilité et la généricité de nos méthodes de manière plus certaine qu'en étudiant un seul problème. Pour une instance de problème donnée, afin de tenir compte de l'aspect stochastique des méthodes d'optimisation, il convient de les comparer sur un ensemble d'évaluations. Pour comparer deux algorithmes $\mathcal{A}$ et $\mathcal{B}$, nous avons suivi le processus suivant :

- Exécutions des algorithmes : Plusieurs exécutions de chaque algorithme sont effectuées sur chaque instance de problème. Il s'agit souvent de plusieurs dizaines d'exécutions, parfois plus, le but étant d'en faire suffisamment pour donner un sens statistique aux résultats. Pour chaque exécution, le fait que chaque algorithme puisse s'initier avec les mêmes solutions aléatoires permet de réduire l'impact du choix de la solution initiale dans les résultats.
- Évaluation statistique : le résultat des exécutions est dans un premier temps décrit par la moyenne des indicateurs obtenus. Cela permet d'avoir de manière très simple une idée globale assez précise de l'efficacité des algorithmes. Nous appliquons en plus une procédure de test statistique afin de pouvoir conclure que *"l'algorithme $\mathcal{A}$ est meilleur que l'algorithme $\mathcal{B}$"*, *"l'algorithme $\mathcal{B}$ est meilleur que l'algorithme $\mathcal{A}$"*, ou *"il n'y a pas de différence significative entre l'algorithme $\mathcal{A}$ et l'algorithme $\mathcal{B}$"*, avec une faible marge d'erreur. Nous

avons travaillé sur des échantillons appariés (mêmes graines aléatoires, mêmes solutions initiales), et avons donc pu utiliser des tests de rang (test de Wilcoxon ou de Mann-Whitley) [Conover, 1999].

Dans les analyses expérimentales contenues dans ce document, on notera $\mathcal{A} \succ_{95\%} \mathcal{B}$ lorsque le test statistique appliqué montre que l'algorithme $\mathcal{A}$ est meilleur que l'algorithme $\mathcal{B}$ avec une confiance supérieure à 95% (de même pour $\mathcal{A} \succ_{90\%} \mathcal{B}$ pour une confiance supérieure à 90%). On notera $\mathcal{A} \succ_{avg} \mathcal{B}$ lorsque les résultats moyens de $\mathcal{A}$ sont meilleurs que ceux de $\mathcal{B}$, sans forcément qu'il y ait une dominance statistique avérée.

Chapitre 3

Comparaison des climbers sur des paysages de recherche

Il est utopique de vouloir analyser de manière globale le comportement des algorithmes de recherche de la littérature, même si l'on se restreint aux métaheuristiques pour l'optimisation combinatoire mono-objectif. Dans ce chapitre, nous nous concentrons sur l'analyse du comportement des climbers en fonction des caractéristiques des paysages de recherche. Pour cela, nous avons réalisé une étude empirique sur différents problèmes d'optimisation et structures de voisinage afin d'étudier le comportement de plusieurs variantes classiques de climbers. En particulier, nous avons évalué l'impact du choix de la *règle pivot*, en comparant principalement les stratégies *premier améliorant* et *meilleur améliorant*. Puis, nous comparons différentes politiques de prise en compte des *mouvements neutres* qui indiquent quand un mouvement vers une configuration de fitness équivalent peut être accepté durant la recherche.

3.1 Composants des climbers

La conception d'un climber implique un certain nombre de choix déterminants. L'impact de ces choix est globalement loin d'être évident, pourtant ces aspects sont peu discutés dans les articles de recherche. Nous distinguons ici quatre choix de conceptions susceptibles d'influer sur le comportement des climbers :

Règle pivot

Cela consiste à définir quel voisin améliorant est choisi à chaque étape de la recherche. En général, la définition d'une règle pivot se limite à choisir entre deux méthodes. La stratégie du *meilleur améliorant* consiste à sélectionner, à chaque itération, un voisin qui réalise la meilleure amélioration possible. Cela implique de générer l'ensemble du voisinage à chaque étape de la recherche, sauf si une évaluation incrémentale des voisins peut être effectuée. La stratégie du *premier améliorant*, quant à elle, accepte le premier voisin améliorant qui satisfait la condition de déplacement. Cela permet d'éviter de produire systématiquement l'ensemble du voisinage et offre plus d'options conceptuelles.

Politique de prise en compte de la neutralité

Un climber *basique* ne permet pas les déplacements neutres (*ie* se déplacer vers un voisin de même fitness) lors de la recherche, et effectue seulement des mouvements strictement améliorants jusqu'à atteindre un optimum local. L'autorisation des mouvements neutres durant la recherche peut être considérée pour échapper aux optima locaux (*perturbation neutre*, NP) dès lors que le paysage de fitness contient une proportion significative de transitions neutres (sur les paysages lisses). Dans ce cas, on autorise un déplacement neutre lorsqu'aucun voisin améliorant n'a pu être trouvé. L'utilisation de cette méthode est largement répandue. Une autre variante, appelée *climber stochastique*, accepte indifféremment les voisins neutres ou améliorants tout au long de la recherche, avant même d'atteindre un optimum local. Cela conduit en général à réaliser des mouvements neutres alors qu'il existait des solutions améliorantes dans le voisinage. Il est intéressant de noter que l'algorithme du recuit simulé, qui permet la détérioration des solutions durant la recherche, accepte également systématiquement les voisins neutres.

Évaluation du voisinage

Affirmer avec certitude qu'un climber a atteint un optimum local nécessite l'utilisation d'une politique de prise en compte des mouvements neutres favorisant les mouvements améliorants (basique ou NP). En effet, le climber stochastique accepte les déplacements neutres avant même de savoir si le voisinage contenait des voisins améliorants. Une deuxième condition nécessaire pour prouver l'atteinte d'un optimum local est d'être capable de détecter lorsque le voisinage complet a été généré. Cela n'est possible que si l'on considère une évaluation *exhaustive* du voisinage (évaluation sans remise). Au contraire, si le voisinage est évalué de manière *aléatoire* (avec remise), alors on ne peut pas savoir facilement si l'ensemble du voisinage a été généré et donc si un optimum local est atteint. Globalement, l'évaluation *exhaustive* sera préférée, mais lorsque certains aspects du problème rendent difficile ou impossible une évaluation exhaustive, alors on réalise une évaluation *aléatoire* (dans le cas d'une représentation complexe des solutions, ou de l'utilisation de voisinages spécifiques ou extrêmement larges).

Exploration du voisinage

D'après les définitions proposées dans [Hoos et Stützle, 2004], la stratégie du premier améliorant explore le voisinage dans un ordre *déterministe* invariant durant la totalité de la recherche, le premier voisin satisfaisant la règle pivot étant accepté. Une alternative serait de définir un ordre de génération *aléatoire* à chaque étape de la recherche. Cela permet entre autres de réduire le risque de cycler lorsqu'un climber stochastique est utilisé en combinaison avec une évaluation déterministe du voisinage. Globalement, une génération aléatoire peut sembler préférable, même si certaines propriétés du problème considéré peuvent rendre une exploration déterministe du voisinage intéressante.

Les différentes manières de concevoir un climber que nous venons de présenter sont résumées dans la figure 3.1. Naturellement, la structure des solutions, du voisinage et de la fonction d'évaluation peut mener à favoriser certains choix. Cependant, dans la plupart des cas, en particulier lorsque ces structures sont classiques, toutes les options peuvent être considérées. Concernant les aspects *techniques*, en l'absence de propriétés spécifiques au problème considéré, on se portera

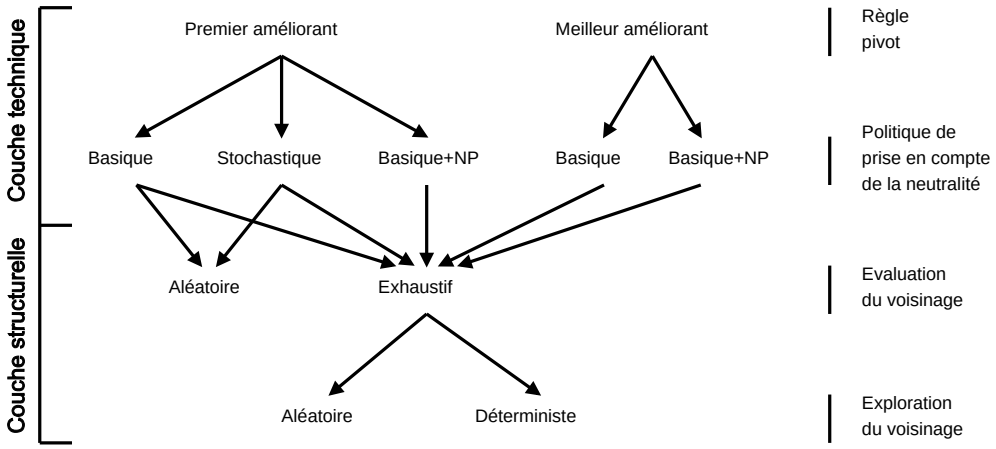


FIGURE 3.1 – Choix de conception d'un climber.

prioritairement sur une évaluation *exhaustive* du voisinage, celui-ci étant généré de manière *aléatoire*. Concernant les aspects *structurels* (règle pivot et prise en compte de la neutralité), les choix sont plus difficiles et déterminants. Nous allons donc par la suite évaluer l'influence de ces choix sur l'efficacité des climbers, en fonction de l'allure du paysage de recherche considéré.

Les choix structurels sont la règle pivot et la politique de prise en compte de la neutralité. Nous avons détaillé deux règles pivots classiques (premier améliorant et meilleur améliorant) et trois politiques de prise en compte de la neutralité (basique, basique avec perturbations neutres et stochastique). En combinant ces trois politiques avec les deux règles pivot de la section précédente, on obtient cinq variantes de climbers : deux de ces variantes, ne tiennent pas compte de la possibilité d'accepter des mouvements neutres, donc seule la règle pivot définit ces climbers. Nous les appellerons par la suite `FIRST` (premier améliorant) et `BEST` (meilleur améliorant). Les deux règles pivots se combinent également avec la politique d'accepter les mouvements neutres lorsqu'il n'y a plus de voisin améliorant. Nous appellerons ces climbers `FIRST+NP` (premier améliorant et perturbations neutres) et `BEST+NP` (meilleur améliorant et perturbations neutres). Enfin, la politique de prise en compte de la neutralité consistant à accepter indifféremment les mouvements neutres et améliorants est incompatible avec la règle pivot meilleur améliorant. Nous appellerons cette dernière stratégie `STOCH`.

Nous obtenons donc cinq climbers classiques, qui induisent des recherches de voisinages aux comportements différents comme le montre l'exemple de la figure 3.2. Cette figure représente un paysage comportant 9 configurations et montre que les graphes de transitions de recherche itérée sont différents et que la probabilité d'atteindre chacun des 3 optima locaux varient selon la stratégie utilisée. L'objectif de l'étude empirique réalisée dans ce chapitre est d'évaluer quelle stratégie permet au mieux d'atteindre les bons optima locaux avec une forte probabilité. Remarquons qu'il est assez facile de calculer les probabilités d'atteindre chaque optimum local, mais uniquement sur des petit graphes de transition de recherche itérée.

Dans la suite, nous allons séparer l'évaluation des deux règles pivots basiques (`FIRST` et `BEST`) de l'évaluation des trois autres stratégies prenant en compte la possibilité d'effectuer des mouvements neutres (`FIRST+NP`, `BEST+NP` et `STOCH`).

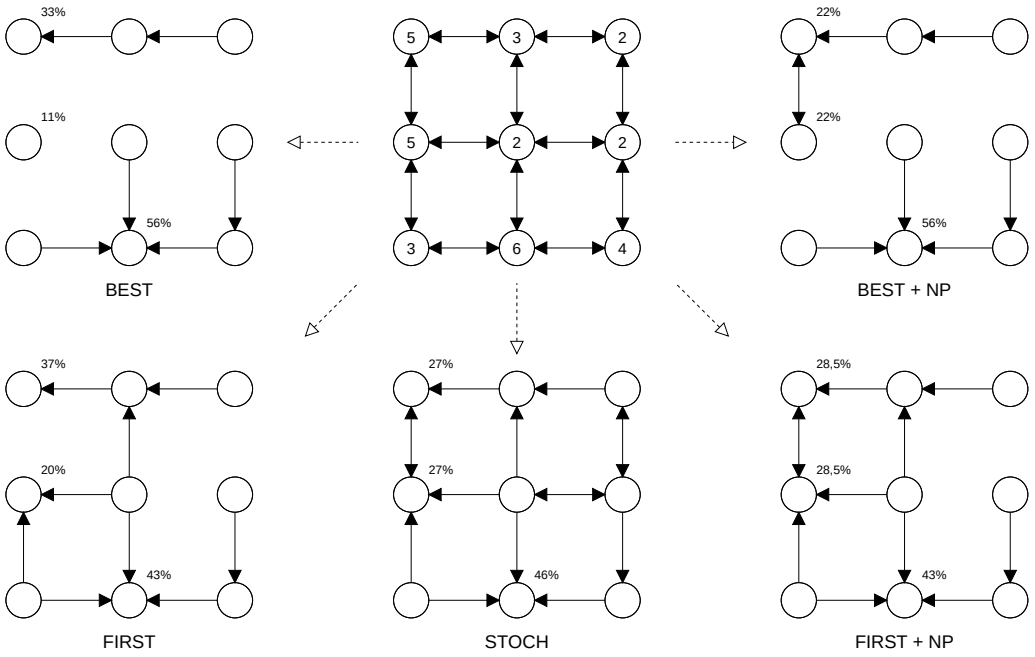


FIGURE 3.2 – Impact des choix de conception structurelle des climbers sur le graphe de transition de recherche itérée résultant et par conséquent sur les probabilités de terminer la recherche sur chaque optimum local.

3.2 Évaluation des règles pivot

Dans cette section, nous nous concentrons sur l'évaluation de l'efficacité des deux règles pivot classiques de la littérature : premier améliorant (`FIRST`) et meilleur améliorant (`BEST`).

Algorithme 3.1 Climber `FIRST`

Entrée : une configuration initiale x
tant que x n'est pas un optimum local **faire**
  sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}(x') > \mathfrak{f}(x)$
  $x \leftarrow x'$
Retourner x

Algorithme 3.2 Climber `BEST`

Entrée : une configuration initiale x
tant que x n'est pas un optimum local **faire**
  $x \leftarrow argmax_{x' \in \mathcal{N}(x)}(\mathfrak{f}(x'))$
Retourner x

Une description générale de `FIRST` et `BEST` est donnée respectivement dans les algorithmes 3.1 et 3.2. Ces algorithmes sont décrits de manière à être les plus simples possibles, mais des précisions doivent être données pour détailler l'aspect implémentation de ces algorithmes.

L'ordre de génération des voisins d'une solution est entièrement aléatoire et sans remise, à chaque pas de la recherche. À chaque pas de la recherche, `FIRST` stoppe la génération du voi-

sinage dès lors qu'une solution améliorante est trouvée. Par contre BEST nécessite de générer la totalité du voisinage à chaque pas de la recherche. De même, la vérification du critère d'arrêt se fait durant chaque étape de recherche et ne nécessite bien sûr aucune évaluation de configuration supplémentaire.

3.2.1 Protocole expérimental

Nous avons réalisé une série de tests sur 288 des 480 paysages étudiés lors de l'analyse de paysage réalisée dans le chapitre 2, soit 16 paysages NK, 27 paysages $\mathrm{FSP}_{\mathrm{ins}}$, 132 paysages $\mathrm{QAP}_{\mathrm{swap}}$ et 129 paysages MAXSAT. Nous n'avons pas étudié ici les paysages NK avec neutralité.

Nous avons réalisé 100 exécutions de chaque climber par paysage. Comme FIRST et BEST n'autorisent pas les mouvements neutres, ils s'arrêtent naturellement lorsqu'un optimum local (non nécessairement strict) est atteint. Puis, nous avons calculé les fitness moyen atteints par chaque climber sur chaque paysage. Enfin, nous avons réalisé une étude statistique sur chaque instance, en appliquant le test de Mann-Whitney.

Les tables 3.1, 3.2, 3.3 et 3.4 des sections suivantes répertorient les résultats obtenus par les climbers FIRST et BEST sur différents types de paysages de recherche : NK (table 3.1), $\mathrm{QAP}_{\mathrm{swap}}$ (table 3.2), $\mathrm{FSP}_{\mathrm{ins}}$ (table 3.3) et MAXSAT (table 3.4). Dans ces tables, le fitness moyen le plus élevé sur chaque paysage apparait en gras. De plus, une case grisée correspond à un climber n'étant pas dominé statistiquement par le meilleur climber sur un paysage donné (p-value de 95%). Une case apparait en gris clair lorsque la p-value n'est que de 90%.

3.2.2 Étude sur les paysages NK

N_K	$\rho_1^>$	FIRST	BEST
128_1	0,52%	0,7021	**0,7090**
128_2	1,24%	0,7021	**0,7082**
128_4	3,12%	0,7254	**0,7260**
128_8	7,40%	**.7142**	0,7108
256_1	0,26%	0,7021	**0,7079**
256_2	0,61%	0,7066	**0,7094**
256_4	1,57%	**0,7235**	0,7204
256_8	3,98%	**0,7166**	0,7122

N_K	$\rho_1^>$	FIRST	BEST
512_1	0,12%	0,6897	**0,6953**
512_2	0,32%	0,7135	**0,7174**
512_4	0,80%	**0,7200**	0,7193
512_8	2,06%	**0,7206**	0,7155
1024_1	0,07%	0,6969	**0,7039**
1024_2	0,14%	0,7146	**0,7197**
1024_4	0,39%	**0,7246**	0,7242
1024_8	1,05%	**0,7216**	0,7174

TABLE 3.1 – Résultats des climbers basiques sur les paysages NK : comparaison des règles pivot *premier améliorant* et *meilleur améliorant*. $\rho_1^>$ représente la rugosité locale évaluée pour le paysage correspondant.

Les résultats sur les paysages NK, pour lesquels la rugosité et la dimension sont ajustables en fonction de N et K, nous fournissent plusieurs éléments d'information. La stratégie du meilleur améliorant (BEST) surpasse statistiquement la stratégie du premier améliorant (FIRST) pour $K \in \{1,2\}$, c'est-à-dire pour les paysages peu rugueux. Par contre, FIRST semble plus efficace lorsque K, et par conséquent la rugosité, augmente. La taille de l'espace de recherche (2^N) ne semble pas affecter à première vue l'efficacité relative des deux règles pivots. Cependant, on peut voir que le taux de rugosité des paysages n'est pas seulement déterminé par K, car il diminue lorsque N croît. Par conséquent, l'efficacité relative des climbers est non seulement affectée par la rugosité

du paysage, mais aussi par sa dimension. La figure 3.3 (a) met l'accent sur la relation entre la robustesse, la dimension du paysage et l'efficacité relative entre FIRST et BEST. Cela nous permet d'observer que plus un paysage NK est difficile (grand et rugueux), plus il est préférable d'utiliser la stratégie du premier améliorant.

3.2.3 Étude sur le problème d'assignement quadratique

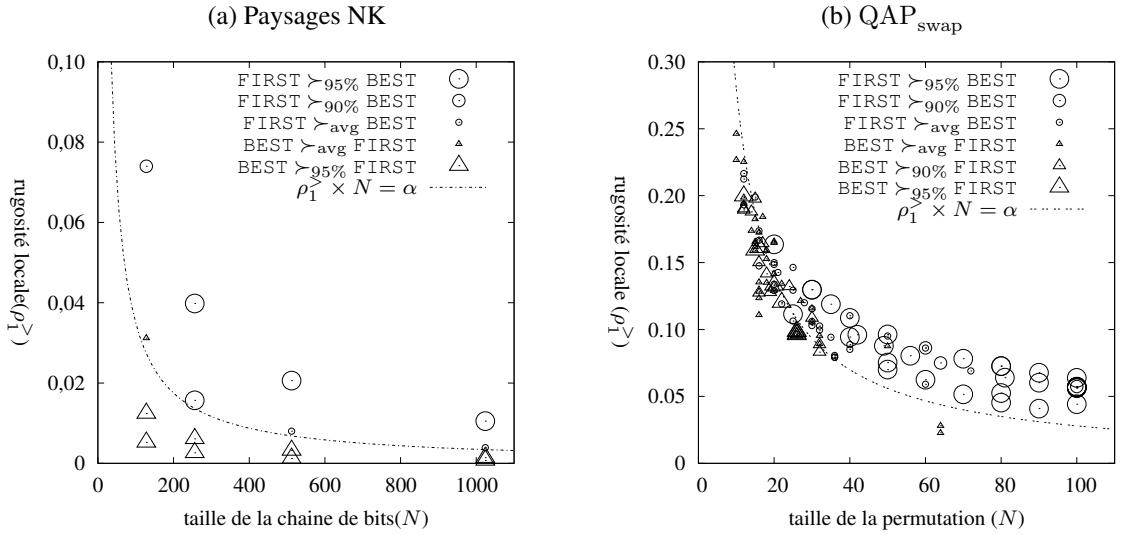


FIGURE 3.3 – Supériorité de FIRST (cercles) ou BEST (triangles) sur les différents paysages NK et QAP$_{\text{swap}}$, en fonction de leur taille et de leur rugosité. La ligne pointillée met l'accent sur la corrélation entre rugosité, taille et l'efficacité des climbers BEST et FIRST (ici, $\alpha = 3{,}5$ pour le NK et $\alpha = 2{,}8$ pour QAP$_{\text{swap}}$).

La table 3.2 résume les résultats obtenus sur les paysages QAP$_{\text{swap}}$. Ces paysages sont dérivés d'instances du problème d'optimisation combinatoire classique qu'est le QAP. Nous avons considéré un large panel d'instances, dont les paysages ont des propriétés diverses. De manière globale, on observe que FIRST $\succ_{95\%}$ BEST sur 32 des 124 paysages considérés, tandis que BEST $\succ_{95\%}$ FIRST sur uniquement 9 paysages. Plus précisément, BEST semble plus approprié pour traiter les instances de Burkard et Offermann (bur*) ainsi que les instances de Christofides et Benavent (chr*). FIRST est particulièrement adapté pour résoudre les instances de Skorin-Kapov (sko*), Steinberg (ste*) et les grandes instances de Taillard (tai*).

La figure 3.3.b met l'accent sur la relation entre robustesse, dimension et efficacité relative des deux climbers. Cela tend à confirmer les observations faites pour les paysages NK, c'est à dire que FIRST tends à être plus performant lorsque la dimension et la rugosité des paysages augmente. Cela nous pousse à penser qu'il existe une relation entre ces propriétés des paysages de recherche et l'efficacité relative de FIRST et BEST. En effet, les tests réalisés sur les paysages NK et QAP$_{\text{swap}}$ montrent qu'un simple produit de la dimension de l'espace de recherche [1] par la rugosité du paysage associé permet de discriminer convenablement l'efficacité relative de FIRST par

1. Ici, ce n'est pas la dimension de l'espace de recherche, mais la taille N de la configuration. Malgré tout, N définit la dimension : $\|\mathcal{X}\| = 2^N$ pour les paysages NK et $\|\mathcal{X}\| = N!$ pour les paysages dérivés du QAP.

Instance	$\rho_1^>$	FIRST	BEST
bur26a	9,65%	**5 444 341,11**	5 444 564,55
bur26b	9,65%	3 835 171,43	**3 832 617,90**
bur26c	9,85%	5 452 501,58	**5 446 368,78**
bur26d	9,82%	3 842 352,83	**3 836 063,35**
bur26e	9,65%	5 407 830,26	**5 407 338,95**
bur26f	9,65%	3 801 482,16	**3 799 065,59**
bur26g	9,59%	10 161 339,46	**10 157 156,05**
bur26h	9,59%	7 133 939,98	**7 126 406,74**
chr12a	19,86%	14 595,14	**14 430,54**
chr12b	19,40%	15 512,32	**14 800,70**
chr12c	19,96%	15 903,82	**15 283,30**
chr15a	16,40%	15 019,92	**14 655,54**
chr15b	15,91%	13 455,44	**12 803,86**
chr15c	16,56%	15 703,68	**15 597,06**
chr18a	14,16%	19 067,42	**18 400,66**
chr18b	13,50%	1 813,46	**1 797,36**
chr20a	13,11%	3 305,02	**3 241,84**
chr20b	13,30%	3 325,40	**3 212,56**
chr20c	12,91%	26 596,74	**26 280,50**
chr22a	11,91%	**7 027,72**	7 039,80
chr22b	12,01%	7 127,76	**7 049,98**
chr25a	10,63%	**6 025,08**	6 026,96
els19	12,77%	22 280 466,04	**21 630 921,32**
esc16a	14,75%	**70,28**	70,68
esc16b	12,35%	292,04	**292,02**
esc16c	16,10%	163,06	**161,44**
esc16d	15,00%	17,78	**17,44**
esc16e	12,83%	30,86	**30,66**
esc16g	12,73%	29,02	**28,46**
esc16h	14,41%	**996,00**	**996,00**
esc16i	13,53%	14,24	**14,14**
esc16j	11,08%	9,26	**9,14**
esc32a	9,53%	160,54	**160,36**
esc32b	10,26%	**216,92**	217,12
esc32c	8,87%	646,58	**645,16**
esc32d	8,31%	214,28	**213,18**
esc32e	3,55%	**2,00**	**2,00**
esc32f	3,53%	**2,00**	**2,00**
esc32g	3,32%	**6,06**	**6,06**
esc32h	8,91%	455,04	**453,98**
esc64a	2,81%	118,42	**117,96**
had12	18,93%	1 676,52	**1 673,72**
had14	17,38%	2 751,38	**2 750,18**
had16	16,67%	**3 753,48**	3 754,78
had18	15,27%	5 425,24	**5 424,16**
had20	14,16%	7 006,54	**7 000,06**
kra30a	10,60%	**95 771,10**	95 854,80
kra30b	10,55%	96 633,40	**96 428,50**
kra32	9,95%	**18 974,76**	19 032,76
lipa20a	15,00%	**3 788,74**	3 789,29
lipa20b	16,47%	31 305,07	**31 137,48**
lipa30a	10,85%	13 447,09	**13 441,44**
lipa30b	12,98%	**176 474,92**	176 739,00
lipa40a	8,52%	**32 014,02**	32 014,85
lipa40b	10,87%	**565 732,69**	566 510,70
lipa50a	7,01%	**62 888,38**	62 907,60
lipa50b	9,51%	**1 434 252,14**	1 436 410,95
lipa60a	5,91%	**108 401,43**	108 403,75
lipa60b	8,61%	**3 019 482,01**	3 025 897,53
lipa70a	5,16%	**171 380,64**	171 400,42
lipa70b	7,82%	**5 543 957,79**	5 559 790,86
lipa80a	4,54%	**255 307,64**	255 334,38
lipa80b	7,26%	**9 435 185,09**	9 444 877,75
lipa90a	4,09%	**363 384,10**	363 429,86
lipa90b	6,77%	**15 203 151,52**	15 216 234,09

Instance	$\rho_1^>$	FIRST	BEST
nug12	21,24%	**610,10**	610,62
nug14	18,79%	1 067,90	**1 062,52**
nug15	18,25%	1 204,80	**1 201,34**
nug16a	17,37%	**1 688,24**	1 692,00
nug16b	17,25%	1 310,26	**1 307,24**
nug17	16,42%	1 809,30	**1 803,50**
nug18	15,93%	2 019,74	**2 018,48**
nug20	14,86%	**2 676,88**	2 678,52
nug21	14,26%	**2 547,40**	2 547,86
nug22	13,42%	3 731,10	**3 722,00**
nug24	13,21%	3 657,92	**3 643,88**
nug25	12,94%	**3 884,56**	3 887,48
nug27	12,14%	5 454,68	**5 453,06**
nug28	12,00%	**5 391,92**	5 398,20
nug30	11,55%	**6 368,52**	6 384,80
rou12	22,52%	248 706,78	**248 356,40**
rou15	19,75%	379 573,68	**378 164,46**
rou20	16,36%	**759 464,96**	764 630,10
scr12	19,32%	**33 702,62**	33 710,66
scr15	16,16%	57 027,94	**56 401,26**
scr20	13,38%	**121 454,24**	121 880,78
sko100a	5,63%	**154 795,60**	155 195,18
sko100b	5,63%	**156 675,30**	156 962,44
sko100c	5,72%	**150 997,60**	151 312,00
sko100d	5,73%	**152 286,52**	152 626,80
sko100e	5,67%	**152 173,38**	152 600,80
sko100f	5,71%	**151 733,36**	152 129,30
sko42	9,61%	**16 321,16**	16 370,32
sko49	8,78%	**24 041,86**	24 160,86
sko56	8,04%	**35 369,66**	35 474,52
sko64	7,50%	**49 707,76**	49 782,98
sko72	6,90%	**67 898,94**	67 965,76
sko81	6,40%	**92 874,70**	93 124,36
sko90	6,03%	**117 860,54**	118 112,88
ste36a	8,06%	**10 603,04**	10 639,14
ste36b	7,91%	**18 992,66**	19 078,18
ste36c	7,94%	**8 945 385,72**	9 021 581,74
tai100a	6,39%	**21 811 727,64**	21 843 265,60
tai100b	4,42%	**1 237 857 238,07**	1 249 060 042,55
tai10a	24,63%	143 711,68	**142 578,46**
tai10b	22,67%	1 339 375,60	**1 319 043,74**
tai12a	21,65%	**244 924,42**	245 395,56
tai12b	19,02%	44 193 333,33	**43 651 098,92**
tai15a	19,89%	**406 509,00**	407 274,54
tai15b	15,91%	52 063 153,51	**52 050 124,01**
tai17a	18,42%	520 972,34	**519 811,32**
tai20a	16,58%	747 454,68	**745 418,74**
tai20b	12,91%	**141 012 824,06**	142 677 265,99
tai25a	14,64%	**1 227 561,55**	1 228 539,72
tai25b	11,15%	**391 785 290,84**	405 482 117,50
tai30a	12,97%	**1 907 004,30**	1 911 719,42
tai30b	10,29%	**717 135 475,20**	726 629 805,20
tai35a	11,89%	**2 536 171,48**	2 550 449,86
tai35b	9,42%	**306 198 132,06**	306 229 161,19
tai40a	11,03%	**3 293 480,92**	3 296 135,10
tai40b	8,87%	**699 305 121,85**	703 984 330,08
tai50a	9,63%	**5 175 285,48**	5 185 014,66
tai50b	7,53%	**487 675 806,50**	491 485 627,51
tai60a	8,63%	**7 535 747,82**	7 544 353,22
tai60b	6,25%	**649 061 353,23**	657 167 128,71
tai64c	2,27%	1 865 318,78	**1 864 589,22**
tai80a	7,27%	**14 032 294,42**	14 075 016,98
tai80b	5,25%	**863 181 601,34**	869 008 411,81
tho30	11,59%	157 233,30	**156 888,48**
tho40	9,44%	**251 350,88**	251 944,34
wil100	5,74%	**275 844,92**	275 987,04
wil50	8,75%	49 557,58	**49 544,90**

TABLE 3.2 – Comparaison empirique des règles pivot sur le QAP : FIRST vs BEST.

rapport à BEST. Globalement, l'utilisation de la stratégie du premier améliorant semble indiquée pour traiter des paysages difficiles, c'est à dire de grande taille et/ou avec une rugosité élevée.

3.2.4 Étude sur le problème de flow-shop

Instance	$\rho_1^>$	FIRST	BEST
20_5_01_ta001	10,10%	**1 306,65**	1 311,03
20_10_01_ta011	11,94%	**1 629,34**	1 634,26
20_15_01	12,31%	**1 961,41**	1 972,55
20_20_01_ta021	13,22%	**2 374,68**	2 391,85
30_5_01	7,43%	**1 736,83**	1 737,45
30_10_01	9,99%	**2 072,23**	2 084,73
30_15_01	10,59%	**2 510,73**	2 516,80
30_20_01	11,47%	**2 856,61**	2 869,51
50_5_01_ta031	4,89%	**2 743,25**	2 749,92
50_10_01_ta041	7,63%	**3 140,22**	3 147,70
50_15_01	8,37%	**3 535,57**	3 546,65
50_20_01_ta051	9,06%	**4 033,70**	4 056,03
70_5_01	4,14%	**3 794,44**	3 799,28
70_10_01	6,22%	**4 168,01**	4 175,18

Instance	$\rho_1^>$	FIRST	BEST
70_15_01	7,22%	**4 566,58**	4 580,14
70_20_01	7,73%	**4 980,50**	5 004,06
100_5_01_ta061	4,00%	**5 509,95**	5 516,47
100_10_01_ta071	5,26%	**5 864,46**	5 894,90
100_15_01	6,01%	**6 155,31**	6 182,83
100_20_01_ta081	6,55%	**6 560,49**	6 614,15
150_5_01	2,54%	**8 066,32**	8 068,63
150_10_01	4,30%	**8 240,64**	8 284,83
150_15_01	4,84%	**8 676,75**	8 712,15
150_20_01	5,47%	**9 327,67**	9 396,75
200_10_01_ta091	3,56%	**11 006,30**	11 049,20
200_15_01	4,27%	**11 400,00**	11 469,20
200_20_01_ta101	4,73%	**11 650,70**	11 711,40

TABLE 3.3 – Comparaison empirique des règles pivot sur le flow-shop : FIRST vs BEST.

Les résultats obtenus sur les paysages $\mathrm{FSP}_{\mathrm{ins}}$ (table 3.3) montrent une claire supériorité de FIRST, qui obtient des résultats meilleurs en moyenne sur l'ensemble des paysages testés. De plus, statistiquement parlant, FIRST $\succ_{95\%}$ BEST sur 23 des 27 paysages. Ce résultat est intéressant puisque dans la littérature, les méthodes proposées semblent utiliser principalement des recherches de voisinage basées sur le principe du premier améliorant. On peut citer le travail de Juan [Juan *et al.*, 2013] pour la recherche locale itérée et [Taillard, 1990, Ben-Daya et Al-Fawzan, 1998] pour des recherches Tabu utilisant la stratégie du premier améliorant (à l'origine la recherche Tabu a été décrite comme étant basée sur la stratégie du meilleur améliorant). Néanmoins, dans ces travaux, l'utilisation ou non de FIRST n'est pas débattue, et semble plus résulter d'aspects pratiques puisque sur ce problème il n'existe pas de méthode efficace permettant de tirer un bénéfice significatif d'une évaluation incrémentale du voisinage.

3.2.5 Étude sur le problème de satisfiabilité

Les résultats obtenus sur les paysages dérivés de MAXSAT nous amènent à des conclusions plus contrastées (voir table 3.4). Ici, n and $\rho_1^>$ ne nous permettent pas de discriminer convenablement FIRST et BEST. Malgré cela, nous pouvons noter une tendance assez claire sur les instances aléatoires, pour lesquelles FIRST n'est jamais dominé statistiquement alors qu'il domine BEST sur 11 instances (table 3.4, partie gauche). Sur les instances MAXSAT construites à partir de problèmes d'optimisation combinatoire concrets, les résultats semblent directement liés aux spécificités de ces problèmes et donc des instances MAXSAT associées (table 3.4, partie gauche, pour différents types d'instances construites à partir de problèmes d'optimisation combinatoire concrets).

Notons que Whitley *et al.* sont arrivés à des conclusions relativement proches, à savoir que la stratégie du premier améliorant est plus efficace pour traiter les instances aléatoires tandis que

Instances 3-SAT aléatoires				
Instance	n	$\rho_1^>$	FIRST	BEST
uf20-01	20	1,81%	2,39	**2,24**
uf50-01	50	0,31%	**5,25**	5,62
uuf50-01	50	0,31%	**6,46**	6,58
uf75-01	75	0,18%	**8,67**	8,90
uuf75-01	75	0,15%	9,24	**8,83**
uf100-01	100	0,09%	**10,77**	11,44
uuf100-01	100	0,08%	**11,34**	11,38
uf125-01	125	0,05%	**12,29**	13,11
uuf125-01	125	0,05%	**14,10**	14,78
uf150-01	150	0,03%	**14,76**	15,27
uuf150-01	150	0,03%	**15,42**	15,73
uf175-01	175	0,03%	18,26	**18,22**
uuf175-01	175	0,03%	**19,25**	19,71
uf200-01	200	0,02%	**20,67**	21,94
uuf200-01	200	0,03%	**23,42**	24,27
uf225-01	225	0,02%	**21,58**	22,83
uuf225-01	225	0,02%	**24,79**	25,46
uf250-01	250	0,01%	**23,66**	24,38
uuf250-01	250	0,01%	**25,67**	26,83
RTI_...m429_0	100	0,08%	**9,51**	9,96
BMS_...m429_0	100	0,03%	**7,89**	8,11
CBS_...m403_b10_0	100	0,07%	**9,50**	9,61
CBS_...m403_b30_0	100	0,08%	9,48	**9,46**
CBS_...m403_b50_0	100	0,07%	**9,59**	9,81
CBS_...m403_b70_0	100	0,07%	**9,98**	10,50
CBS_...m403_b90_0	100	0,09%	**10,66**	11,02
CBS_...m411_b10_0	100	0,08%	**9,42**	9,82
CBS_...m411_b30_0	100	0,07%	**10,01**	10,37
CBS_...m411_b50_0	100	0,05%	11,03	**10,87**
CBS_...m411_b70_0	100	0,06%	**10,69**	10,78
CBS_...m411_b90_0	100	0,06%	10,84	**10,78**
CBS_...m418_b10_0	100	0,08%	**8,58**	9,00
CBS_...m418_b30_0	100	0,09%	**9,83**	9,98
CBS_...m418_b50_0	100	0,08%	**10,51**	10,74
CBS_...m418_b70_0	100	0,11%	10,20	**10,12**
CBS_...m418_b90_0	100	0,08%	**10,34**	10,53
CBS_...m423_b10_0	100	0,09%	**9,99**	10,56
CBS_...m423_b30_0	100	0,09%	**9,76**	10,00
CBS_...m423_b50_0	100	0,09%	9,87	**9,80**
CBS_...m423_b70_0	100	0,09%	**8,96**	9,01
CBS_...m423_b90_0	100	0,10%	10,31	**10,14**
CBS_...m429_b10_0	100	0,08%	9,83	**9,52**
CBS_...m429_b30_0	100	0,08%	**9,28**	9,31
CBS_...m429_b50_0	100	0,11%	**10,55**	11,27
CBS_...m429_b70_0	100	0,09%	**9,54**	9,98
CBS_...m429_b90_0	100	0,10%	**10,73**	10,87
CBS_...m435_b10_0	100	0,09%	**9,21**	9,65
CBS_...m435_b30_0	100	0,09%	**10,35**	11,28
CBS_...m435_b50_0	100	0,08%	**10,07**	10,66
CBS_...m435_b70_0	100	0,09%	**11,01**	11,42
CBS_...m435_b90_0	100	0,09%	11,17	**10,98**
CBS_...m441_b10_0	100	0,07%	**9,05**	9,82
CBS_...m441_b30_0	100	0,09%	**9,86**	10,24
CBS_...m441_b50_0	100	0,09%	9,80	**9,71**
CBS_...m441_b70_0	100	0,08%	**10,59**	11,08
CBS_...m441_b90_0	100	0,07%	**11,53**	12,37
CBS_...m449_b10_0	100	0,08%	**9,24**	10,03
CBS_...m449_b30_0	100	0,10%	11,00	**10,87**
CBS_...m449_b50_0	100	0,06%	**10,30**	10,58
CBS_...m449_b70_0	100	0,09%	**11,48**	**11,48**
CBS_...m449_b90_0	100	0,07%	**12,41**	12,70

Instances basées sur des problèmes				
Instance	n	$\rho_1^>$	FIRST	BEST
flat30-1	90	0,09%	7,08	**5,76**
flat50-1	150	0,06%	12,45	**10,68**
flat75-1	225	0,03%	19,58	**16,88**
flat100-1	300	0,02%	25,81	**21,94**
flat125-1	375	0,02%	32,45	**26,94**
flat150-1	450	0,02%	39,77	**32,46**
flat175-1	525	0,01%	44,08	**37,82**
flat200-1	600	0,01%	51,11	**42,74**
sw100-8-lp0-c5-1	500	0,00%	18,68	**14,00**
sw100-8-lp1-c5-1	500	0,00%	19,19	**14,68**
sw100-8-lp2-c5-1	500	0,00%	22,23	**17,34**
sw100-8-lp3-c5-1	500	0,00%	24,91	**20,27**
sw100-8-lp4-c5-1	500	0,00%	27,98	**23,19**
sw100-8-lp5-c5-1	500	0,00%	29,98	**25,43**
sw100-8-lp6-c5-1	500	0,00%	30,72	**25,69**
sw100-8-lp7-c5-1	500	0,00%	31,25	**26,24**
sw100-8-lp8-c5-1	500	0,00%	31,68	**26,75**
sw100-8-p0-c5-1	500	0,00%	31,79	**27,17**
bw_anomaly	48	0,67%	8,68	**8,21**
bw_medium	116	0,15%	15,49	**14,77**
bw_huge	459	0,01%	**37,93**	43,41
bw_large.a	459	0,01%	**37,85**	41,98
bw_large.b	1087	0,00%	**58,97**	78,66
bw_large.c	3016	0,00%	**98,06**	160,70
bw_large.d	6325	0,00%	**133,83**	265,98
logistics.a	828	0,00%	49,45	**30,20**
logistics.b	843	0,01%	39,48	**22,78**
logistics.c	1141	0,00%	56,22	**33,29**
logistics.d	4713	0,00%	**276,23**	399,81
ais6	61	0,15%	2,23	**1,93**
ais8	113	0,04%	3,40	**2,92**
ais10	181	0,01%	4,34	**3,59**
ais12	265	0,00%	5,25	**4,46**
qg1-07	343	0,00%	37,26	**36,56**
qg1-08	512	0,00%	**46,29**	47,76
qg2-07	343	0,00%	**36,69**	38,31
qg2-08	512	0,00%	**47,06**	49,55
qg3-08	512	0,00%	77,82	**66,39**
qg3-09	729	0,00%	97,25	**82,44**
qg4-08	512	0,00%	70,61	**61,40**
qg4-09	729	0,00%	88,81	**77,00**
qg5-09	729	0,00%	122,38	**113,72**
qg5-10	1000	0,00%	153,18	**139,46**
qg5-11	1331	0,00%	187,56	**168,20**
qg5-12	1728	0,00%	222,36	**202,46**
qg5-13	2197	0,00%	263,75	**239,93**
qg6-09	729	0,00%	105,93	**99,58**
qg6-10	1000	0,00%	130,62	**123,89**
qg6-11	1331	0,00%	159,42	**148,69**
qg6-12	1728	0,00%	191,12	**178,70**
qg7-09	729	0,00%	118,49	**105,67**
qg7-10	1000	0,00%	145,32	**131,54**
qg7-11	1331	0,00%	174,86	**158,97**
qg7-12	1728	0,00%	209,06	**191,58**
qg7-13	2197	0,00%	244,80	**221,46**
bmc-ibm-1	9685	0,00%	676,73	**642,07**
bmc-ibm-2	2810	0,02%	292,17	**278,14**
bmc-ibm-3	14930	0,00%	1 772,67	**1 665,60**
bmc-ibm-4	28161	0,00%	2 968,92	**2 932,36**
bmc-ibm-5	9396	0,00%	**1 019,46**	1 036,26
bmc-ibm-6	51639	0,00%	4 658,90	**4 595,30**
bmc-ibm-7	8710	0,00%	896,42	**864,46**
bmc-galileo-8	58074	0,00%	**7 183,37**	7 913,57
bmc-galileo-9	63624	0,00%	**7 824,87**	8 657,53
bmc-ibm-10	59056	0,00%	**5 795,01**	6 119,76
bmc-ibm-11	32109	0,00%	**3 436,25**	3 476,99
bmc-ibm-12	39598	0,00%	**3 968,42**	4 064,75
bmc-ibm-13	13215	0,00%	1 478,34	**1 418,25**

TABLE 3.4 – Comparaison empirique des règles pivot sur MAXSAT : FIRST vs BEST.

la stratégie du meilleur améliorant a tendance à être plus efficace pour traiter les problèmes industriels [Whitley *et al.*, 2013]. Dans leur article, Whitley *et al.* expliquent que contrairement aux instances aléatoires où la distribution de l'interaction des variables entre elles est uniforme, les instances industrielles impliquent souvent des variables critiques ayant un fort impact sur la fonction d'évaluation et qui devraient être fixées tôt dans la recherche. Nous pensons que ces paysages ont des spécificités hétérogènes, c'est à dire que les indicateurs utilisés ici ne permettent pas de tenir compte de la variabilité de la rugosité de l'espace de recherche.

3.2.6 Nombre d'évaluations

Dans cette étude, nous nous sommes concentrés sur la capacité des climbers à atteindre de bons optima locaux, sans se focaliser sur le coût en termes de ressources. Mais cela a bien sûr de l'importance, surtout dans le cadre de l'intégration des climbers dans une métaheuristique avancée. `BEST` se doit d'évaluer la totalité du voisinage à chaque étape de la recherche. En revanche, intuitivement, `FIRST` nécessite de réaliser plus de mouvements avant d'atteindre un optimum local, puisqu'il accepte en moyenne des mouvements moins améliorants que `BEST`. Malgré cela, il est notable de préciser que logiquement, `FIRST` requiert systématiquement moins d'évaluations que `BEST` pour atteindre un optimum local. Le rapport moyen #eval(`FIRST`)/#eval(`BEST`), en considérant l'ensemble des tests réalisés sur l'ensemble des instances et l'ensemble des problèmes traités est approximativement de 13%. En particulier, ce rapport varie de $1{,}5\%$ à 19% pour les paysages NK, de 3% à 21% pour les paysages $\mathrm{QAP}_{\mathrm{swap}}$, de 7% à 38% pour les paysages $\mathrm{FSP}_{\mathrm{ins}}$ et de $0{,}05\%$ à 33% pour les instances MAXSAT. Pour l'ensemble des problèmes traités, ce rapport tend à décroître lorsque la taille de l'instance augmente.

3.2.7 Discussion

Nous avons étudié les 2 règles pivot classiques (`FIRST` et `BEST`) et nous les avons éprouvées sur un panel de paysages aussi large que possible. Les expérimentations réalisées, alliées aux études de paysages, nous ont permis de montrer que très souvent, `BEST` est plus performant que `FIRST` sur les paysages peu rugueux, tandis que la tendance s'inverse lorsque la rugosité des paysages est plus importante. À cela nous pouvons ajouter deux observations :

- Lorsque la rugosité est nulle, `BEST` et `FIRST` sont de performance égale, puisque le paysage ne contient pas d'épistasie et par conséquent ce type de paysage ne contient qu'un seul optimum local qui est l'optimum global (ceci a été montré dans [Poelwijk *et al.*, 2011]).
- La rugosité des paysages aux fitness aléatoires tend naturellement à être proche de 0,5. Sans perte de généralité, on peut supposer que les fitness d'un paysage aléatoire soient répartis de manière uniforme sur l'intervalle [0,1]. Intuitivement, pour une valeur de fitness donnée, la probabilité d'être sur un optimum local peut être calculée et ne dépend pas de la règle pivot considérée. La probabilité d'améliorer une configuration de fitness z est de $(1-z)^{|\mathcal{N}|}$. La probabilité d'atteindre ou de dépasser un fitness correspond au produit des probabilités de trouver une solution améliorante à chaque pas de la recherche. Par conséquent, la probabilité d'atteindre ou de dépasser un fitness sera plus élevée pour `BEST` que pour `FIRST` qui réalisera généralement plus de transitions pour atteindre ou dépasser le fitness but. Nous avons vérifié cette intuition par l'intermédiaire d'expérimentations sur des fonctions objectif aléatoires.

Au vu de ces deux observations et des expérimentations réalisées précédemment, il nous a semblé cohérent de penser que l'allure de la courbe de comparaison BEST/FIRST en fonction de la rugosité des paysages ait l'allure représentée dans la figure 3.4. Ceci n'a pas pu être vérifié expérimentalement, car le modèle NK ne permet pas de générer des paysages fortement rugueux pour des grands espaces de recherche car de telles instances nécessitent un espace de stockage exponentiel en fonction de K. De même, K prenant des valeurs entières, nous ne pouvons pas générer des paysages ayant des rugosités intermédiaires à celle obtenues pour les différentes valeurs de K, et en particulier pour les faibles rugosités. Une étude future pourrait permettre de vérifier ce phénomène, en considérant un autre modèle de type paysages NK, mais qui permettrait de régler la rugosité tout en contrôlant la taille des données et de générer des faibles taux de rugosité. Le modèle des paysages NM, proposé très récemment dans [Manukyan *et al.*, 2014] pourrait être adapté à une telle étude.

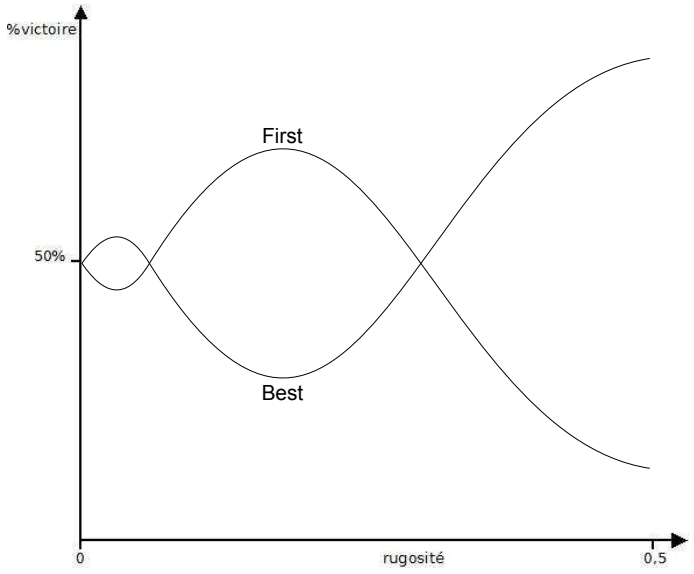


FIGURE 3.4 – Extrapolation de la comparaison First/Best en fonction de la rugosité des paysages, pour une taille fixée de l'espace de recherche.

Notons que des résultats présentés dans [Ochoa *et al.*, 2010] vont dans le sens de notre analyse et de la figure 3.4. En effet, Ochoa *et al.*ont montré par une étude exhaustive qu'il existe une corrélation entre taille des bassins d'attraction et fitness des optima locaux, pour différents niveaux de rugosité. Or, la taille du bassin d'attraction influe directement sur la probabilité d'atteindre l'optimum local correspondant. Ils obtiennent des résultats similaires à la figure 3.4, même si les faibles rugosités n'ont pas été testées et que la taille très limitée des instances étudiées fait que FIRST domine BEST sur un intervalle de rugosité plus restreint (car plus l'espace de recherche est petit, plus BEST tend à l'emporter sur FIRST).

Une grande part des paysages de recherches résultant de la modélisation de problèmes d'optimisation combinatoire réels induisent un taux de rugosité où FIRST s'avère être plus performant. En effet, BEST tend à être plus efficace sur les paysages faciles et sur ceux ayant une rugosité très élevée. Dans ce cas, le paysage est peu structuré est quasiment aléatoire, ce qui indique un problème de modélisation ou de struture de voisinage, ou que le problème ne peux être optimisé efficacement (dans le cas d'un paysage aléatoire, une recherche aléatoire est alors adaptée). Il

semble donc cohérent de privilégier FIRST, sauf si le problème semble peu difficile. Par exemple, le problème de voyageur de commerce avec distances euclidiennes, bien que NP, est plutôt simple à résoudre puisque des instances de très grandes tailles ont été résolues et on sait que la stratégie du meilleur améliorant est plus efficace pour ce problème [Hansen et Mladenovic, 2006].

3.3 Évaluation des politiques de prise en compte des mouvements neutres

Afin de compléter notre étude sur l'évaluation des différents climbers classiques de la littérature, nous nous sommes intéressés à l'évaluation des politiques de prise en compte des mouvements neutres. Dans la section 3.1, nous avons décrit trois climbers prenant en compte la neutralité durant la recherche. La première variante est l'approche stochastique (STOCH), qui consiste en une stratégie du premier améliorant où les mouvements neutres sont considérés comme améliorants et peuvent ainsi être acceptés durant toute la recherche. Les deux autres approches, plus classiques, consistent à accepter les mouvements neutres à chaque fois qu'aucune configuration du voisinage n'est meilleure que la configuration courante (FIRST+NP lorsque l'on considère la règle pivot du premier améliorant et BEST+NP pour le meilleur améliorant). Les climbers BEST+NP, FIRST+NP et STOCH sont décrits respectivement dans les algorithmes 3.3, 3.4 et 3.5.

Algorithme 3.3 Climber BEST+NP

Entrée : une configuration initiale x
tant que x n'est pas un optimum local et nombre maximal d'évaluations non atteint **faire**
  $x \leftarrow argmax_{x' \in \mathcal{N}(x)}(\mathfrak{f}(x'))$
Retourner x

Algorithme 3.4 Climber FIRST+NP

Entrée : une configuration initiale x
tant que x n'est pas un optimum local et nombre maximal d'évaluations non atteint **faire**
  si x est un optimum local non-strict
    alors sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}(x') = \mathfrak{f}(x)$
    sinon sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}(x') > \mathfrak{f}(x)$
  $x \leftarrow x'$
Retourner x

Algorithme 3.5 Climber STOCH

Entrée : une configuration initiale x
tant que x n'est pas un optimum local et nombre maximal d'évaluations non atteint **faire**
  sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}(x') \geqslant \mathfrak{f}(x)$
  $x \leftarrow x'$
Retourner x

Nous avons réalisé une série de tests sur 480 paysages :

– NKp : 64 paysages NKp, dérivés des 16 paysages NK basiques étudiés précédemment, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de p (0,50, 0,80, 0,95 et 0,99).

– NKq : 64 paysages NKq, dérivés des mêmes 16 paysages NK basiques, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de q (2, 3, 5 et 10).
– NKr : 64 paysages NKq, dérivés des mêmes 16 paysages NK basiques, l'ajout de la neutralité étant paramétrée par 4 valeurs différentes de r (10, 100, 1000 et 10000).
– Flow-Shop (27 paysages), QAP (132 paysages) et MAXSAT (129 paysages) : les mêmes instances que celles utilisées dans la section 3.2.

Le protocole expérimental est défini comme cela a été présenté dans la section précédente, sauf pour le critère d'arrêt. En effet, les climbers basiques (sans prise en compte de la neutralité) s'arrêtent systématiquement une fois un optimum local atteint, alors que l'exploration des plateaux des paysages de recherche implique que la recherche ne se termine généralement pas naturellement. Par conséquent, nous avons fixé un nombre maximal d'évaluations $\mathfrak{n}$, suffisamment grand pour laisser converger la recherche au maximum. Pour les paysages de recherche basés sur les chaînes de bits, nous avons fixé $\mathfrak{n} = 10.|\mathcal{N}_{\text{flip}}|^2$, et pour les paysages basés sur des permutations, nous avons fixé $\mathfrak{n} = 500.|\mathcal{N}_{\{\text{ins,swap}\}}|$.

Rappelons que les configurations initiales utilisées pour nos expérimentations sont les mêmes, quel que soit le climber considéré (`FIRST`, `BEST`, `STOCH`, `FIRST+NP`, `BEST+NP`).

3.3.1 Étude sur les paysages NK

La table 3.5 (a) récapitule les résultats obtenus sur les paysages NKp. Ces paysages ne permettent pas de conclure clairement sur l'efficacité relative des politiques de prise en compte de la neutralité. Cependant, `STOCH` semble légèrement plus performant puisqu'il est statistiquement dominé sur 8 paysages tandis que `FIRST+NP` est dominé sur 14 paysages et `BEST+NP` sur 12 paysages (avec $p < 0{,}05$). `BEST+NP` tend à dominer les variantes basées sur le premier améliorant sur les paysages avec des petits niveaux de neutralité (voir ν) et de rugosité ($K \in \{1{,}2\}$) ; `STOCH` atteint de meilleures solutions sur les paysages plutôt rugueux et neutres. Globalement, il y a peu de conclusions statistiques obtenues sur les paysages NKp. Cela est partiellement dû au fait que l'exploration de ces paysages est rendues facile par la méthode d'ajout de neutralité. En particulier, l'ensemble des tests statistiques effectués sur les paysages N_K_p99 ne sont pas significatifs (sauf `FIRST+NP` $\succ_{95\%}$ `BEST+NP` sur le paysage 512_8_p99).

Les résultats obtenus sur les paysages NKq (table 3.5 (b)) nous amènent à des conclusions moins nuancées. Les différences entre les climbers sont ici plus significatives statistiquement. En particulier, `STOCH` réalise les meilleures performances sur beaucoup de paysages, surtout lorsqu'ils sont caractérisés par un haut niveau de neutralité et/ou de rugosité. Ceci est un résultat intéressant, car beaucoup de méthodes tendent à intensifier la recherche avant de traiter la neutralité afin de sortir des plateaux, alors qu'il semble plus efficace de considérer les mouvements neutres tout au long de la recherche. On observe bien cette différence en se concentrant sur la comparaison entre `STOCH` et `FIRST+NP`, puisque la politique d'acceptation des mouvements neutres est la seule différence entre ces deux versions : sur les paysages NKq, `STOCH` domine en moyenne `FIRST+NP` sur 61 instances sur les 64 testées. `BEST+NP` reste cependant efficace pour traiter les paysages peu rugueux, en particulier lorsque ceux-ci possèdent un faible taux de neutralité. Dans ce cas, la supériorité de la règle pivot `BEST` n'est pas suffisamment contrebalancée par l'efficacité de `STOCH` qui utilise nécessairement `FIRST` comme règle pivot.

Les résultats obtenus sur les paysages NKr (table 3.5 (c)) confirment la tendance observée pour les paysages NKq. `STOCH` est meilleur que les autres variantes lorsque $\nu > 10\%$. Notons que les

Instance	ν	STOCH	F+NP	B+NP
128_1_p50	11,23%	0,4320	0,4317	**0,4366**
128_1_p80	44,55%	0,2241	0,2239	**0,2253**
128_1_p95	81,75%	0,0651	0,0651	**0,0653**
128_1_p99	94,92%	**0,0146**	**0,0146**	**0,0146**
128_2_p50	5,57%	0,4754	0,4763	**0,4806**
128_2_p80	30,85%	**0,2876**	0,2840	0,2868
128_2_p95	73,62%	0,1130	0,1131	**0,1138**
128_2_p99	94,40%	0,0305	0,0305	**0,0306**
128_4_p50	1,17%	**0,5257**	0,5250	0,5233
128_4_p80	14,52%	**0,3380**	0,3367	0,3363
128_4_p95	60,50%	**0,1780**	0,1774	0,1769
128_4_p99	89,66%	0,0788	0,0785	**0,0789**
128_8_p50	0,05%	**0,5233**	0,5223	0,5161
128_8_p80	3,42%	**0,3367**	0,3346	0,3317
128_8_p95	41,32%	**0,1922**	0,1901	0,1896
128_8_p99	83,39%	**0,1138**	0,1123	0,1129
256_1_p50	11,91%	0,4237	0,4235	**0,4283**
256_1_p80	45,09%	0,2208	0,2210	**0,2222**
256_1_p95	81,46%	**0,0655**	0,0654	**0,0655**
256_1_p99	96,11%	**0,0139**	**0,0139**	**0,0139**
256_2_p50	5,40%	0,4795	0,4788	**0,4850**
256_2_p80	30,82%	0,2888	0,2893	**0,2915**
256_2_p95	74,86%	0,1169	0,1170	**0,1176**
256_2_p99	94,22%	0,0303	0,0303	**0,0304**
256_4_p50	1,18%	0,5241	**0,5268**	0,5230
256_4_p80	15,19%	**0,3386**	0,3381	0,3364
256_4_p95	60,85%	0,1800	0,1775	**0,1801**
256_4_p99	90,77%	**0,0791**	0,0790	0,0788
256_8_p50	0,05%	0,5251	**0,5261**	0,5208
256_8_p80	3,55%	**0,3409**	0,3364	0,3342
256_8_p95	41,25%	**0,1931**	0,1901	0,1918
256_8_p99	83,70%	**0,1155**	0,1150	0,1144
512_1_p50	12,17%	0,4195	0,4195	**0,4238**
512_1_p80	45,27%	0,2183	0,2184	**0,2199**
512_1_p95	81,69%	0,0674	0,0674	**0,0676**
512_1_p99	96,29%	**0,0142**	**0,0142**	**0,0142**
512_2_p50	5,39%	0,4853	0,4855	**0,4901**
512_2_p80	31,18%	0,2912	0,2907	**0,2934**
512_2_p95	74,31%	0,1184	0,1187	**0,1192**
512_2_p99	94,57%	**0,0317**	**0,0317**	**0,0317**
512_4_p50	1,23%	0,5228	**0,5231**	0,5216
512_4_p80	15,24%	0,3346	**0,3357**	0,3341
512_4_p95	61,17%	0,1782	0,1767	**0,1783**
512_4_p99	90,19%	**0,0785**	0,0780	0,0781
512_8_p50	0,05%	0,5325	**0,5327**	0,5260
512_8_p80	3,51%	0,3419	**0,3432**	0,3383
512_8_p95	41,40%	**0,1936**	0,1929	0,1935
512_8_p99	83,64%	0,1158	**0,1162**	0,1147
1024_1_p50	12,00%	0,4270	0,4272	**0,4311**
1024_1_p80	44,70%	0,2231	0,2231	**0,2246**
1024_1_p95	81,67%	0,0683	0,0683	**0,0685**
1024_1_p99	96,10%	**0,0147**	**0,0147**	**0,0147**
1024_2_p50	5,62%	0,4861	0,4864	**0,4904**
1024_2_p80	31,46%	0,2915	0,2916	**0,2937**
1024_2_p95	74,31%	0,1189	0,1188	**0,1194**
1024_2_p99	94,15%	0,0319	0,0319	**0,0320**
1024_4_p50	1,23%	**0,5269**	0,5252	0,5258
1024_4_p80	15,11%	**0,3378**	0,3357	0,3361
1024_4_p95	61,24%	0,1787	0,1779	**0,1796**
1024_4_p99	90,72%	**0,0784**	0,0783	0,0783
1024_8_p50	0,07%	0,5317	**0,5323**	0,5270
1024_8_p80	3,56%	**0,3449**	0,3444	0,3397
1024_8_p95	41,27%	**0,1959**	0,1943	0,1943
1024_8_p99	83,62%	**0,1163**	**0,1163**	0,1160

(a) NKp

Instance	ν	STOCH	F+NP	B+NP
128_1_q10	6,35%	0,7294	0,7283	**0,7304**
128_1_q5	15,21%	0,7484	0,7478	**0,7507**
128_1_q3	24,38%	0,7902	0,7909	**0,7950**
128_1_q2	41,12%	**0,8594**	**0,8594**	**0,8594**
128_2_q10	4,76%	0,7297	0,7300	**0,7345**
128_2_q5	12,42%	**0,7687**	0,7664	0,7683
128_2_q3	21,32%	**0,8186**	0,8175	0,8176
128_2_q2	33,33%	**0,8914**	0,8898	0,8904
128_4_q10	2,91%	0,7557	0,7553	**0,7567**
128_4_q5	9,37%	**0,7922**	0,7893	0,7912
128_4_q3	16,21%	**0,8396**	0,8376	0,8372
128_4_q2	26,16%	**0,9309**	0,9273	0,9265
128_8_q10	1,75%	**0,7417**	0,7383	0,7351
128_8_q5	6,85%	**0,7734**	0,7693	0,7652
128_8_q3	11,80%	**0,8199**	0,8145	0,8126
128_8_q2	19,23%	**0,9020**	0,8941	0,8965
256_1_q10	6,62%	0,7271	0,7267	**0,7323**
256_1_q5	15,26%	0,7518	0,7516	**0,7557**
256_1_q3	25,03%	0,7938	0,7938	**0,7948**
256_1_q2	39,88%	**0,8672**	0,8670	0,8668
256_2_q10	4,73%	0,7318	0,7307	**0,7340**
256_2_q5	12,21%	0,7610	0,7608	**0,7640**
256_2_q3	21,29%	0,8183	0,8159	**0,8193**
256_2_q2	33,26%	**0,8907**	0,8879	0,8886
256_4_q10	2,99%	**0,7503**	0,7498	0,7490
256_4_q5	9,44%	**0,7865**	0,7838	0,7819
256_4_q3	16,18%	**0,8436**	0,8407	0,8429
256_4_q2	26,15%	**0,9363**	0,9297	0,9288
256_8_q10	1,78%	**0,7425**	0,7402	0,7369
256_8_q5	6,85%	**0,7762**	0,7741	0,7710
256_8_q3	11,84%	**0,8255**	0,8196	0,8200
256_8_q2	19,26%	**0,9098**	0,9039	0,9018
512_1_q10	6,70%	0,7126	0,7122	**0,7170**
512_1_q5	15,19%	0,7408	0,7402	**0,7430**
512_1_q3	26,07%	0,7823	0,7824	**0,7835**
512_1_q2	40,16%	**0,8368**	0,8354	0,8355
512_2_q10	4,65%	0,7399	0,7385	**0,7422**
512_2_q5	12,35%	0,7722	0,7704	**0,7724**
512_2_q3	21,03%	0,8187	0,8175	**0,8196**
512_2_q2	33,47%	**0,9060**	0,9023	0,9021
512_4_q10	3,02%	**0,7494**	0,7461	0,7468
512_4_q5	9,36%	**0,7862**	0,7824	0,7812
512_4_q3	16,14%	**0,8398**	0,8356	0,8351
512_4_q2	26,15%	**0,9348**	0,9291	0,9286
512_8_q10	1,80%	**0,7457**	0,7454	0,7413
512_8_q5	6,84%	**0,7804**	0,7760	0,7765
512_8_q3	11,84%	**0,8320**	0,8260	0,8248
512_8_q2	19,25%	**0,9132**	0,9074	0,9087
1024_1_q10	6,55%	0,7203	0,7195	**0,7256**
1024_1_q5	15,23%	0,7514	0,7509	**0,7550**
1024_1_q3	25,87%	0,7955	0,7953	**0,7971**
1024_1_q2	40,51%	**0,8416**	0,8411	0,8414
1024_2_q10	4,66%	0,7434	0,7422	**0,7462**
1024_2_q5	12,33%	0,7768	0,7763	**0,7783**
1024_2_q3	21,22%	0,8259	0,8253	**0,8260**
1024_2_q2	33,65%	**0,9067**	0,9036	0,9045
1024_4_q10	3,02%	**0,7540**	0,7511	0,7501
1024_4_q5	9,40%	**0,7897**	0,7850	0,7858
1024_4_q3	16,18%	**0,8469**	0,8424	0,8432
1024_4_q2	26,18%	**0,9348**	0,9286	0,9294
1024_8_q10	1,78%	**0,7479**	0,7467	0,7428
1024_8_q5	6,85%	**0,7830**	0,7789	0,7772
1024_8_q3	11,85%	**0,8350**	0,8299	0,8292
1024_8_q2	19,27%	**0,9191**	0,9125	0,9121

(b) NKq

Instance	ν	STOCH	F+NP	B+NP
128_1_r10000	0,80%	0,7023	0,7017	**0,7087**
128_1_r1000	9,38%	0,7082	0,7057	**0,7103**
128_1_r100	65,57%	0,7127	**0,7131**	0,7125
128_1_r10	93,68%	**0,5250**	0,5000	0,5000
128_2_r10000	0,66%	0,7061	0,7036	**0,7088**
128_2_r1000	7,72%	**0,7155**	0,7108	0,7132
128_2_r100	58,47%	**0,7313**	0,7299	0,7306
128_2_r10	93,07%	**0,5310**	0,5020	0,5020
128_4_r10000	0,59%	**0,7279**	0,7256	0,7264
128_4_r1000	5,85%	**0,7362**	0,7304	0,7341
128_4_r100	49,50%	**0,7750**	0,7608	0,7625
128_4_r10	91,13%	**0,6000**	0,5340	0,5330
128_8_r10000	0,43%	0,7145	**0,7156**	0,7120
128_8_r1000	4,27%	**0,7216**	0,7176	0,7138
128_8_r100	39,01%	**0,7510**	0,7421	0,7353
128_8_r10	88,16%	**0,6000**	0,5560	0,5560
256_1_r10000	1,82%	0,7046	0,7039	**0,7091**
256_1_r1000	17,86%	**0,7134**	0,7095	0,7117
256_1_r100	82,23%	**0,6874**	0,6848	0,6842
256_1_r10	95,85%	**0,5000**	**0,5000**	**0,5000**
256_2_r10000	1,59%	0,7075	0,7062	**0,7104**
256_2_r1000	15,09%	**0,7211**	0,7157	0,7168
256_2_r100	78,39%	**0,7150**	0,7124	0,7117
256_2_r10	95,20%	**0,5000**	**0,5000**	**0,5000**
256_4_r10000	1,18%	**0,7243**	0,7232	0,7222
256_4_r1000	11,73%	**0,7410**	0,7312	0,7311
256_4_r100	72,10%	**0,7720**	0,7575	0,7599
256_4_r10	93,68%	**0,5000**	**0,5000**	**0,5000**
256_8_r10000	0,86%	**0,7216**	0,7177	0,7135
256_8_r1000	8,57%	**0,7282**	0,7224	0,7227
256_8_r100	62,97%	**0,7714**	0,7575	0,7583
256_8_r10	91,56%	**0,5010**	0,5000	0,5000
512_1_r10000	3,97%	0,6932	0,6923	**0,6966**
512_1_r1000	34,48%	**0,7054**	0,7016	0,7023
512_1_r100	91,18%	**0,6296**	0,6021	0,6017
512_1_r10	97,09%	**0,5000**	**0,5000**	**0,5000**
512_2_r10000	3,08%	0,7170	0,7149	**0,7189**
512_2_r1000	28,95%	**0,7398**	0,7298	0,7301
512_2_r100	89,20%	**0,6667**	0,6443	0,6430
512_2_r10	96,49%	**0,5000**	**0,5000**	**0,5000**
512_4_r10000	2,35%	**0,7240**	0,7228	0,7222
512_4_r1000	22,73%	**0,7543**	0,7400	0,7383
512_4_r100	86,00%	**0,6926**	0,6784	0,6794
512_4_r10	95,59%	**0,5000**	**0,5000**	**0,5000**
512_8_r10000	1,72%	**0,7228**	0,7220	0,7192
512_8_r1000	16,90%	**0,7429**	0,7310	0,7304
512_8_r100	81,05%	0,7066	0,7115	**0,7129**
512_8_r10	94,02%	**0,5000**	**0,5000**	**0,5000**
1024_1_r10000	7,53%	0,7031	0,6999	**0,7059**
1024_1_r1000	58,10%	**0,7135**	0,7115	0,7114
1024_1_r100	95,57%	**0,5857**	0,5498	0,5492
1024_1_r10	97,88%	**0,5000**	**0,5000**	**0,5000**
1024_2_r10000	6,17%	0,7228	0,7189	**0,7233**
1024_2_r1000	50,82%	**0,7498**	0,7418	0,7418
1024_2_r100	94,65%	**0,6043**	0,5657	0,5652
1024_2_r10	97,54%	**0,5000**	**0,5000**	**0,5000**
1024_4_r10000	4,71%	**0,7328**	0,7280	0,7276
1024_4_r1000	41,83%	**0,7748**	0,7592	0,7589
1024_4_r100	92,95%	**0,6158**	0,5832	0,5851
1024_4_r10	96,93%	**0,5000**	**0,5000**	**0,5000**
1024_8_r10000	3,44%	**0,7277**	0,7233	0,7229
1024_8_r1000	32,29%	**0,7617**	0,7469	0,7481
1024_8_r100	90,48%	**0,6208**	0,6014	0,6016
1024_8_r10	95,80%	**0,5000**	**0,5000**	**0,5000**

(c) NKr

TABLE 3.5 – Comparaison des politiques de prise en compte de la neutralité sur les paysages NKp, NKq et NKr.

Fitness	F_{NK}	F_{NKr}	
Arrondi	–	r = 1000	r = 100
Climber	$\star$	STOCH	STOCH
128_1	0,7090	0,7082	**0,7127**
128_2	0,7082	0,7155	**0,7313**
128_4	0,7260	0,7362	**0,7750**
128_8	0,7142	0,7216	**0,7510**
256_1	0,7079	**0,7134**	0,6874
256_2	0,7094	**0,7211**	0,7150
256_4	0,7235	0,7410	**0,7720**
256_8	0,7166	0,7282	**0,7714**

Fitness	F_{NK}	F_{NKr}	
Arrondi	–	r = 1000	r = 100
Climber	$\star$	STOCH	STOCH
512_1	0,6953	**0,7054**	0,6296
512_2	0,7174	**0,7398**	0,6667
512_4	0,7200	**0,7543**	0,6926
512_8	0,7206	**0,7429**	0,7066
1024_1	0,7039	**0,7135**	0,5857
1024_2	0,7197	**0,7498**	0,6043
1024_4	0,7246	**0,7748**	0,6158
1024_8	0,7216	**0,7617**	0,6208

TABLE 3.6 – L'ajout de neutralité dans des paysages non-neutres permet d'atteindre de meilleures configurations avec les climbers stochastiques : comparaison entre le climber le plus efficace pour traiter le paysage NK original (F_{NK}) et STOCH sur un paysage équivalent avec fonction de fitness arrondie (F_{NKr} — $r = \{100,1000\}$). La colonne $\star$ correspond à $\max$(FIRST,BEST).

paysages engendrés avec $r = 10$ sont tellement plats que les différentes variantes sont incapables de sortir des gigantesques plateaux et sont donc statistiquement incomparables.

Il est important de rappeler que chaque instance de paysage NKr correspond exactement à une instance NK basique. La différence entre deux paysages NK et NKr se situe exclusivement sur la précision de la fonction d'évaluation. Puisque, d'après notre définition des paysages NKr, $\forall x \in \mathcal{X}, F_{\mathrm{NKr}}(x,r) \leqslant F_{\mathrm{NK}}(x) < F_{\mathrm{NKr}}(x,r) + \frac{1}{r}$, il est cohérent de comparer directement les résultats obtenus sur les instances NK et NKr pour chaque couple (N,K). En observant les tables 3.1 et 3.5 (c), on remarque que certaines valeurs de r permettent à STOCH d'atteindre de meilleures configurations, bien que l'arrondi réalisé lui soit défavorable (les fitness sont tronqués). Dans la table 3.6 nous avons reporté les résultats obtenus sur les paysages NK basiques, ainsi que les résultats de STOCH sur les paysages NKr correspondants, pour $r = 100$ et $r = 1000$. Pour chaque paysage NK testé, l'une des valeurs d'arrondi permet d'améliorer de manière très significative la qualité des configurations atteintes par STOCH. Il est également clair que l'intérêt d'arrondir la fonction de fitness n'est pas systématique et dépend fortement de la valeur de r. Cependant les résultats obtenus dans ces expérimentations montrent qu'arrondir la fonction de fitness devient très utile lorsque la valeur de r utilisée définit un paysage dont le taux de neutralité est compris entre 15% et 75% (voir la table 3.5 (c)). En conclusion, pour chaque couple (N,K), il semble exister un arrondi approprié de la fonction de fitness qui permet d'améliorer fortement l'efficacité des climbers exploitant cette neutralité. En fait, la neutralité ajoutée permet de faire décroître la rugosité $\rho_k^{\leqslant}$ mais augmente également le nombre de mouvements nécessaires pour atteindre un optima local. Trouver une bonne valeur de r consiste donc à trouver un bon compromis entre chercher à réduire la rugosité du paysage initial tout en gardant un taux de neutralité suffisamment faible pour converger en un temps raisonnable. Une étude plus poussée de ce paramètre r a été réalisée lors du stage de master recherche de Hugo Traverson réalisé en 2014 (voir section 4.2).

3.3.2 Étude sur le problème de flow-shop

La table 3.7 nous permet d'observer une forte supériorité de STOCH sur les paysages $\mathrm{FSP}_{\mathrm{ins}}$. En effet, STOCH obtient toujours le meilleur fitness moyen et domine statistiquement FIRST+NP et BEST+NP sur 22 des 27 paysages testés (avec $p > 0{,}95$). Pour les cinq paysages où la dominance statistique n'est pas clairement établie, trois sont ceux où la neutralité est la plus faible

Instance	ν	STOCH	F+NP	B+NP
20_5_01_ta001	21,54%	**1 278,00**	1 280,61	1 281,42
20_10_01_ta011	9,85%	**1 609,17**	1 613,42	1 611,15
20_15_01	8,89%	**1 925,14**	1 931,98	1 930,95
20_20_01_ta021	5,65%	**2 350,99**	2 355,69	2 370,98
30_5_01	24,68%	**1 726,00**	1 727,08	1 726,36
30_10_01	11,79%	**2 014,29**	2 028,54	2 029,57
30_15_01	8,72%	**2 449,63**	2 461,49	2 461,44
30_20_01	5,25%	**2 825,35**	2 830,29	2 840,90
50_5_01_ta031	33,00%	**2 724,35**	2 734,82	2 732,78
50_10_01_ta041	13,77%	**3 038,76**	3 077,42	3 077,97
50_15_01	8,88%	**3 442,74**	3 467,12	3 472,52
50_20_01_ta051	7,09%	**3 958,07**	3 973,89	3 982,11
70_5_01	33,84%	**3 792,00**	**3 792,00**	**3 792,00**

Instance	ν	STOCH	F+NP	B+NP
70_10_01	15,55%	**4 059,78**	4 088,27	4 089,77
70_15_01	10,23%	**4 457,23**	4 471,74	4 475,01
70_20_01	7,96%	**4 835,74**	4 873,36	4 889,42
100_5_01_ta061	32,08%	**5 493,00**	5 493,86	5 493,64
100_10_01_ta071	17,65%	**5 780,09**	5 813,09	5 820,24
100_15_01	11,18%	**5 974,85**	6 037,93	6 048,83
100_20_01_ta081	9,04%	**6 329,26**	6 416,62	6 428,95
150_5_01	35,59%	**8 064,00**	8 064,06	8 064,02
150_10_01	17,82%	**8 115,65**	8 166,79	8 174,62
150_15_01	12,63%	**8 543,08**	8 623,12	8 626,60
150_20_01	9,57%	**9 024,92**	9 158,25	9 184,84
200_10_01_ta091	20,45%	**10 875,40**	10 917,80	10 926,40
200_15_01	13,17%	**11 146,10**	11 249,00	11 288,80
200_20_01_ta101	10,71%	**11 337,60**	11 460,50	11 478,60

TABLE 3.7 – Comparaison des politiques de prise en compte de la neutralité sur les paysages $\mathrm{FSP}_{\mathrm{ins}}$.

et les deux autres sont des paysages faciles où un optimum global est quasiment toujours atteint. L'intérêt d'accepter les mouvements neutres tout au long de la recherche est encore plus clair ici que pour les paysages NKp, NKq et NQr.

Notons que sur les paysages dérivés des instances de flow shop à 5 machines, STOCH obtient systématiquement un optimum global sur l'ensemble des exécutions, ce qui est remarquable pour un simple climber (FIRST+NP et BEST+NP n'atteignent un optimum global systématiquement que pour l'instance 70_5_01).

3.3.3 Étude sur le problème d'assignement quadratique

Les résultats obtenus sur les paysages $\mathrm{QAP}_{\mathrm{swap}}$ se trouvent partiellement dans la table 3.8 (les résultats de 55 des 132 paysages n'apparaissent pas car ils ne contiennent pas de neutralité — dans le cas de paysages sans neutralité, FIRST $\leftrightarrow$ STOCH $\leftrightarrow$ FIRST+NP et BEST $\leftrightarrow$ BEST+NP. La supériorité de STOCH demeure, bien que moins marquée que pour le flow shop : STOCH n'est dominé statistiquement que sur 6 paysages et obtient les meilleurs résultats moyens sur 44 des 77 paysages, tandis que FIRST+NP et BEST+NP sont dominés statistiquement sur respectivement 14 et 23 paysages.

3.3.4 Étude sur le problème de satisfiabilité

Les résultats sur les paysages MAXSAT contenus dans la table 3.9 nous amènent à des conclusions relativement claires. Pour les instances 3-SAT aléatoires (à gauche de la table), STOCH domine statistiquement FIRST+NP et BEST+NP sur toutes les instances sauf une. Ces paysages 3-SAT contiennent tous un taux important de transitions neutres ($\nu > 20\%$), qui se doivent donc d'être considérées tout au long de la recherche afin d'atteindre de meilleurs optima locaux. Les valeurs de la table correspondent au nombre moyen de clauses non satisfaites des instances, qui sont toutes satisfiables. Il est donc assez facile d'évaluer les différences numériques entre les climbers, qui s'avèrent assez importantes au bénéfice de STOCH. Par exemple, l'instance uf20-01 est toujours résolue par une simple descente stochastique mais pas par les autres climbers ; l'instance CBS_...m423_b10_0 est très souvent résolue par STOCH, et assez rarement par FIRST+NP et

Instance	ν	STOCH	F+NP	B+NP
bur26a	1,03%	5444414,60	**5444336,06**	5444520,51
bur26b	2,04%	3834718,97	3835096,79	**3832615,88**
bur26c	1,07%	5450892,49	5452501,44	**5446368,78**
bur26d	2,13%	3840939,53	3842342,53	**3836004,37**
bur26e	1,12%	**5405722,15**	5407828,63	5407338,95
bur26f	2,27%	**3798790,73**	3800973,99	3799057,10
bur26g	1,01%	10160063,12	10161134,69	**10157156,05**
bur26h	1,99%	7135315,56	7133939,91	**7126390,10**
chr18b	2,51%	**1703,40**	1721,20	1726,86
chr20a	0,64%	3212,68	3231,72	**3169,92**
chr20b	0,35%	3315,66	3286,38	**3190,36**
chr20c	0,14%	26552,40	26531,86	**26232,04**
chr22a	0,27%	**6974,84**	6995,66	7019,04
chr22b	0,21%	7085,58	7098,36	**7037,92**
chr25a	0,32%	6059,52	**5981,30**	6011,06
els19	0,40%	22069953,06	22280065,14	**21630921,32**
esc16a	30,06%	**68,06**	68,14	68,08
esc16b	38,94%	**292,00**	**292,00**	**292,00**
esc32a	12,04%	**145,16**	148,22	147,62
esc32b	17,67%	**190,36**	192,24	193,60
esc32d	32,56%	**204,04**	204,94	205,24
esc32h	25,74%	**448,18**	448,66	448,58
esc64a	54,71%	**116,00**	**116,00**	**116,00**
had12	3,94%	1671,44	**1669,98**	1670,10
had14	3,44%	**2744,54**	2746,32	2746,08
had16	3,13%	**3747,12**	3749,32	3751,20
had18	2,64%	**5412,96**	5416,50	5414,84
had20	2,43%	6993,56	7000,22	**6991,48**
kra30a	2,99%	**94667,60**	95086,90	94895,90
kra30b	2,95%	**95624,80**	96005,70	95922,50
kra32	3,11%	**18831,74**	18839,70	18881,92
lipa20a	2,68%	**3780,12**	3784,79	3783,16
lipa30a	1,98%	13439,05	13442,60	**13436,30**
lipa40a	1,31%	**32002,44**	32006,16	32009,00
lipa50a	1,05%	62888,65	**62879,74**	62897,39
lipa60a	0,84%	**108378,27**	108387,58	108395,95
lipa70a	0,72%	**171365,66**	171372,55	171385,86
lipa80a	0,62%	255300,82	**255292,75**	255307,09
lipa90a	0,56%	**363365,85**	363370,20	363418,80

Instance	ν	STOCH	F+NP	B+NP
nug12	3,25%	606,68	**604,08**	604,44
nug14	2,69%	**1059,26**	1061,56	1059,88
nug15	2,57%	1200,34	1201,06	**1194,86**
nug16a	2,23%	1681,06	**1678,26**	1684,94
nug16b	2,33%	**1301,08**	1303,56	1302,38
nug17	2,21%	**1795,80**	1804,94	1796,28
nug18	2,12%	2011,66	2010,98	**2010,28**
nug20	1,85%	2666,82	**2666,22**	2666,98
nug21	1,90%	**2531,28**	2537,06	2542,84
nug22	1,39%	3717,00	3720,48	**3714,20**
nug24	1,71%	**3638,90**	3640,66	3639,40
nug25	1,81%	**3866,14**	3869,62	3872,48
nug27	1,37%	**5427,66**	5442,74	5438,44
nug28	1,46%	**5370,82**	5379,50	5391,24
nug30	1,39%	**6347,18**	6353,74	6361,70
scr12	0,26%	33644,78	**33597,04**	33635,92
scr15	0,17%	57154,28	56970,38	**56397,80**
sko100a	0,44%	**154586,08**	154712,46	155045,02
sko100b	0,43%	**156489,74**	156598,26	156934,74
sko100c	0,45%	**150833,48**	150940,88	151165,18
sko100d	0,44%	152215,78	**152210,86**	152571,90
sko100e	0,43%	**152097,68**	152102,46	152500,18
sko100f	0,45%	**151587,40**	151674,60	152070,20
sko42	0,99%	**16302,60**	16303,52	16346,60
sko49	0,91%	**23994,28**	24023,54	24109,10
sko56	0,72%	35343,80	**35338,98**	35410,80
sko64	0,69%	**49563,76**	49639,60	49719,14
sko72	0,59%	**67715,28**	67847,30	67915,30
sko81	0,53%	**92739,72**	92769,58	92953,98
sko90	0,49%	**117689,38**	117810,48	117995,16
ste36a	0,58%	**10570,36**	10578,32	10616,38
ste36b	0,28%	18992,64	**18969,84**	19038,52
ste36c	0,17%	8961529,02	**8945385,72**	9021250,54
tai64c	67,12%	1866610,72	1865318,78	**1864589,22**
tho30	0,04%	157150,58	157184,58	**156876,82**
tho40	0,04%	251436,92	**251297,32**	251889,96
wil100	0,43%	**275608,98**	275780,80	275881,18
wil50	0,79%	49530,32	49528,60	**49528,34**

TABLE 3.8 – Comparaison des politiques de prise en compte de la neutralité sur les paysages $\mathrm{QAP}_{\mathrm{swap}}$.

Instances 3-SAT aléatoires					
Instance	n	ν	STOCH	F+NP	B+NP
uf20-01	20	20,93%	**0,00**	0,09	0,14
uf50-01	50	22,53%	**0,93**	1,13	1,33
uuf50-01	50	24,08%	1,70	**1,69**	1,84
uf75-01	75	23,67%	**1,05**	1,61	1,88
uuf75-01	75	23,69%	**1,73**	2,13	2,27
uf100-01	100	23,23%	**1,43**	2,28	2,18
uuf100-01	100	23,84%	**2,45**	2,96	2,92
uf125-01	125	23,46%	**1,51**	2,10	2,22
uuf125-01	125	23,77%	**2,60**	3,20	3,26
uf150-01	150	23,12%	**1,16**	1,90	2,17
uuf150-01	150	23,47%	**2,22**	3,01	3,03
uf175-01	175	22,91%	**2,29**	3,71	3,31
uuf175-01	175	23,27%	**3,29**	4,03	4,28
uf200-01	200	23,15%	**2,42**	3,78	3,95
uuf200-01	200	23,67%	**4,36**	5,62	5,81
uf225-01	225	23,11%	**1,40**	3,08	2,77
uuf225-01	225	23,37%	**2,94**	4,71	4,89
uf250-01	250	22,92%	**0,74**	2,71	2,64
uuf250-01	250	23,46%	**3,09**	4,85	4,98
RTI_...m429_0	100	23,32%	**0,85**	1,46	1,48
BMS_...m429_0	100	33,46%	**0,98**	1,24	1,16
CBS_...m403_b10_0	100	24,17%	**0,59**	1,39	1,27
CBS_...m403_b30_0	100	24,18%	**1,05**	1,52	1,55
CBS_...m403_b50_0	100	24,20%	**1,35**	2,04	2,07
CBS_...m403_b70_0	100	24,27%	**1,58**	1,90	2,01
CBS_...m403_b90_0	100	25,10%	**1,97**	2,57	2,58
CBS_...m411_b10_0	100	23,48%	**0,62**	0,99	1,39
CBS_...m411_b30_0	100	24,42%	**1,18**	1,76	1,73
CBS_...m411_b50_0	100	24,71%	**1,18**	1,86	1,86
CBS_...m411_b70_0	100	24,71%	**1,27**	2,01	2,06
CBS_...m411_b90_0	100	24,32%	**1,45**	2,32	2,19
CBS_...m418_b10_0	100	23,41%	**0,61**	0,99	0,90
CBS_...m418_b30_0	100	23,25%	**1,00**	1,51	1,78
CBS_...m418_b50_0	100	23,37%	**1,46**	2,00	1,96
CBS_...m418_b70_0	100	23,37%	**1,23**	1,67	1,74
CBS_...m418_b90_0	100	23,71%	**1,68**	2,05	2,05
CBS_...m423_b10_0	100	23,05%	**0,32**	0,79	0,89
CBS_...m423_b30_0	100	22,94%	**0,87**	1,55	1,60
CBS_...m423_b50_0	100	23,32%	**1,42**	2,05	1,85
CBS_...m423_b70_0	100	22,73%	**1,27**	1,56	1,80
CBS_...m423_b90_0	100	23,32%	**1,48**	1,85	1,94
CBS_...m429_b10_0	100	22,84%	**0,62**	1,25	1,16
CBS_...m429_b30_0	100	22,71%	**0,65**	1,41	1,37
CBS_...m429_b50_0	100	23,20%	**1,74**	2,21	2,23
CBS_...m429_b70_0	100	23,16%	**0,59**	1,41	1,49
CBS_...m429_b90_0	100	23,16%	**1,63**	2,36	2,41
CBS_...m435_b10_0	100	22,90%	**0,06**	0,62	0,56
CBS_...m435_b30_0	100	23,05%	**1,21**	1,69	1,84
CBS_...m435_b50_0	100	22,51%	**1,54**	1,88	1,94
CBS_...m435_b70_0	100	23,46%	**1,29**	1,96	2,05
CBS_...m435_b90_0	100	22,90%	**1,95**	2,28	2,38
CBS_...m441_b10_0	100	22,33%	**0,70**	1,28	1,16
CBS_...m441_b30_0	100	22,62%	**0,67**	0,98	1,13
CBS_...m441_b50_0	100	22,71%	**1,31**	1,90	1,76
CBS_...m441_b70_0	100	22,07%	**1,80**	2,36	2,18
CBS_...m441_b90_0	100	23,27%	**2,10**	2,81	2,56
CBS_...m449_b10_0	100	22,11%	**0,38**	0,83	0,80
CBS_...m449_b30_0	100	22,68%	**1,40**	1,74	1,83
CBS_...m449_b50_0	100	22,09%	**1,41**	1,71	1,74
CBS_...m449_b70_0	100	22,65%	**1,06**	1,75	1,61
CBS_...m449_b90_0	100	23,16%	**2,00**	2,61	2,88

Instances basées sur des problèmes					
Instance	n	ν	STOCH	F+NP	B+NP
flat30-1	90	6,68%	**0,82**	0,98	**0,82**
flat50-1	150	5,55%	**2,35**	2,47	2,50
flat75-1	225	5,47%	**4,35**	**4,35**	4,49
flat100-1	300	5,37%	**5,34**	5,53	5,63
flat125-1	375	5,05%	**6,22**	6,86	6,61
flat150-1	450	5,11%	**7,75**	8,09	8,03
flat175-1	525	5,34%	8,82	9,36	**8,77**
flat200-1	600	5,23%	**9,35**	9,50	9,37
sw100-8-lp0-c5-1	500	0,46%	**1,27**	1,52	1,40
sw100-8-lp1-c5-1	500	0,35%	0,67	0,81	**0,57**
sw100-8-lp2-c5-1	500	0,31%	**0,66**	0,92	0,89
sw100-8-lp3-c5-1	500	0,25%	**1,62**	1,65	1,69
sw100-8-lp4-c5-1	500	0,22%	**1,32**	2,21	2,14
sw100-8-lp5-c5-1	500	0,21%	**1,11**	3,04	3,14
sw100-8-lp6-c5-1	500	0,20%	**1,24**	3,92	3,74
sw100-8-lp7-c5-1	500	0,20%	**1,14**	4,59	4,47
sw100-8-lp8-c5-1	500	0,20%	**0,04**	5,12	5,01
sw100-8-p0-c5-1	500	0,19%	**0,06**	5,28	5,22
bw_anomaly	48	9,59%	**2,17**	2,59	3,06
bw_medium	116	5,18%	**4,88**	5,80	6,29
bw_huge	459	1,48%	**12,48**	13,37	14,58
bw_large.a	459	2,44%	**8,88**	10,14	10,52
bw_large.b	1087	1,19%	**14,21**	16,98	16,53
bw_large.c	3016	0,41%	**25,45**	27,62	27,12
bw_large.d	6325	0,17%	**36,50**	39,65	38,38
logistics.a	828	1,94%	20,56	21,22	**19,40**
logistics.b	843	2,55%	15,71	16,21	**15,12**
logistics.c	1141	1,71%	22,23	23,50	**21,62**
logistics.d	4713	9,04%	**98,64**	102,36	103,06
ais6	61	1,57%	**1,19**	1,25	**1,19**
ais8	113	0,57%	1,86	1,88	**1,74**
ais10	181	0,24%	2,38	2,40	**2,15**
ais12	265	0,10%	2,85	2,87	**2,50**
qg1-07	343	0,00%	**22,27**	23,46	23,69
qg1-08	512	0,00%	**27,96**	28,33	29,09
qg2-07	343	0,00%	**21,84**	22,81	22,71
qg2-08	512	0,00%	**26,93**	28,34	28,53
qg3-08	512	0,01%	**30,08**	30,77	32,27
qg3-09	729	0,00%	38,76	**38,05**	41,16
qg4-08	512	0,02%	34,49	41,56	**34,07**
qg4-09	729	0,01%	45,08	53,21	**43,97**
qg5-09	729	0,01%	**86,75**	91,07	87,78
qg5-10	1000	0,00%	**108,67**	111,06	109,40
qg5-11	1331	0,00%	**133,00**	138,96	135,01
qg5-12	1728	0,00%	160,64	162,52	**159,54**
qg5-13	2197	0,00%	**189,69**	196,17	189,85
qg6-09	729	0,05%	**69,96**	74,52	73,28
qg6-10	1000	0,02%	**90,04**	93,06	92,45
qg6-11	1331	0,01%	**108,42**	113,06	111,61
qg6-12	1728	0,00%	**132,13**	134,07	135,78
qg7-09	729	0,00%	**74,32**	76,96	77,73
qg7-10	1000	0,00%	**92,39**	96,42	95,95
qg7-11	1331	0,00%	**112,39**	117,34	116,39
qg7-12	1728	0,00%	**134,27**	139,56	138,85
qg7-13	2197	0,00%	**159,68**	165,91	163,85
bmc-ibm-1	9685	10,22%	**212,93**	226,78	236,29
bmc-ibm-2	2810	12,84%	**95,04**	97,83	109,27
bmc-ibm-3	14930	12,64%	**538,19**	559,67	577,21
bmc-ibm-4	28161	11,77%	**898,87**	942,16	969,38
bmc-ibm-5	9396	14,08%	**208,92**	216,21	270,44
bmc-ibm-6	51639	10,35%	**1 370,69**	1 448,51	1 498,72
bmc-ibm-7	8710	11,17%	**305,40**	323,16	332,70
bmc-galileo-8	58074	13,68%	1 392,82	**1 371,32**	2 638,74
bmc-galileo-9	63624	13,62%	**1 496,00**	1 843,45	2 554,83
bmc-ibm-10	59056	10,69%	**1 287,22**	1 494,22	1 395,31
bmc-ibm-11	32109	13,90%	**1 135,26**	1 181,41	1 232,00
bmc-ibm-12	39598	12,70%	**1 299,49**	1 354,53	1 371,22
bmc-ibm-13	13215	14,69%	**399,06**	419,25	446,89

TABLE 3.9 – Comparaison des politiques de prise en compte de la neutralité (paysages MAXSAT).

BEST+NP, etc.

Les résultats sur les instances MAXSAT basés sur des problèmes (à droite de la table) sont également très favorables à STOCH. Rappelons que nous avions conclu dans la section précédente que la stratégie du meilleur améliorant était globalement plus performante que la stratégie du premier améliorant (voir la table 3.4, à droite). Malgré cela, STOCH est plus performant que BEST+NP sur la plupart des instances malgré le fait qu'il utilise une règle pivot de type premier améliorant : STOCH obtient les meilleurs résultats moyens sur 55 des 68 paysages testés, et n'est battu statistiquement que sur 5 paysages, à chaque fois par BEST+NP.

L'efficacité de STOCH peut être encore plus mise en exergue si l'on se focalise sur les 10 paysages sw1000-8-*-c5-1. Ces paysages sont dérivés d'une formulation SAT de problèmes de coloration de graphes, chaque instance étant dépendante d'un paramètre de *morphing* qui détermine la part de l'aléatoire utilisée pour générer l'instance [Gent *et al.*, 1999]. Les graphes résultants sont construits en partie de manière aléatoire et en partie sous forme d'anneau régulier, chaque sommet étant connecté à ses 8 plus proches voisins. Les 10 instances testées sont triées par degrés croissants de structure, de sw100-8-lp0-c5, correspondant à une formulation SAT d'un graphe aléatoire, à sw100-8-p0-c5 correspondant à un anneau régulier. Dans [Gent *et al.*, 1999], les auteurs ont conclu qu'un haut niveau de structure dans de tels graphes induit des instances SAT qui sont beaucoup plus difficiles à résoudre pour les algorithmes de recherche locale. Cette observation est confirmée par les résultats des deux climbers classiques que sont FIRST+NP and BEST+NP, puisque le nombre de clauses non satisfaites tend à croître avec le niveau de structure des instances. Par contre, STOCH est efficace sur l'ensemble de ces instances et est particulièrement efficace sur les instances les plus structurées, où l'optimum global est quasiment systématiquement atteint : 97 fois sur 100 exécutions pour l'instance sw100-8-lp8-c5-1 et 98 fois sur 100 exécutions pour l'instance sw100-8-p0-c5-1. Cela contredit les affirmations de [Gent *et al.*, 1999], qui ne sont valables que pour les climbers FIRST+NP et BEST+NP très largement utilisés dans la littérature. Ce résultat nous conforte encore plus sur la capacité de STOCH à atteindre de bons optima locaux.

Concentrons-nous maintenant sur les paysages qg*, obtenus par formulation SAT d'instances de carrés latins [Zhang et Stickel, 2000]. L'estimation de la neutralité de ces paysages donne des résultats quasiment nuls, pourtant il apparait que la prise en considération des mouvements neutres améliore considérablement la qualité des optima locaux obtenus (voir table 3.4 vs table 3.9). Cette incohérence nous a amené à penser que la neutralité effectivement observée durant l'exécution des climbers est sous-estimée par l'indicateur ν, qui se base sur un échantillonnage de l'espace de recherche. D'autres expérimentations ont donc été effectuées afin de montrer qu'il existe effectivement une neutralité non négligeable lorsque la recherche atteint de bonnes configurations des paysages qg*, ce qui n'était pas observable par simple échantillonnage. Nous avons pu observer une évolution significative du taux de neutralité durant la recherche pour de nombreux paysages comme le montre la figure 3.5. De telles expérimentations ont été également effectuées pour la rugosité, mais sans observer de variation significative selon les paysages considérés.

3.3.5 Nombre d'évaluations

Comme pour l'évaluation des règles pivot, nous nous sommes focalisés ici sur la qualité des configurations atteintes par les climbers après un grand nombre d'évaluations. Nous avons observé que STOCH est globalement plus performant que FIRST+NP et BEST+NP en terme de qualité des configurations atteintes. Néanmoins, la vitesse de convergence des algorithmes est également un

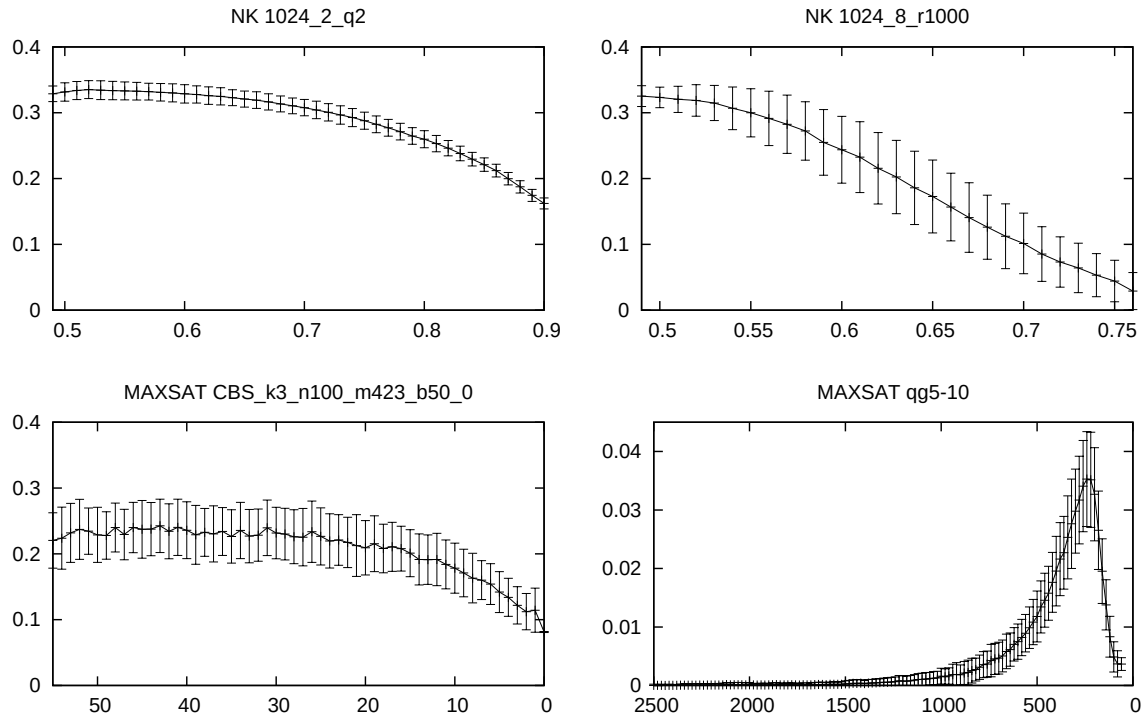


FIGURE 3.5 – Variation du taux estimé de neutralité en fonction des fitness des configurations : exemple pour 4 paysages de chaînes de bits représentatifs (NKq, NKr, 3-SAT aléatoire et MAX-SAT basée sur les quasi-groupes). Pour chaque paysage on a évalué le taux de neutralité des solutions parcourues durant l'exécution de 100 climbers stochastiques. Pour l'instance qg5-10, la neutralité n'apparaît réellement que dans le voisinage des configurations de bonne qualité.

aspect déterminant pour la comparaison des climbers, en particulier si l'on souhaite les intégrer comme composant d'une métaheuristique avancée, de type ILS par exemple. Nous avons donc cherché à déterminer si le fait d'accepter des mouvements neutres dès le début de la recherche augmentait le temps de convergence de `STOCH` par rapport aux autres variantes.

Nous avons donc calculé l'évolution moyenne des fitness atteints par les climbers tout au long de la recherche. Cette évolution est illustrée dans la figure 3.6, pour 6 instances représentatives de l'ensemble des paysages testés dans nos expérimentations. Il s'avère que non seulement la convergence de `STOCH` ne semble pas ralentie par l'acceptation des mouvements neutres, mais cela permet d'atteindre plus rapidement de bonnes configurations des paysages. Avec le climber `STOCH`, la probabilité d'accepter un voisin strictement améliorant est plus grande lorsque le nombre de voisins améliorants est important. Par conséquent `STOCH` va avoir tendance à s'orienter vers les zones du paysage contenant beaucoup de possibilités d'amélioration et permet donc ensuite d'améliorer plus facilement les solutions en plus de se diriger vers des bassins d'attractions potentiellement plus grands (amenant généralement à de bons optima locaux [Ochoa *et al.*, 2010]). Au contraire, `FIRST+NP` et `BEST+NP` cherchent systématiquement à améliorer la configuration courante, quitte à intensifier la recherche dans une zone où les possibilités d'amélioration sont restreintes et de rester ainsi dans un bassin d'attraction de petite taille.

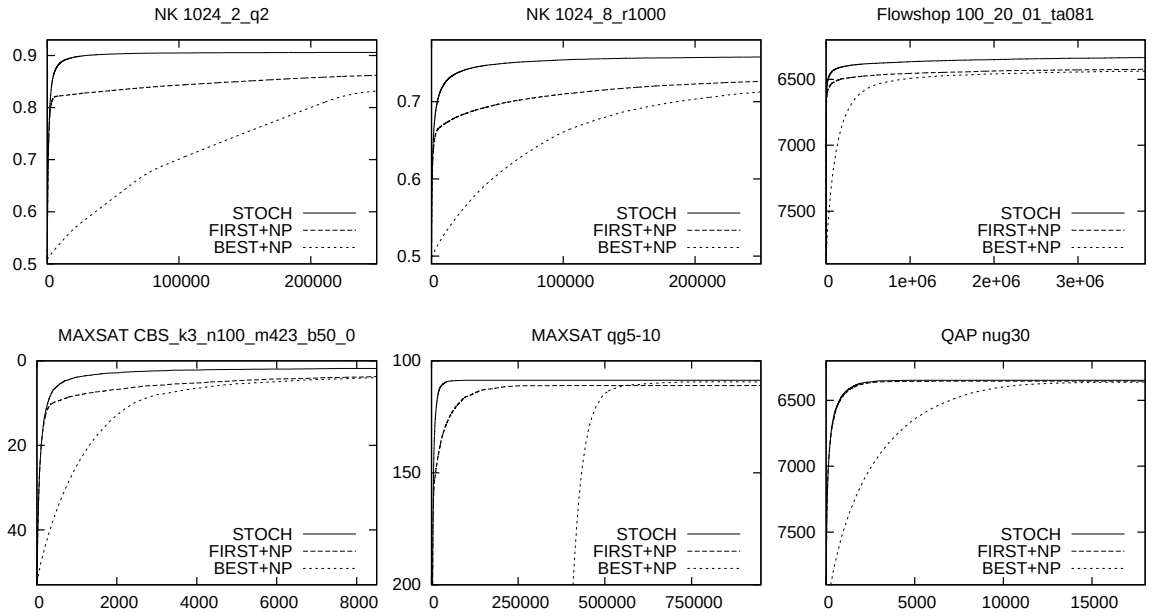


FIGURE 3.6 – Évolution moyenne du fitness durant l'exécution des climbers `STOCH`, `FIRST+NP` et `BEST+NP` (100 exécutions, en considérant le nombre de configurations évaluées).

3.3.6 Discussion

Dans cette section, nous nous sommes intéressés à la politique de prise en compte des mouvements neutres dans les climbers. Nous avons répertorié 3 approches classiques (`FIRST+NP`, `BEST+NP` et `STOCH`) et les avons éprouvés sur un large panel de paysages.

Nous avons mené des expérimentations sur des paysages dérivés de problèmes combinatoires classiques (Flow-shop, QAP, MAXSAT), ainsi que différents types de paysages avec neutralité, basés sur des modèles NK.

La prise en compte de la neutralité est très importante pour atteindre de bons optima locaux sur les paysages contenant de la neutralité. La performance de `FIRST+NP`, `BEST+NP` et `STOCH` est largement plus élevée que pour les climbers basiques `BEST` et `FIRST`. En se concentrant sur les différents moyens de prendre en compte la neutralité, il s'avère que les résultats obtenus sont clairement en faveur de `STOCH`. Ceci indique qu'il est préférable de considérer les mouvements neutres comme des mouvements améliorants tout au long de la recherche, alors que beaucoup de méthodes de la littérature cherchent à atteindre un optimum local non-strict avant de gérer les mouvements neutres afin de sortir du plateau atteint (`FIRST+NP`, `BEST+NP`). Cette conclusion est quasiment générale à l'ensemble des paysages testés contenant de la neutralité, même lorsque la stratégie du meilleure améliorant est meilleure que la stratégie du premier améliorant. Il existe néanmoins quelques exceptions sur les paysages MAXSAT basés sur des problèmes. Il serait intéressant d'étudier les causes d'un tel résultat, qui pourrait être dû par exemple au fait que certaines variables ont un poids très conséquent dans la fonction de fitness et doivent être traitées prioritairement pour s'orienter rapidement dans la bonne zone de l'espace de recherche ; dans ce cas la stratégie du meilleur améliorant semble particulièrement adaptée, ce qui rend `BEST+NP`

efficace. Pour généraliser les résultats obtenus dans les deux dernières sections (règles pivots et politiques de prise en compte des mouvements neutres), nous pouvons dire que dans la majorité des cas `STOCH` est à privilégier (si un paysage est non-neutre `STOCH` est équivalent à `FIRST`). On privilégiera `BEST+NP` uniquement lorsque le paysage est non-neutre ET très lisse ainsi que dans certains cas très spécifiques comme nous l'avons observé pour quelques instances MAXSAT.

Enfin, les expérimentations nous ont permis de mettre en exergue l'intérêt de créer artificiellement de la neutralité dans un paysage n'en contenant pas ou peu. La comparaison relative des résultats de `STOCH` sur les paysages NK et NKr a permis de montrer qu'ajouter raisonnablement de la neutralité aux paysages NK permet d'atteindre des optima locaux bien plus élevés. NKr consistant en une simple discrétisation de la fonction objectif, ce principe peut être étendu à tout problème d'optimisation contenant initialement peu de neutralité. Cependant, le paramètre permettant de contrôler le niveau de neutralité doit être fixé intelligemment afin qu'il y ait un réel apport en termes de convergence tout en ne perdant pas la recherche dans des plateaux gigantesques.

Au vu des conclusions obtenues grâce aux expérimentations menées dans ce chapitre et des résultats sur le taux de couverture de l'optimum global, nous nous sommes intéressés à la conception de climbers plus efficaces que ceux utilisant des stratégies classiques. Les différents travaux menés sont présentés dans le chapitre suivant.

Chapitre 4

Vers la conception de climbers performants

4.1 Pire améliorant

La plupart des climbers de la littérature sont intégrés dans des métaheuristiques avancées et la règle pivot utilisée est soit de type premier améliorant soit de type meilleur améliorant. De tels climbers sont simples à mettre en œuvre, mais ne sont clairement pas suffisants pour approximer correctement les problèmes d'optimisation. Par conséquent, la plupart des travaux se concentrent sur la mise au point de mécanismes de diversification efficaces permettant de relancer la recherche vers d'autres optima locaux. Pourtant, il parait probable que l'optimum global soit généralement atteignable par un climber à partir de la plupart des solutions initiales de l'espace de recherche, comme vu dans la section 2.2.3. Il parait donc très envisageable de mettre au point des règles pivots rendant les climbers plus performants et éventuellement de pouvoir se passer complètement de mécanisme de diversification de la recherche.

Dans cette section, nous nous intéressons donc principalement à la règle pivot des climbers. Nous avons montré précédemment que la règle pivot premier améliorant tend à être plus efficace que celle du meilleur améliorant dès lors que le paysage étudié est de grande taille ou rugueux. Intuitivement, choisir systématiquement le meilleur améliorant tend à conduire la recherche vers le pic le plus "proche" de la solution initiale dans le paysage de recherche, alors que choisir le premier améliorant réalise souvent des améliorations plus réduites sans nécessairement chercher à grimper directement vers le premier pic rencontré. D'après les résultats du chapitre précédent, ceci permet d'atteindre des zones plus hautes de l'espace de recherche qui mènent à de meilleurs optima locaux. Afin d'extrapoler ce résultat, il parait cohérent de chercher à réduire au maximum les améliorations réalisées afin de se déplacer doucement vers des zones du paysage plus prometteuses en évitant de grimper sur les petits pics du paysage. Par analogie avec le monde réel, cela consisterait à grimper doucement le long d'une vallée afin d'atteindre les montagnes les plus hautes et potentiellement éloignées plutôt que de grimper sur la première colline rencontrée (ceci est une illustration visuelle, les problèmes combinatoires sont bien plus complexes puisque les paysages que nous connaissons ne dépendent que de deux variables correspondant aux coordonnées de la position courante). Afin de réduire les améliorations de chaque pas d'un climber, nous nous sommes donc intéressés à la règle pivot consistant à choisir systématiquement la plus petite

amélioration possible parmi les voisins admissibles. Par opposition au meilleur améliorant, nous appellerons cette stratégie *pire améliorant* (WORST). Dans cette section, nous nous sommes donc intéressés à l'évaluation de WORST, ainsi qu'à d'autres règles pivot intermédiaires.

4.1.1 Du meilleur au pire améliorant

À ma connaissance, la stratégie du pire améliorant n'a jamais vraiment été envisagée et utilisée comme règle pivot d'un climber au sein d'une métaheuristique. Cette section se focalise principalement sur l'évaluation de stratégies favorisant la sélection de voisins peu améliorants et en particulier de la règle pivot du pire améliorant. Afin de perturber le moins possible les expérimentations, nous nous sommes focalisés sur les paysages NK basiques afin de ne pas devoir considérer de paysages neutres et de devoir tenir compte d'une éventuelle politique de prise en compte de la neutralité.

Soit $\mathcal{N}^{>}(x)$ l'ensemble des voisins améliorants d'une configuration x. Le fonctionnement de WORST est décrit dans l'algorithme 4.1.

Algorithme 4.1 Climber WORST

Entrée : une configuration initiale x
tant que x n'est pas un optimum local **faire**
  $x \leftarrow argmin_{x' \in \mathcal{N}^{>}(x)}(\mathfrak{f}(x'))$
Retourner x

Règles pivot basées sur le rang

Afin d'évaluer la pertinence de WORST, nous nous sommes intéressés d'une manière générale à l'impact du rang de la solution améliorante sélectionnée parmi l'ensemble des solutions améliorantes.

Soit $I_x = (x_1,x_2,\ldots,x_w)$ un tuple formé à partir des configurations de $\mathcal{N}^{>}(x)$ triées par ordre décroissant de fitness : $\forall i \in \{1,\ldots,w\}, x_i \in \mathcal{N}(x)$, $\mathfrak{f}(x_i) > \mathfrak{f}(x)$ et $\mathfrak{f}(x_i) \geqslant \mathfrak{f}(x_{i+1})$ $(i < w)$.

Étant donné une configuration x et son tuple associé $I_x = (x_1,x_2,\ldots,x_w)$, nous introduisons trois règles pivot génériques :

- $S_i^{=}$ sélectionne le $i^{\text{ème}}$ meilleur améliorant à chaque pas de la recherche (x_i).
- $S_i^{\geqslant}$ sélectionne aléatoirement l'un des i meilleurs voisins améliorants (cela correspond à choisir aléaroirement parmi $\{x_1,x_2,\ldots,x_i\}$).
- $S_i^{\leqslant}$ sélectionne aléatoirement l'un des $(w-i+1)$ pires voisins améliorants (choix aléatoire parmi $\{x_i,x_{i+1},\ldots,x_w\}$).

Soient $q_1 = \lceil w/4 \rceil$, $m = \lceil w/2 \rceil$ et $q_3 = \lceil 3w/4 \rceil$ les indices des premiers, seconds et troisièmes quartiles de I_x. $b = 1$ et w correspondent respectivement aux indices de la meilleure et de la pire configuration de I_x. Dans ce qui suit nous considérerons les combinaisons de $S_i^{=}$, $S_i^{\geqslant}$ et $S_i^{\leqslant}$ avec les cinq rangs $i \in \{b,q_1,m,q_3,w\}$. La figure 4.1 illustre les choix correspondant à ces stratégies et permet de visualiser que certaines stratégies sont équivalentes : $S_b^{=}$ et $S_b^{\geqslant}$, $S_b^{\leqslant}$ et $S_w^{\geqslant}$, $S_w^{=}$ et $S_w^{\leqslant}$. Notons également que, $S_b^{=}$ correspond à BEST, $S_w^{=}$ correspond à WORST et $S_w^{\geqslant}$ correspond à FIRST (bien que la définition suppose que le voisinage est systématiquement entièrement évalué).

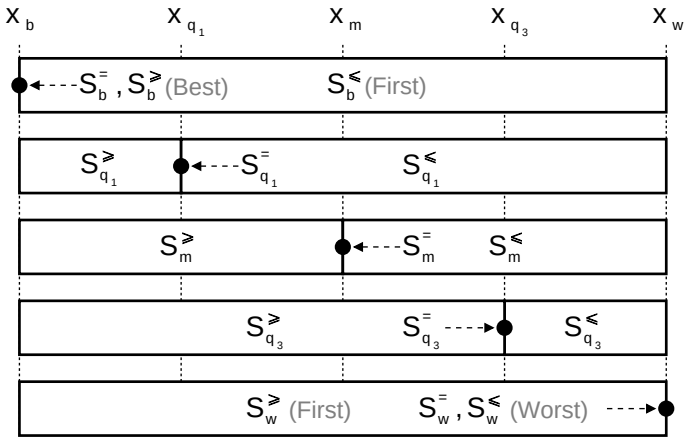


FIGURE 4.1 – Illustration des 12 règles pivot basées sur le rang. Les voisins améliorants sont triés du meilleur (x_b, à l'extrême gauche) au pire (x_w, à l'extrême droite). Ce classement détermine l'ensemble des voisins améliorants acceptables selon la règle pivot considérée. Par exemple, $S^{\geqslant}_{Q_3}$ consiste à sélectionner aléatoirement une solution améliorante parmi les 75% meilleures.

Munis des trois règles pivot génériques et des cinq indices introduits, nous obtenons donc 12 règles pivot distinctes, incluant `BEST`, `FIRST` et `WORST`. Nous nous sommes intéressés à l'efficacité relative de ces règles.

Expérimentations

Comme pour le chapitre précédent, nous avons évalué empiriquement les 12 règles pivot basées sur le rang qui viennent d'être introduites afin d'évaluer leur capacité à atteindre de bons optima locaux. Bien que nous focalisons notre étude sur le climber `WORST` principalement, les autres règles pivot nous permettront d'évaluer l'impact réel du rang des solutions sélectionnées sur la qualité globale des climbers.

Nous avons donc comparé 12 climbers, chacun correspondant à une règle pivot présentée dans la section 4.1.1 : $S^{=}_{b}$, $S^{=}_{\mathrm{q1}}$, $S^{=}_{m}$, $S^{=}_{\mathrm{q3}}$, $S^{=}_{w}$, $S^{\geqslant}_{\mathrm{q1}}$, $S^{\geqslant}_{m}$, $S^{\geqslant}_{\mathrm{q3}}$, $S^{\geqslant}_{w}$, $S^{\leqslant}_{\mathrm{q1}}$, $S^{\leqslant}_{m}$ et $S^{\leqslant}_{\mathrm{q3}}$. Afin de se concentrer sur des paysages sans neutralité tout en essayant de garder un panel suffisamment large d'instances, nous nous sommes concentrés sur 280 paysages NK (au lieu de 16 dans le chapitre précédent). Nous avons considéré 28 paramétrages (N,K), correspondants aux combinaisons de $N \in \{128,256,512,1024\}$ et $K \in \{1,2,4,6,8,10,12\}$. Pour chaque couple (N,K), nous avons généré aléatoirement et testé 10 paysages.

Le protocole expérimental utilisé est le même que pour le chapitre précédent (100 exécutions par instance, ensembles de configurations initiales identiques, arrêt lorsqu'un optimum local est atteint). Néanmoins, comme nous avons ici 10 instances par paramétrage (N,K), nous avons regroupé les résultats obtenus pour chaque ensemble d'instances, celles-ci ayant été générées de la même manière. Chaque valeur moyenne apparaissant dans la table 4.1 correspond donc à la moyenne des fitness obtenus pour 1000 exécutions d'un climber pour un paramétrage (N,K) (donc pour un groupe de 10 instances). Pour plus de lisibilité, seuls les résultats des climbers de type $S^{=}_{i}$ sont donnés, ainsi que les résultats de $S^{\geqslant}_{w}$ (`FIRST`) afin de donner un point de compa-

N_K	$S_b^=$	$S_{q1}^=$	$S_m^=$	$S_{q3}^=$	$S_w^=$	$S_w^\geqslant$
128_1	**0,6973**	0,6916	0,6877	0,6845	0,6809	0,6896
256_1	**0,7033**	0,6986	0,6966	0,6944	0,6918	0,6980
512_1	**0,6937**	0,6886	0,6856	0,6828	0,6803	0,6872
1024_1	**0,6997**	0,6954	0,6920	0,6895	0,6867	0,6936
128_2	**0,7169**	0,7142	0,7121	0,7107	0,7081	0,7133
256_2	**0,7130**	0,7104	0,7080	0,7055	0,7035	0,7080
512_2	**0,7120**	0,7097	0,7077	0,7063	0,7041	0,7076
1024_2	**0,7132**	0,7109	0,7092	0,7073	0,7052	0,7088
128_4	0,7216	0,7214	0,7231	0,7230	**0,7250**	0,7207
256_4	0,7211	0,7224	0,7237	0,7247	**0,7274**	0,7218
512_4	0,7228	0,7238	0,7255	0,7268	**0,7285**	0,7232
1024_4	0,7232	0,7244	0,7262	0,7275	**0,7298**	0,7238
128_6	0,7170	0,7178	0,7194	0,7217	**0,7251**	0,7187
256_6	0,7207	0,7217	0,7257	0,7269	**0,7313**	0,7234
512_6	0,7214	0,7232	0,7261	0,7291	**0,7338**	0,7239
1024_6	0,7223	0,7243	0,7276	0,7302	**0,7353**	0,7250
128_8	0,7124	0,7135	0,7150	0,7166	**0,7199**	0,7136
256_8	0,7147	0,7163	0,7201	0,7214	**0,7267**	0,7179
512_8	0,7163	0,7190	0,7220	0,7256	**0,7306**	0,7200
1024_8	0,7176	0,7206	0,7239	0,7274	**0,7330**	0,7215
128_10	0,7049	0,7056	0,7083	0,7106	**0,7115**	0,7078
256_10	0,7082	0,7095	0,7132	0,7158	**0,7197**	0,7126
512_10	0,7106	0,7128	0,7163	0,7195	**0,7244**	0,7143
1024_10	0,7121	0,7150	0,7185	0,7223	**0,7270**	0,7165
128_12	0,6970	0,6980	0,6996	0,7022	**0,7033**	0,7009
256_12	0,7015	0,7026	0,7062	0,7087	**0,7129**	0,7053
512_12	0,7045	0,7066	0,7104	0,7133	**0,7179**	0,7082
1024_12	0,7064	0,7089	0,7125	0,7163	**0,7209**	0,7107

N_K	$S_b^=$	$S_{q1}^=$	$S_m^=$	$S_{q3}^=$	$S_w^=$	$S_w^\geqslant$
128_1	**0,7103**	0,7065	0,7036	0,7022	0,6992	0,7061
256_1	**0,7127**	0,7108	0,7083	0,7058	0,7035	0,7092
512_1	**0,7011**	0,6970	0,6943	0,6917	0,6895	0,6962
1024_1	**0,7049**	0,7013	0,6981	0,6954	0,6930	0,6998
128_2	**0,7454**	0,7419	0,7418	0,7382	0,7386	0,7397
256_2	**0,7316**	0,7288	0,7273	0,7257	0,7233	0,7279
512_2	**0,7257**	0,7240	0,7219	0,7196	0,7182	0,7220
1024_2	**0,7233**	0,7199	0,7196	0,7175	0,7150	0,7192
128_4	0,7569	0,7569	0,7595	0,7578	**0,7604**	0,7544
256_4	0,7466	0,7502	0,7510	0,7507	**0,7544**	0,7477
512_4	0,7413	0,7435	0,7429	0,7448	**0,7470**	0,7425
1024_4	0,7363	0,7366	0,7389	0,7410	**0,7439**	0,7372
128_6	0,7569	0,7575	0,7579	0,7589	**0,7603**	0,7554
256_6	0,7449	0,7512	0,7524	0,7536	**0,7564**	0,7507
512_6	0,7425	0,7420	0,7443	0,7476	**0,7528**	0,7435
1024_6	0,7368	0,7388	0,7407	0,7446	**0,7478**	0,7394
128_8	0,7532	0,7521	**0,7562**	0,7533	0,7560	0,7504
256_8	0,7433	0,7428	0,7464	0,7483	**0,7524**	0,7427
512_8	0,7369	0,7409	0,7407	0,7442	**0,7486**	0,7410
1024_8	0,7308	0,7343	0,7371	0,7395	**0,7457**	0,7365
128_10	0,7432	0,7447	**0,7480**	0,7436	0,7471	0,7448
256_10	0,7356	0,7344	0,7380	0,7420	**0,7453**	0,7373
512_10	0,7317	0,7315	0,7348	0,7377	**0,7425**	0,7337
1024_10	0,7266	0,7307	0,7335	0,7348	**0,7401**	0,7312
128_12	0,7370	0,7350	0,7373	0,7389	**0,7400**	0,7392
256_12	0,7314	0,7280	0,7320	0,7348	**0,7375**	0,7300
512_12	0,7255	0,7261	0,7293	0,7321	**0,7360**	0,7271
1024_12	0,7228	0,7222	0,7264	0,7307	**0,7338**	0,7238

TABLE 4.1 – À gauche : comparaison des règles pivot de type $S_i^=$ (moyenne des fitness des 1000 optima locaux obtenus à partir de 10 instances). Rappel : $S_b^= \leftrightarrow$ `BEST`, $S_w^= \leftrightarrow$ `WORST` et $S_w^\geqslant \leftrightarrow$ `FIRST`. À droite : moyenne des meilleurs optima locaux obtenus pour chaque type d'instance (10 instances pour chaque paramétrage (N,K)).

raison. Les résultats des autres variantes n'apparaissent pas dans ce tableau, mais seront discutés ultérieurement.

Les résultats de la table 4.1 montrent clairement que `BEST` obtient les meilleurs résultats sur les paysages plutôt lisses ($K = \{1, 2\}$), tandis que `WORST` domine toutes les autres variantes sur les paysages plus rugueux ($K = \{4, 6, 8, 10, 12\}$). Ceci est vérifié clairement que ce soit au niveau des fitness moyens obtenus qu'au niveau des résultats statistiques. Nous avions observé la même tendance dans le chapitre précédent lorsque nous avions comparé `BEST` avec `FIRST`. Nous remarquons ici que les différences entre `WORST` et `BEST` sont encore plus marquées, à savoir qu'à chaque fois que `FIRST` est plus performant que `BEST`, alors `WORST` est encore plus performant et inversement.

D'une manière générale, il semble que la qualité des optima locaux atteints est directement liée au rang moyen des solutions sélectionnées parmi les voisins améliorants. Cette tendance est confirmée dans la figure 4.2, où l'on observe que l'évolution du fitness moyen est fortement corrélée au rang moyen des configurations sélectionnées. Cela implique que si l'on veut atteindre les meilleurs optima locaux possibles, il convient de choisir exclusivement entre `BEST` (paysages peu rugueux) et `WORST` (paysages rugueux).

La table 4.1 (partie droite) reporte les moyennes des meilleurs optima locaux obtenus par chaque règle pivot sur chaque paysage. Pour chaque couple (stratégie, paysage), le meilleur opti-

mum local atteint sur l'ensemble des 100 exécutions est calculé. Les valeurs données dans la table correspondent au fitness moyen des meilleurs optima locaux obtenus par chaque stratégie sur les 10 instances de chaque paramétrage (N,K). Les observations faites à partir des moyennes restent valables. Cela nous permet de penser que la supériorité de `WORST` pourrait être également valable dans un contexte de recherche locale itérée, voire même pour tout type de métaheuristique avancée utilisant un climber (en supposant que l'on exécute un nombre fixé d'exécutions du climber, sans considération de temps d'exécution).

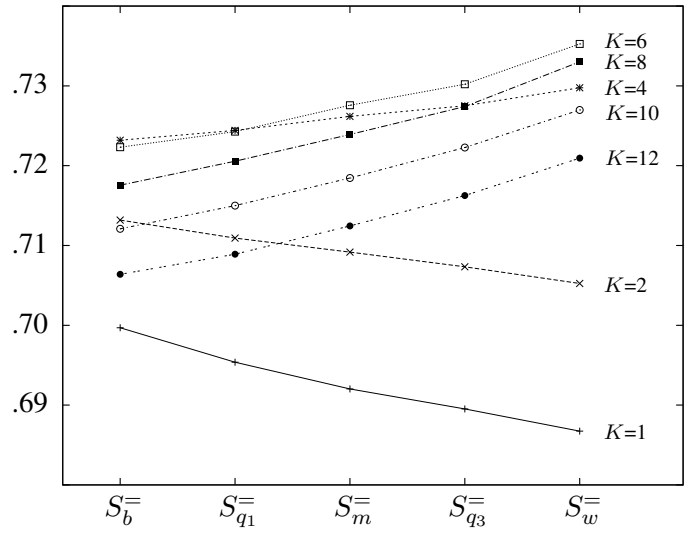


FIGURE 4.2 – Résultats des règles pivot de type $S_i^=$ en fonction de la rugosité des paysages considérés, pour $N = 1024$. La tendance est similaire pour les autres valeurs de N (voir la table 4.1).

Les valeurs moyennes des fitness atteints par les 12 règles pivot pour $N = 1024$ apparaissent dans le graphique de la figure 4.3. Les règles pivot sont triées selon le rang moyen des solutions sélectionnées du plus petit (à gauche) au plus grand (à droite). Pour chaque niveau de rugosité (valeurs de K), on peut observer une évolution quasi monotone des fitness moyens atteints. Cela confirme que l'efficacité des climbers est directement corrélée (positivement ou négativement) avec la quantité d'amélioration réalisée à chaque pas de la recherche. La table apparaissant dans la figure 4.3 donne les valeurs numériques pour $K = 12$, et permet d'observer la corrélation rang/fitness atteint, pour les trois règles pivot génériques ($S^=$, $S^\geqslant$ et $S^\leqslant$).

Les différents résultats obtenus lors des expérimentations (table 4.1 et figure 4.3) montrent qu'il existe un intérêt certain à utiliser la stratégie du pire améliorant pour atteindre de bons optima locaux sur les paysages rugueux. Cependant, il est aussi évident que cela implique d'effectuer beaucoup plus d'évaluations de configurations que les stratégies classiques comme `BEST` et `FIRST`. En effet, `BEST` permet d'effectuer peu de pas puisque les améliorations sont toujours maximales, alors que `WORST` en réalise nécessairement un grand nombre car les améliorations réalisées sont souvent minimes. `FIRST` ne nécessite pas de générer l'ensemble du voisinage à chaque étape de la recherche, ce qui réduit considérablement le nombre d'évaluations effectuées pour atteindre un optima local. En considérant comme référence le nombre moyen d'évaluations nécessaires à `FIRST` pour atteindre un optimum local, `BEST` nécessite presque 10 fois plus d'évaluations et `WORST` nécessite plus de 200 fois plus d'évaluations. C'est pourquoi nous nous inté-

Stratégie	b	q_1	m	q_3	w
$S^{=}$	0,7064	0,7089	0,7125	0,7163	**0,7209**
$S^{\geqslant}$	0,7064	0,7072	0,7083	0,7094	0,7107
$S^{\leqslant}$	0,7107	0,7133	0,7158	0,7185	0,7209

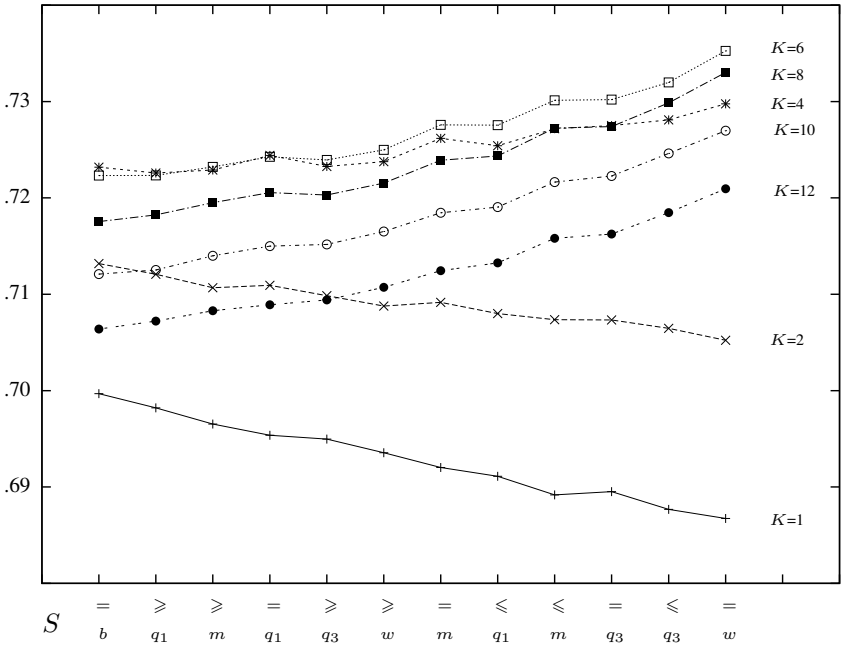


FIGURE 4.3 – Performance moyenne des 12 règles pivot sur les paysages NK (N=1024), groupés en 7 ensembles par niveaux de rugosité (k=1, 2, 4, 6, 8, 10 ou 12). Les stratégies sont triées selon le rang moyen $\frac{\text{rang\_moyen\_du\_voisin\_sélectionné}-1}{\text{nombre\_voisins\_améliorants}-1}$ (de $S_B^{=}$ à $S_W^{=}$: 0, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{1}{2}$, $\frac{5}{8}$, $\frac{3}{4}$, $\frac{3}{4}$, $\frac{7}{8}$, 1). La table (en haut) donne les valeurs numériques pour K=12.

ressons au nombre d'évaluations dans la section suivante, en proposant une approche pour tenter de garder les bénéfices obtenus par la minimisation des améliorations durant la recherche, tout en réalisant un nombre réduit d'évaluations de solutions.

4.1.2 Approximation du pire améliorant

Les expérimentations réalisées pour les règles pivot basées sur le rang des solutions améliorantes ont montré que pour qu'un climber atteigne de bons optima locaux d'un paysage rugueux, il est préférable de sélectionner les mouvements les moins améliorants possible. Cependant, cela est très coûteux en temps de calcul, d'une part car cela tend à augmenter le nombre de pas nécessaires pour atteindre un optimum local, d'autre part car sélectionner le pire améliorant nécessite d'évaluer la totalité du voisinage pour chaque mouvement.

Nous nous proposons donc d'approximer `WORST` en ne générant qu'une partie du voisinage à chaque pas de la recherche, la solution améliorante sélectionnée étant alors celle qui améliore le moins possible le fitness. Un tel mécanisme est couramment utilisé pour approximer la stratégie

du meilleur améliorant sur de larges voisinages [Mladenović et Hansen, 1997].

Nous introduisons donc la stratégie *pire améliorant partiel* ($\texttt{WORST}_\kappa$), consistant à stopper la génération du voisinage dès que κ voisins améliorants ont été trouvés, générant la totalité du voisinage s'il contient moins de κ améliorations possibles. À chaque étape de la recherche, $\texttt{WORST}_\kappa$ sélectionne le moins bon améliorant parmi les κ (ou moins) voisins améliorants générés. Cela induit que le paramètre κ affecte directement le nombre de solutions évaluées pour réaliser un pas de recherche locale, mais affecte aussi la qualité de l'approximation du pire améliorant. Notons que $\texttt{WORST}_1$ correspond à la stratégie du premier améliorant, tandis que $\texttt{WORST}_N$ correspond à la stratégie du pire améliorant.

N_K	BEST	FIRST	WORST partiel				WORST
	-	$\texttt{WORST}_1$	$\texttt{WORST}_2$	$\texttt{WORST}_4$	$\texttt{WORST}_8$	$\texttt{WORST}_{16}$	$\texttt{WORST}_N$
128_4	0,7216	0,7207	0,7224	0,7237	0,7234	0,7243	**0,7250**
	5k	1k	2k	4k	8k	16k	31k
256_4	0,7211	0,7218	0,7233	0,7254	0,7257	0,7262	**0,7274**
	19k	2k	5k	10k	21k	41k	136k
512_4	0,7228	0,7232	0,7248	0,7262	0,7278	0,7281	**0,7285**
	76k	5k	11k	23k	49k	97k	568k
1024_4	0,7232	0,7238	0,7253	0,7270	0,7286	0,7291	**0,7298**
	302k	12k	25k	54k	112k	223k	2336k
128_6	0,7170	0,7187	0,7214	0,7228	0,7242	0,7240	**0,7251**
	4k	1k	2k	5k	10k	21k	46k
256_6	0,7207	0,7234	0,7264	0,7285	0,7301	0,7309	**0,7313**
	15k	2k	5k	12k	26k	56k	221k
512_6	0,7214	0,7239	0,7272	0,7298	0,7315	0,7328	**0,7338**
	63k	5k	12k	28k	63k	135k	979k
1024_6	0,7223	0,7250	0,7280	0,7310	0,7330	0,7343	**0,7353**
	251k	13k	29k	65k	145k	311k	4151k
128_8	0,7124	0,7136	0,7168	0,7194	0,7197	0,7192	**0,7199**
	3k	1k	2k	5k	11k	24k	56k
256_8	0,7147	0,7179	0,7218	0,7243	0,7259	**0,7267**	**0,7267**
	13k	2k	5k	13k	30k	66k	284k
512_8	0,7163	0,7200	0,7242	0,7266	0,7285	0,7297	**0,7306**
	53k	5k	13k	31k	73k	163k	1324k
1024_8	0,7176	0,7215	0,7251	0,7285	0,7306	0,7316	**0,7330**
	214k	13k	31k	74k	172k	382k	5837k
128_10	0,7049	0,7078	0,7103	0,7122	0,7116	**0,7127**	0,7115
	3k	1k	2k	5k	11k	24k	59k
256_10	0,7082	0,7126	0,7159	0,7176	0,7196	**0,7201**	0,7198
	11k	2k	5k	13k	32k	71k	325k
512_10	0,7106	0,7143	0,7190	0,7212	0,7232	0,7237	**0,7243**
	46k	5k	13k	33k	81k	183k	1593k
1024_10	0,7121	0,7165	0,7204	0,7235	0,7255	0,7265	**0,7270**
	187k	14k	33k	80k	189k	439k	7325k
128_12	0,6970	0,7009	0,7021	0,7035	**0,7037**	0,7028	0,7033
	2k	1k	2k	5k	11k	24k	59k
256_12	0,7015	0,7053	0,7089	0,7105	0,7124	0,7122	**0,7129**
	10k	2k	5k	13k	33k	74k	346k
512_12	0,7045	0,7082	0,7127	0,7151	0,7168	**0,7179**	**0,7179**
	41k	5k	13k	34k	85k	199k	1816k
1024_12	0,7064	0,7107	0,7150	0,7178	0,7197	0,7206	**0,7210**
	166k	14k	34k	84k	206k	487k	8544k

TABLE 4.2 – Comparaison de BEST, FIRST et WORST avec $\texttt{WORST}_\kappa$ (fitness moyens des optima locaux atteints sur 1000 exécutions des climbers distribuées sur 10 instances). Sur la partie inférieure de chaque cellule se trouve le nombre moyen d'évaluations réalisées (en milliers).

Nous avons réalisé des expérimentations pour différentes valeurs de κ correspondant aux règles pivot $\texttt{WORST}_2$, $\texttt{WORST}_4$, $\texttt{WORST}_8$, $\texttt{WORST}_{16}$, mais également $\texttt{WORST}_1$ et $\texttt{WORST}_N$ afin de pouvoir

comparer les résultats avec respectivement FIRST et WORST. Le protocole expérimental établi est le même que celui proposé dans la section précédente, nous permettant de comparer les 6 règles pivots sur les paysages NK. Étant donné que WORST est préférable à BEST uniquement sur les instances rugueuses ($K \geqslant 4$), nous nous sommes restreints à ces paysages, soit 200 au total.

Les moyennes obtenues, les résultats statistiques, mais aussi le nombre d'évaluations réalisées se trouvent dans la table 4.2. Rappelons que $\texttt{WORST}_1$ et $\texttt{WORST}_N$ étaient nommés respectivement $S_w^=$ et $S_w^\geqslant$ dans la table 4.1, ce qui fait que les résultats de ces méthodes correspondent dans les deux tables (par exemple $\texttt{WORST}_N$ domine statistiquement $\texttt{WORST}_1$ lorsque $K \geqslant 4$).

En observant la table ainsi que la figure 4.4 qui résume graphiquement les résultats, on remarque que $\texttt{WORST}_N \succ \texttt{WORST}_{16} \succ \texttt{WORST}_8 \succ \texttt{WORST}_4 \succ \texttt{WORST}_2 \succ \texttt{WORST}_1$, si l'on considère que $A \succ B$ indique qu'un climber A est meilleur en moyenne qu'un climber B sur un paramétrage particulier des paysages NK. Cela confirme la tendance générale observée dans la section précédente, à savoir qu'à chaque fois que FIRST est meilleur que BEST, l'efficacité globale du climber est inversement proportionnelle à la qualité des voisins améliorants sélectionnés. On observe également que même en définissant une valeur relativement faible pour κ, on augmente sensiblement la performance du climber relativement à une stratégie du premier améliorant et on arrive assez souvent à obtenir des résultats qui ne sont pas statistiquement dominés par la stratégie du pire améliorant.

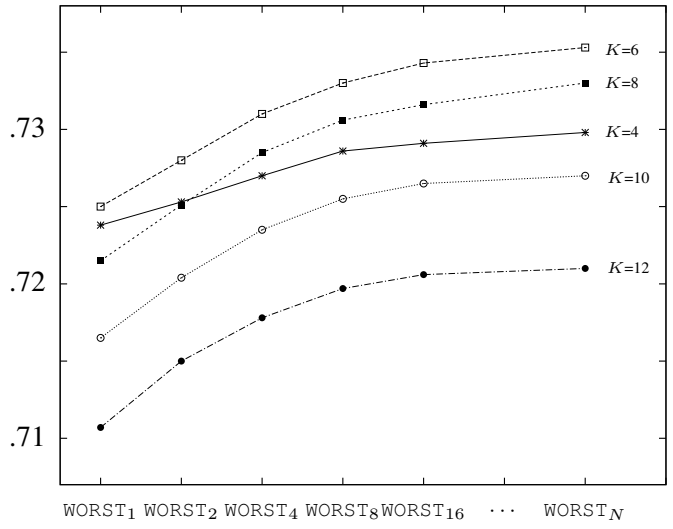


FIGURE 4.4 – Résultats des règles pivot $\texttt{WORST}_\kappa$, $\kappa \in \{1,2,4,8,16,1024\}$) sur les paysages rugueux de dimension $N = 1024$ ($K \in \{4,6,8,10,12\}$).

L'aspect particulièrement intéressant de ces résultats est que $\texttt{WORST}_\kappa$ nécessite un nombre réduit d'évaluations, surtout lorsque κ est petit. Très souvent, on arrive à atteindre des résultats statistiquement non dominés par WORST avec des valeurs réduites de κ et un nombre d'évaluation beaucoup plus petit (voir table 4.2). La détermination de κ consiste donc à trouver un compromis entre avoir un climber efficace mais nécessitant un temps d'exécution peu élevé.

Sur la figure 4.5, nous pouvons observer l'évolution moyenne du fitness en fonction du nombre d'évaluations réalisées sur un paysage 1024_6. Cela permet de visualiser le fait que le WORST est

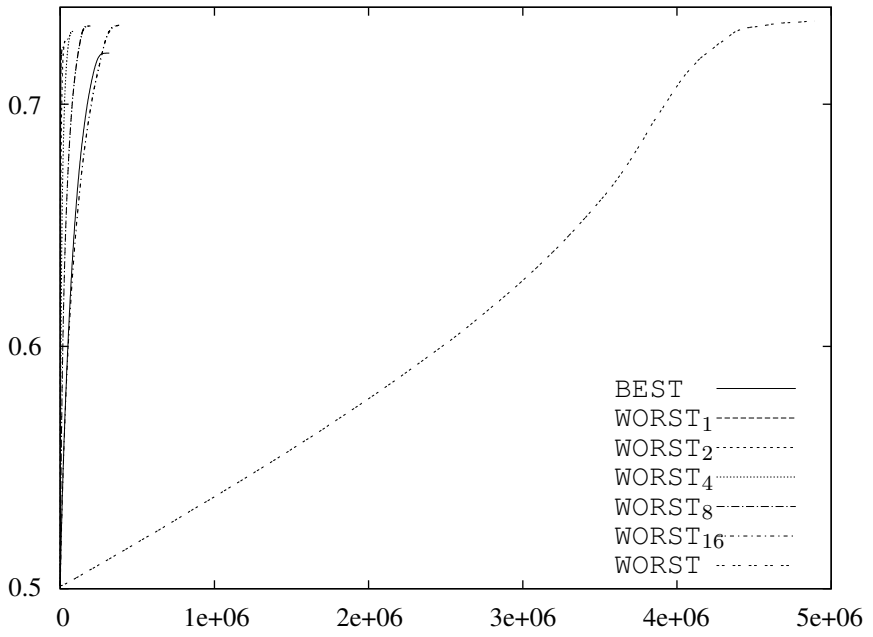


FIGURE 4.5 – Évolution de la convergence de BEST, WORST, WORST_1 (FIRST), WORST_2, WORST_4, WORST_8, WORST_{16} et WORST_N sur un paysage 1024_6 (abscisses : nombre d'évaluations ; ordonnées : fitness moyen).

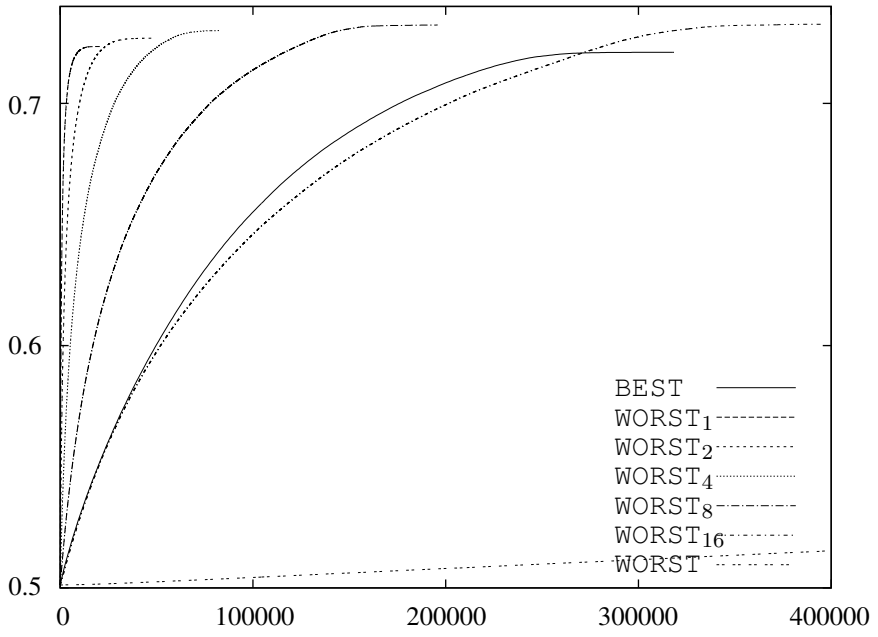


FIGURE 4.6 – Évolution de la convergence de BEST, WORST, WORST_1 (FIRST), WORST_2, WORST_4, WORST_8, WORST_{16} et WORST_N (zoom sur les 400000 premières évaluations de la figure 4.5).

largement plus long à converger que les autres stratégies, même pour $\texttt{WORST}_{16}$. Il est tout de même intéressant de noter que bien que WORST soit très coûteux en nombre d'évaluations, la convergence tend à s'accélérer durant la recherche [1]. Cette accélération de la convergence durant la recherche a été observée sur l'ensemble des instances, ce qui est une particularité assez remarquable en comparaison au comportement de la plupart des recherches locales.

La figure 4.6 reprend les caractéristiques de la figure 4.5, mais en se focalisant sur les 400000 premières évaluations qui sont suffisantes pour terminer les autres règles pivot. Cette figure nous permet de visualiser dans un premier temps la nette dominance des règles pivot de type $\texttt{WORST}_\kappa$ par rapport à BEST. De plus, le compromis entre vitesse de convergence et qualité de l'optimum atteint apparait clairement. Accroître $kappa$ permet d'atteindre des meilleurs optima locaux tout en augmentant le coût de la recherche. Lorsque $\kappa = 16$, la vitesse de convergence est proche de BEST, sauf que la recherche est piégée plus tard dans un optimum local qui est souvent de meilleure qualité.

4.1.3 Discussion

Suite à nos observations du chapitre précédent, nous avons proposé la stratégie du pire améliorant (WORST), qui permet de réduire l'amplitude des améliorations successives du climber afin d'atteindre doucement une zone du paysage contenant des optima locaux de bonne qualité. Nous avons montré cette capacité au travers d'expérimentations qui ont montré que sur les paysages difficiles WORST est plus performant que FIRST, sachant que FIRST est lui-même plus performant que BEST sur ces mêmes instances. D'une manière générale, nous avons montré que la qualité des optima locaux atteints par un climber est négativement corrélée avec la qualité des voisins améliorants sélectionnés (pour les paysages rugueux).

Nous avons également montré que même si WORST est très gourmand en ressource de temps, il est possible d'utiliser une règle pivot approximative ($\texttt{WORST}_\kappa$), qui permet d'obtenir des résultats très proches mais en un temps beaucoup plus réduit. De plus, le paramètre κ permet de pouvoir contrôler le compromis entre qualité et coût de la recherche.

Bien évidemment, il conviendra par la suite de confirmer l'efficacité de WORST sur d'autres types de paysages, associés à des problèmes d'optimisation combinatoires académiques ou réels. Il sera intéressant également d'éprouver cette règle pivot en la combinant avec STOCH pour traiter les paysages contenant de la neutralité. Enfin, l'objectif majeur est d'évaluer la capacité de $\texttt{WORST}_\kappa$ à être performant lorsqu'il est intégré à des métaheuristiques avancées.

Nous avons obtenu des premiers résultats très encourageants allant dans ce sens. Des expérimentations ont été effectuées sur le flow shop et montrent qu'un climber combinant $\texttt{WORST}_\kappa$ et STOCH permet de dépasser les résultats obtenus par STOCH. D'autre part, $\texttt{WORST}_\kappa$ a été intégré dans une métaheuristique avancée proposée pour résoudre le problème de partitionnement de cliques dans les graphes. La métaheuristique initiale était très performante puisque qu'elle a permis de dépasser des meilleures solutions connues de la littérature. L'intégration de $\texttt{WORST}_\kappa$ pour remplacer la règle pivot initiale a permis d'améliorer encore plus ces résultats. Ces travaux sont en cours et n'ont pas encore été publiés. Mais il nous semble très important d'appliquer le principe de $\texttt{WORST}_\kappa$ à un large panel de problèmes et de métaheuristiques avancées.

1. Pour l'ensemble des exécutions de WORST, la convergence s'accélère durant la recherche et en particulier lors des derniers pas. Sur la figure 4.5, la convergence de WORST semble ralentir après 4.10^6 évaluations. Ceci est dû au fait que les 100 exécutions ne se terminent pas simultanément, mais dans un intervalle situé entre 4.10^6 and 5.10^6 évaluations.

4.2 Neutralité artificielle par discrétisation de la fonction de fitness

Dans la section 3.3, nous avons vu que l'ajout artificiel de neutralité dans les paysages de recherche en arrondissant la fonction de fitness peut mener les climbers à atteindre de bien meilleurs optima locaux qu'en se concentrant sur la fonction de fitness originelle (particulièrement pour le climber `STOCH`). Cet apport n'est pas systématiquement effectif, la neutralité apportée devant être ni trop basse ni trop haute (voir figure 4.7).

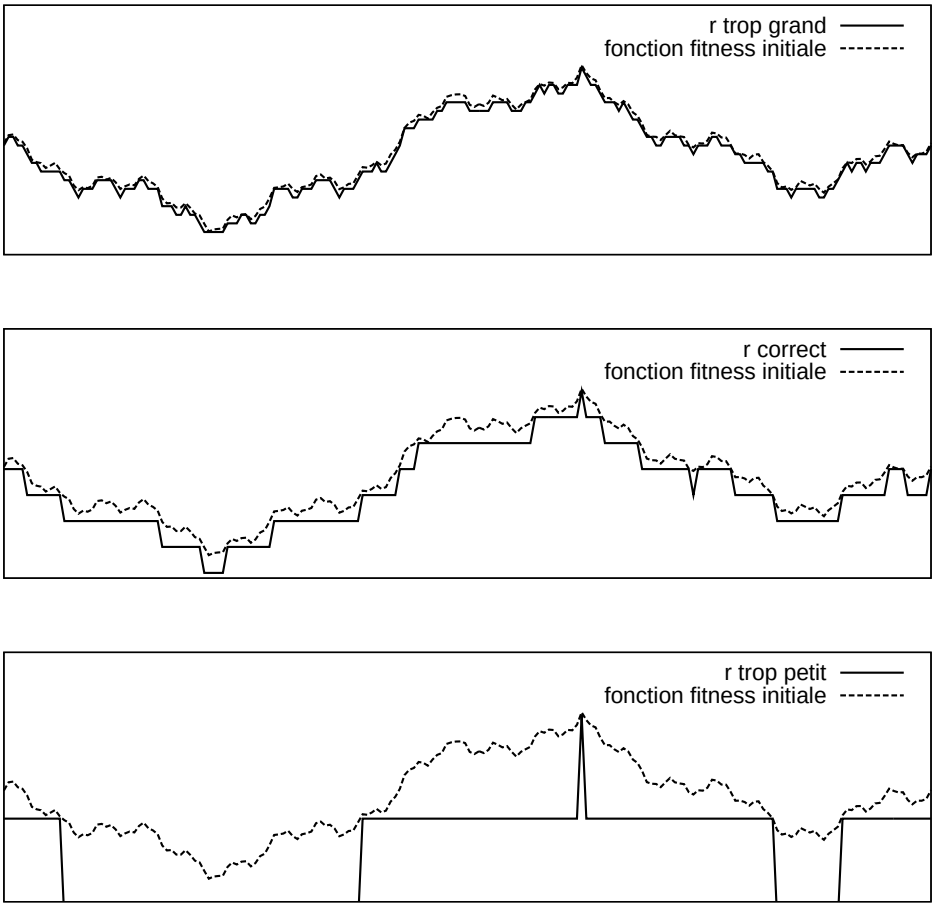


FIGURE 4.7 – Illustration de la modification d'un paysage par arrondi de la fonction de fitness : peu d'arrondi aboutit à peu de changement sur le comportement des climbers (en haut), un arrondi approprié peut faciliter la recherche de l'optimum global (au milieu) et un arrondi trop fort peut perdre la recherche dans des plateaux trop grands (en bas).

Durant le stage de Master recherche d'Hugo Traverson réalisé au premier semestre 2014, nous nous sommes penchés plus particulièrement sur cet aspect afin d'essayer de mettre au point un climber capable d'atteindre directement de bons optima locaux, sans avoir à paramétrer a priori la fonction d'arrondi du fitness des configurations. Dans ce travail, nous nous sommes concentrés sur les paysages NK, bien que tout soit facilement adaptable à d'autres types de problèmes combinatoires, à partir du moment où ceux-ci possèdent un taux de neutralité naturelle relativement

faible. L'objectif est donc d'exploiter une neutralité artificielle pour permettre au climber STOCH d'atteindre de bons optima locaux. Dans ce qui suit, le climber stochastique appliqué à un paysage NK aux fitness arrondis sera nommé r−STOCH.

4.2.1 Neutralité par arrondi fixe du fitness

Dans les résultats de la section 3.3, nous avions observé que trop peu d'arrondi changeait peu les résultats des climbers, tandis qu'un arrondi trop important aboutissait à des recherches qui se perdent dans d'immenses plateaux, incapables de gravir les marches trop rares du paysage modifié. Lorsque l'arrondi est bien dosé entre ces deux extrêmes, cela permet d'éviter beaucoup d'optima locaux de faible qualité en acceptant parfois des mouvements à l'origine légèrement détériorants, tout en gardant la structure du paysage de départ et ainsi pouvoir le parcourir efficacement.

Il nous a donc semblé naturel de penser que la courbe du fitness moyen atteint par r−STOCH en fonction de l'arrondi effectué devrait avoir une forme plutôt parabolique. Nous nous sommes donc intéressés à identifier l'intervalle d'arrondis pour lequel r−STOCH est le plus efficace.

La première étape a été d'évaluer la performance de r−STOCH pour différentes valeurs d'arrondis, afin de vérifier notre intuition, mais aussi de déterminer les valeurs d'arrondi optimales pour chaque paysage NK. Afin d'éviter d'effectuer un nombre incalculable d'expérimentations, nous avons cherché à déterminer les meilleures valeurs d'arrondi par dichotomie, en conservant à chaque fois les arrondis dont les résultats associés ne sont pas dominés statistiquement par ceux obtenus par d'autres valeurs d'arrondi. La méthode est décrite plus précisément dans le rapport de Master d'Hugo Traverson [Traverson, 2014].

N_K	STOCH	r−STOCH	r non dominés
128_1	0,7021	**0,7227**	$[206 \cdots 285]$
256_1	0,7021	**0,7206**	$[321 \cdots 350]$
512_1	0,6897	**0,7069**	$[731 \cdots 824]$
1024_1	0,6969	**0,7157**	$[1371 \cdots 1562]$
128_2	0,7021	**0,7395**	$[94 \cdots 148]$
256_2	0,7066	**0,7410**	$[212 \cdots 264]$
512_2	0,7135	**0,7484**	$[454 \cdots 523]$
1024_2	0,7146	**0,7507**	$[959 \cdots 1178]$

N_K	STOCH	r−STOCH	r non dominés
128_4	0,7254	**0,7837**	$[54 \cdots 81]$
256_4	0,7235	**0,7843**	$[117 \cdots 152]$
512_4	0,7200	**0,7790**	$[234 \cdots 304]$
1024_4	0,7246	**0,7844**	$[449 \cdots 570]$
128_8	0,7142	**0,7685**	$[40 \cdots 64]$
256_8	0,7166	**0,7755**	$[84 \cdots 112]$
512_8	0,7206	**0,7810**	$[173 \cdots 206]$
1024_8	0,7216	**0,7836**	$[331 \cdots 403]$

TABLE 4.3 – Comparaison de STOCH appliqué sur le paysage original par rapport à STOCH appliqué sur un paysage aux fitness arrondis (r−STOCH). Les valeurs reportées pour r−STOCH sont les moins bonnes en moyenne de l'intervalle des arrondis non dominés. La dernière colonne donne l'intervalle des arrondis non dominés correspondant aux valeurs d'arrondis pour lesquelles les fitness moyens atteints ne sont pas dominés statistiquement par ceux obtenus par une autre valeur d'arrondi.

La table 4.3 résume les résultats obtenus sur les paysages NK. La remarque principale que l'on peut formuler est que les résultats de r−STOCH sont nettement supérieurs à ceux obtenus par STOCH, quel que soit le paysage considéré. Bien sûr, on se concentre ici sur les meilleures valeurs de r et le fait d'accepter implicitement des mouvements détériorants permet à la recherche d'être beaucoup plus performante. Cependant, l'élégance et la simplicité d'une telle approche est à souligner, surtout qu'elle est très compétitive avec les méthodes de recherches locales itérées (voir

plus loin).

Une deuxième observation est que l'apport de l'arrondi est surtout dépendant de la rugosité K. En effet, l'écart entre `STOCH` et `r-STOCH` selon N varie de 0,0172 à 0,0206 pour $K = 1$, de 0,0344 à 0,0374 pour $K = 2$, de 0,0583 à 0,0608 pour $K = 4$ et de 0,0543 à 0,0620 pour $K = 8$.

Pour chaque valeur d'arrondi testée, nous avons évalué le taux de neutralité du paysage résultant par échantillonnage de l'espace de recherche, comme décrit dans la section 3.3. Le rapport entre le taux de neutralité du paysage et la performance de `r-STOCH` se trouve sur la figure 4.8. On observe que comme nous le pensions intuitivement, les courbes sont toutes de forme plutôt parabolique. Mais il est intéressant d'observer que les différentes courbes atteignent leur maximum toujours dans la même zone de taux de neutralité. La figure 4.9 se focalise sur les taux de neutralités de 10% à 40% et permet de bien observer ce phénomène. Globalement, l'ensemble des arrondis non dominés correspond toujours à un taux de neutralité observé se trouvant entre 19% et 32% (zones épaisses de la figure 4.9). Ce résultat nous a poussé à mettre au point une méthode permettant de gérer de manière adaptative la valeur de r afin d'approcher au maximum un taux de neutralité objectif fixé. Ceci est l'objet de la section suivante.

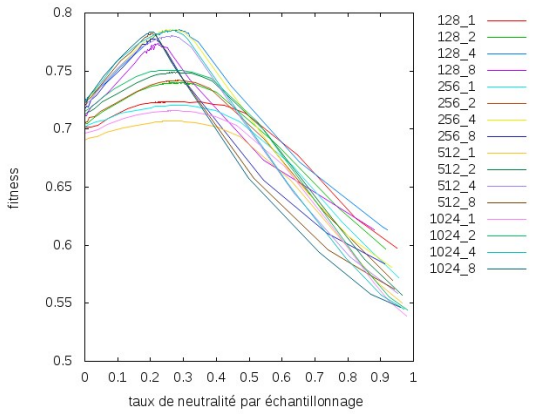


FIGURE 4.8 – Fitness moyen atteint par `r-STOCH` en fonction du taux de neutralité observé pour les différentes valeurs d'arrondi.

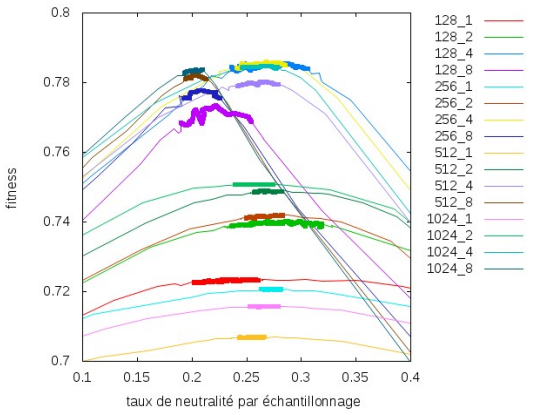


FIGURE 4.9 – Zoom de la figure 4.8 sur l'intervalle de taux de neutralité le plus efficace pour `r-STOCH` (en épais : intervalles non dominés) .

4.2.2 Neutralité par arrondi adaptatif du fitness

Nous avons observé que le climber `STOCH` est particulièrement efficace lorsque l'on créé artificiellement un taux de neutralité de l'ordre de 25% sur les paysages NK, qui ne possèdent pas de neutralité naturelle. Il peut être fastidieux de réaliser une étude de sensibilité au paramètre d'arrondi du fitness pour chaque paysage à résoudre. Pour cela, nous nous sommes intéressés au paramétrage autonome de l'arrondi afin de garder une recherche autonome, sans paramètre à fixer. L'idée générale est de calculer dynamiquement le taux de voisins neutres rencontrés durant la recherche, et d'augmenter l'arrondi lorsqu'il y a trop de neutralité ou de le diminuer lorsqu'il n'y a pas assez de neutralité par rapport au taux de neutralité souhaité.

On pourrait souhaiter viser un taux de neutralité de 25%, mais comme nous l'avons vu dans la figure 3.5, le taux de neutralité observé durant la recherche tend à baisser sensiblement lorsque l'on atteint des hautes zones des paysages NK. Par conséquent, un taux de neutralité de 25% par

échantillonnage correspond en pratique à un taux plus faible observé pendant la recherche. Nous avons donc testé plusieurs valeurs de neutralité objectif (entre 5% et 50%, par pas de 5) afin de savoir quel est le taux le plus efficace. Des détails concernant la méthodologie utilisée se trouvent dans [Traverson, 2014].

La table 4.5 présente les résultats obtenus. La colonne r−STOCH contient le meilleur fitness moyen obtenu par l'ensemble des arrondis fixes testés et la colonne ν−STOCH contient le meilleur fitness moyen obtenu par le taux de neutralité objectif le plus efficace. On voit que pour la plupart des instances, ν−STOCH est capable de régler efficacement l'arrondi et même d'être plus performant que r−STOCH, qui nécessite d'étudier de nombreuses valeurs de r. La dernière colonne indique le taux de neutralité objectif le plus efficace. On remarque qu'il varie peu : d'une manière générale, si l'on fixe un taux de neutralité objectif de 10% les résultats obtenus sont très satisfaisants.

En conclusion, ν−STOCH permet d'une part de régler automatiquement la neutralité durant la recherche, mais permet aussi de s'adapter aux changements d'allure du paysage, en modifiant r constamment afin de toujours conserver une proportion de voisins neutres acceptable.

N_K	r−STOCH	ν−STOCH	meilleur ν
128_1	0,7235	**0,7245**	15%
256_1	0,7213	**0,7217**	15%
512_1	0,7072	**0,7077**	15%
1024_1	0,7156	**0,7159**	10%
128_2	0,7405	**0,7422**	10%
256_2	0,7432	**0,7441**	10%
512_2	0,7495	**0,7499**	10%
1024_2	0,7507	**0,7510**	5%

N_K	r−STOCH	ν−STOCH	meilleur ν
128_4	0,7887	**0,7945**	10%
256_4	0,7884	**0,7913**	10%
512_4	0,7813	**0,7819**	10%
1024_4	**0,7848**	0,7834	5%
128_8	0,7776	**0,7930**	10%
256_8	0,7786	**0,7836**	10%
512_8	0,7834	**0,7853**	10%
1024_8	0,7813	**0,7828**	5%

TABLE 4.4 – Comparaison de ν−STOCH par rapport à r−STOCH. Les valeurs reportées pour r−STOCH sont les *meilleures* en moyenne de l'intervalle des arrondis non dominés.

4.2.3 Arrondi adaptatif du fitness basé sur la neutralité et la convergence

Comme pour les climbers classiques, ν−STOCH et r−STOCH finissent par être piégés tôt ou tard dans un optimum local, ou tout au moins une zone de l'espace de recherche dont ils ne peuvent s'échapper. Même si l'optimum local atteint est d'excellente qualité, il semble qu'il soit possible de trouver de meilleures solutions dans la plupart des paysages considérés en essayant de sortir de la zone atteinte, à la manière des recherches locales itérées (ILS). Nous nous sommes proposé de réaliser cela en nous basant sur le paramètre d'arrondi, qui sera alors adapté en fonction de la neutralité du paysage mais aussi en fonction de la convergence de la recherche.

Le schéma général de fonctionnement de la méthode, décrite dans le rapport d'Hugo Traverson, est le suivant : on réalise dans un premier temps une recherche de type ν−STOCH, en cherchant à approcher le plus possible une neutralité objectif (10% par exemple). Pendant la recherche, on met à jour une variable correspondant au taux γ de mouvements améliorants réalisés par rapport au nombre de solutions évaluées. Lorsque le taux de mouvements améliorants devient trop faible, on baisse alors la neutralité objectif, ce qui permet potentiellement de découvrir des mouvements améliorants puisque certains mouvements neutres sont alors devenus améliorants. À chaque fois

que le taux de convergence devient trop faible, on baisse la neutralité objectif, jusqu'à revenir sur le paysage NK sans neutralité. Dès lors, on atteint rapidement un optimum local. On définit ensuite $r = 1$ et l'on se trouve alors dans un paysage complètement plat. Le premier mouvement est donc complètement aléatoire, le taux de neutralité objectif aura pour effet d'augmenter assez vite le paramètre d'arrondi et de rendre la recherche moins aléatoire.

Cette approche, de type $\nu\gamma-$STOCH, est assez proche idéologiquement parlant des algorithmes de type recherche locale itérée. Nous avons remarqué qu'elle était peu sensible au paramétrage, car elle est très efficace pour un large intervalle de neutralité objectif (entre 10% et 50%), le paramètre de convergence servant principalement à savoir combien de temps on accorde à chaque phase de la recherche. Nous avons comparé les résultats de $\nu\gamma-$STOCH avec les meilleurs résultats moyens obtenus par un ensemble d'ILS, correspondants à différents paramétrages (22 en tout : utilisation de FIRST ou BEST et quantités diverses de mouvements aléatoires pratiqués sur le dernier optimum local ou sur la meilleure solution rencontrée depuis le lancement de la recherche).

N_K	meilleur ILS	$\nu\gamma-$STOCH
128_1	**0,7245**	**0,7245**
256_1	**0,7221**	**0,7221**
512_1	**0,7088**	0,7084
1024_1	**0,7175**	0,7171
128_2	0,7414	**0,7424**
256_2	0,7443	**0,7444**
512_2	0,7501	**0,7516**
1024_2	0,7518	**0,7525**

N_K	meilleur ILS	$\nu\gamma-$STOCH
128_4	0,7921	**0,7953**
256_4	0,7881	**0,7930**
512_4	0,7769	**0,7861**
1024_4	0,7778	**0,7881**
128_8	0,7807	**0,7912**
256_8	0,7815	**0,7877**
512_8	0,7837	**0,7872**
1024_8	0,7817	**0,7850**

TABLE 4.5 – Comparaison de $\nu\gamma-$STOCH avec les meilleurs résultats moyens obtenus obtenus sur chaque paysage par 22 paramétrages d'ILS.

La table 4.5 montre que très souvent $\nu\gamma-$STOCH est plus performant que l'ensemble des ILS, surtout lorsque la rugosité du paysage est élevée. Notons également que l'efficacité des ILS dépend assez fortement du paramétrage utilisé et que le paramétrage optimal change fréquemment selon l'instance considérée, ce qui n'est pas le cas pour $\nu\gamma-$STOCH. Les résultats détaillés obtenus par les 22 versions ILS peuvent être trouvés à l'adresse suivante : http://www.info.univ-angers.fr/pub/basseur/DATA/ils_NK.pdf.

4.2.4 Discussion

Globalement, la discrétisation de paysages de recherche semble être un moyen simple et efficace de les rendre plus faciles à explorer. Il serait évidemment intéressant d'appliquer ce principe à d'autres paysages issus de problèmes combinatoires pour déterminer si l'efficacité est également visible. En particulier, il n'est pas nécessairement évident qu'il y ait la possibilité d'améliorer l'efficacité des climbers par ajout de neutralité sur des paysages en possédant déjà naturellement.

Quoiqu'il en soit, les résultats sont très probants et les méthodes proposées semblent peu sensibles au paramétrage et aux paysages étudiés en comparaison à beaucoup de méthodes d'optimisation. Les méthodes d'arrondi présentées dans cette section sont dédiées aux paysages NK, mais peuvent être aisément adaptées à tout problème d'optimisation mono-objectif, quelle que soit l'allure de sa fonction objectif. Les algorithmes génériques r−STOCH (algorithme 4.2) et $\nu-$STOCH

(algorithme 4.3) sont faciles à mettre en oeuvre et s'appuient sur la définition de fonction de fitness arrondie donnée par l'équation 4.1.

Définition 5 (Fonction fitness arrondie) $\mathfrak{f}_\alpha(x)$ *est la fonction de fitness arrondie de paramètre* α *pour une solution* x*. Elle est définie par :*

$$\mathfrak{f}_\alpha(x) = \frac{\lfloor \alpha * \mathfrak{f}(x) \rfloor}{\alpha} \tag{4.1}$$

Remarquons que l'équation 4.1 s'adapte à toute fonction de fitness numérique classique. Par exemple, pour le flow shop où les fitness sont de l'ordre de quelques milliers, poser $\alpha = 0{,}1$ consiste à oublier le chiffre des unités de la fonction de fitness originale. Pour les paysages NK où les fitness sont dans l'intervalle $[0{,}1]$, poser $\alpha = 100$ consiste globalement à ne garder que deux chiffres de précision après la virgule.

Algorithme 4.2 Climber `r-STOCH`

Entrée : une configuration initiale x, un paramètre d'arrondi α
tant que x n'est pas un optimum local et nombre maximal d'évaluations non atteint **faire**
  sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}_\alpha(x') \geqslant \mathfrak{f}_\alpha(x)$
  $x \leftarrow x'$
Retourner x

Algorithme 4.3 Climber ν`-STOCH`

Entrée : une configuration initiale x, une neutralité objectif
$\alpha \leftarrow 0$
tant que le nombre maximal d'évaluations est non atteint **faire**
  si possible, sélectionner aléatoirement $x' \in \mathcal{N}(x)$ tel que $\mathfrak{f}_\alpha(x') \geqslant \mathfrak{f}_\alpha(x)$
  $x \leftarrow x'$
  si la neutralité observée est trop faible, alors diminuer α
  si la neutralité observée est trop élevée, alors augmenter α
Retourner x

L'avantage de l'algorithme ν`-STOCH` est qu'il peut être entièrement autonome. En effet la neutralité objectif semble facile à paramétrer (une valeur de l'ordre de 10% ou 20% semble toujours adaptée). Notons tout de même que l'étape de modification du paramètre α durant la recherche peut être faite de diverses manières plus ou moins réactives, dont celle proposée dans [Traverson, 2014].

Chapitre 5

Conclusions de la partie I

L'objectif des travaux présentés dans cette première partie était d'avoir une meilleure compréhension du fonctionnement des climbers, élément de base de nombreuses métaheuristiques. Lors de la mise au point de métaheuristiques avancées, ces questions quant aux composants de base des climber reviennent sans cesse, sans qu'il soit possible de s'appuyer sur des études précises pour faire ces choix a priori cruciaux.

Nous avons essayé d'apporter quelques réponses dans les deux premiers chapitres de cette partie. Dans un premier temps, nous avons proposé d'abstraire les problèmes d'optimisation combinatoire sous forme de paysages de recherche tout en proposant des indicateurs pour les caractériser. Nous avons également proposé une étude (préliminaire, restreinte à de petits paysages NK) montrant que l'optimum global est souvent atteignable par un simple climber. Puis, nous avons réalisé une étude expérimentale sur les principales implémentations basiques possibles des climbers afin de pouvoir donner des indications pour l'implémentation de futurs climbers. Ces expérimentations ont permis de mettre en exergue l'interêt de la recherche stochastique, qui consiste à accepter le premier voisin améliorant non-strict à chaque pas de la recherche. L'utilisation de la technique du meilleur améliorant avec perturbations neutres devant se limiter aux problèmes peu rugueux et certains problèmes spécifiques (a priori lorsque les variables de décision ont des poids très variables dans la fonction objectif). Évidemment, ces résultats pourront être complétés dans le futur avec une analyse plus fine des paysages de recherche et en étudiant des paysages encore plus variés, en proposant par exemple de nouveaux modèles de paysages NK.

Dans le chapitre 4, nous avons cherché à proposer de nouveaux climbers génériques, en nous appuyant sur les résultats des expérimentations précédentes ainsi que sur nos résultats sur l'atteignabilité de l'optimum global qui suggèrent qu'il est peut-être possible de concevoir des climbers bien plus performants que ceux existants. Nous avons proposé la règle pivot du pire améliorant, en nous basant sur l'idée que si choisir le premier améliorant est plus efficace que choisir le meilleur améliorant, cela indique qu'il vaut mieux favoriser de petites améliorations. Cela semblait indiquer que choisir le pire améliorant pouvait être encore plus efficace. Les résultats sur les paysages NK et sur le flow shop ont confirmé cette intuition ce qui nous a poussé a proposer une version *praticable* de la stratégie du pire améliorant en l'approximant. Ensuite, en nous appuyant sur les résultats du chapitre 3 sur les paysages NKr, nous avons proposé le climber stochastique appliqué aux paysages NK modifiés dynamiquement afin d'entretenir un certain niveau de neutralité durant la recherche. Les résultats obtenus ont montré un haut niveau de performance, en particulier puisque cela permet d'être plus efficace qu'un ensemble de recherches locales itérées testées.

Les différents travaux que nous avons réalisés sur l'analyse des paysages de recherche et sur l'analyse du comportement des climbers nous offrent de nombreuses perspectives de recherche. Le fait de se concentrer sur ce composant de base des métaheuristiques a permis d'entrevoir de nouvelles opportunités pour créer des métaheuristiques avancées efficaces.

L'**analyse des paysages** de recherche peut apporter des informations primordiales pour savoir comment résoudre un problème d'optimisation difficile. En particulier, le choix de l'opérateur de voisinage est souvent déterminant pour la qualité des métaheuristiques. Pour une fonction de fitness et un espace de recherche donnés, différents opérateurs de voisinages induisent des paysages de recherche qui peuvent être très différents, les connexions entre configurations étant généralement totalement bouleversées. Une étude des caractéristiques de robustesse et de neutralité des paysages engendrés peut certainement permettre de choisir l'opérateur adéquat, celui qui rendra plus efficace l'intensification de la recherche. Pour cela, il reste à définir comment comparer des paysages de recherche ayant des connectivités différentes (par exemple pour comparer des voisinages de tailles linéaires et quadratiques par rapport à la taille de l'instance).

Nous avons vu également que la précision de la fonction d'évaluation des configurations est déterminante pour définir la facilité d'un paysage de recherche à être exploré efficacement. Or, pour certains problèmes d'optimisation, la fonction de fitness n'est pas toujours évidente à définir, souvent parce que la fonction objectif du problème d'optimisation est trop peu discrétisée. On peut citer l'exemple du problème de coloration de graphes, où la fonction objectif est le nombre de couleurs utilisées (si on cherche à minimiser le nombre de couleurs différentes des sommets) ou le nombre de conflits (si l'on cherche à résoudre une coloration valide à k couleurs). Ces fonctions objectif sont assez peu discriminantes, c'est pourquoi il existe des études tentant d'affiner cette fonction pour calculer des fitness plus précis [Porumbel *et al.*, 2008]. Lorsque l'on se trouve devant ce genre de problématique où les paysages sont trop neutres, l'analyse des paysages engendrés par des fonctions de fitness affinées peu permettre de faire un choix efficace : il semble alors approprié de choisir la fonction permettant de diminuer la neutralité tout en augmentant le moins possible la rugosité du paysage résultant.

Enfin, nous avons vu que dans certains cas, baser l'analyse de paysage sur une estimation de ses caractéristiques de manière globale n'est pas forcément adapté lorsque la structure d'un paysage de recherche n'est pas uniforme. Par exemple, la neutralité n'est parfois observée que dans les meilleures zones du paysage et ne peut être détectée par échantilonnage. Or les métaheuristiques passent la majeure partie de leur temps dans ces zones de l'espace de recherche. Une perspective de travail pourrait donc être de déterminer de manière précise comment calculer les caractéristiques des paysages de recherche en tenant compte de cet aspect. Par exemple, on pourrait évaluer la rugosité et la neutralité de manière dynamique durant la recherche, ou définir comment estimer *a priori* ces indicateurs en fonction du fitness de la solution courante.

L'**étude des climbers** offre encore de nombreuses perspectives de recherche afin d'améliorer notre connaissance générale sur ce mécanisme d'intensification et d'en concevoir des plus performants. Nous avons montré que le taux de couverture de l'optimum global est assez élevé pour les paysages NK et il nous parait cohérent de conjecturer que cela soit le cas pour beaucoup de paysages de recherches issus de problèmes combinatoires classiques. Cela semble donc indiquer qu'il est fort probable qu'il soit possible de proposer des politiques de choix de mouvements bien plus performantes que celles existant actuellement. Les conclusions du chapitre 3 nous ont amené à proposer le climber `WORST` et surtout sa variante le `k-WORST` qui ont obtenu des résultats supérieurs aux climbers classiques, bien que cela ne fusse pas évident intuitivement. De nombreux

autres climbers pourraient être proposés et ainsi améliorer l'efficacité de la phase d'intensification des métaheuristiques. Une règle pivot que nous étudions actuellement consiste à systématiquement choisir le voisin améliorant qui permet de garder un maximum de voisins améliorants pour le mouvement suivant de la recherche (règle pivot EXPAND). Les premiers résultats indiquent que cette règle pivot est bien plus performante que les règles pivot classiques, ainsi que WORST. La table 5.1 donne les résultats obtenus par EXPAND comparés avec les résultats obtenus par WORST sur les paysages NK. Notons que sur les paysages faciles, où BEST est efficace, EXPAND est également largement supérieur.

Il convient d'atténuer l'importance de ces résultats car EXPAND est très gourmand en nombre d'évaluations puisqu'il nécessite d'explorer un niveau de voisinage supplémentaire pour chaque voisin améliorant de la solution courante. EXPAND atteint un optimum local en évaluant en moyenne 30 fois plus de solutions que WORST. Cependant, la règle pivot d'EXPAND respecte la définition des climbers et montre qu'il est effectivement possible de se diriger vers les bons optima locaux avec un climber. De plus, des expérimentations ont également montré qu'EXPAND est plus performant pour résoudre les paysages NK qu'un climber dont l'opérateur de voisinage autorise l'inversion de 2 bits au maximum (voisinage large). Cela est d'autant plus remarquable que l'utilisation d'un voisinage large permet d'autoriser le flip d'un bit qui détériore la fonction de fitness. Il doit être certainement possible de concevoir d'autres règles pivots avancées qui permettent d'être aussi efficace qu'EXPAND, mais en nécessitant un nombre d'évaluations de configurations limité.

N_K	WORST	EXPAND	N_K	WORST	EXPAND	N_K	WORST	EXPAND	N_K	WORST	EXPAND	N_K	WORST	EXPAND
128_4	0,7250	**0,7464**	128_6	0,7251	**0,7468**	128_8	0,7199	**0,7415**	128_10	0,7115	**0,7356**	128_12	0,7033	**0,7267**
256_4	0,7274	**0,7478**	256_6	0,7313	**0,7444**	256_8	0,7267	**0,7444**	256_10	0,7198	**0,7376**	256_12	0,7129	**0,7300**
512_4	0,7285	**0,7481**	512_6	0,7338	**0,7503**	512_8	0,7306	**0,7455**	512_10	0,7243	**0,7386**	512_12	0,7179	**0,7316**
1024_4	0,7298	**0,7490**	1024_6	0,7353	**0,7509**	1024_8	0,7330	**0,7464**	1024_10	0,7270	**0,7396**	1024_12	0,7210	**0,7321**

TABLE 5.1 – Résultats préliminaires de la règle pivot EXPAND : comparaison avec WORST sur les paysages NK rugueux.

L'**étude des mécanismes de diversification** est une perspective importante pour l'avenir de mes activités de recherche. Bien que ce soit un domaine de recherche énormément étudié, les résultats obtenus et en cours m'ont permis de repenser ce mécanisme dans le cadre de l'exploration de paysages de recherche. Il me semble que le terme "diversification" ne soit pas forcément adapté pour l'objectif qui est de pouvoir atteindre de nouveaux optima locaux. En effet, les paysages de recherche étant généralement de diamètres dérisoires en comparaison du nombre de configurations réalisables, ce n'est pas nécessairement la distance entre les optima locaux qui est problématique, mais plutôt leur atteignabilité via une amélioration itérative. Il me semble intéressant d'étudier des mécanismes capables de modifier dynamiquement un paysage de recherche de telle sorte que la recherche s'oriente naturellement vers des optima locaux qui n'étaient pas atteignables auparavant. Une autre piste en cours d'étude est de chercher à maximiser les possibilités de transitions durant la recherche (à la manière de EXPAND), mais en ne se limitant pas à de simples climbers.

Enfin, l'**intégration** des résultats obtenus dans des métaheuristiques avancées constitue une perspective de recherche à ne pas négliger. Il faut garder à l'esprit qu'il sera toujours difficile de faire mieux que des méthodes dédiées, tenant compte des spécificités des problèmes étudiés. De même, des méthodes hybrides faisant coopérer plusieurs métaheuristiques, méthodes exactes et modèles parallèles [Jourdan *et al.*, 2009] sont connues pour être globalement efficaces. L'idée est donc d'intégrer nos résultats dans ces métaheuristiques avancées, en modifiant leurs composants

de base puis en étudiant l'impact éventuel de ces modifications. Par exemple, il serait intéressant d'évaluer le comportement de nombreuses recherches locales itérées si l'on remplace la règle pivot utilisée par une règle pivot de type $\texttt{WORST}_{\kappa}$. Un moyen efficace de pouvoir utiliser facilement nos résultats sur les règles pivot ou sur l'analyse de paysage dans des métaheuristiques avancées, serait de les intégrer dans une plateforme d'optimisation telle que ParadisEO [Cahon *et al.*, 2004].

Deuxième partie

Métaheuristiques génériques pour l'optimisation combinatoire multiobjectif

L'optimisation multiobjectif prend ses racines dans les travaux en économie de Edgeworth [Edgeworth, 1881] et de Pareto [Pareto, 1896]. Les premiers travaux traitant de méthodes avancées pour l'optimisation multiobjectif datent des années 80 [Steuer, 1989], et le domaine a pris de l'ampleur depuis la fin des années 90 et l'apparition des algorithmes évolutionnaires multiobjectif utilisant directement la notion de dominance *de Pareto* [Deb, 2001]. Depuis, un nombre croissant de travaux portent sur les métaheuristiques pour l'optimisation multiobjectif, dont une grande part concerne les algorithmes évolutionnaires. Dans cette partie, nous nous intéresserons aux métaheuristiques multiobjectif en général, mais en nous concentrant sur un nouveau type d'approche apparu il y a quelque années : la résolution basée sur les *indicateurs de qualité*. Cette idée proposée initialement dans [Zitzler et Künzli, 2004] fait l'objet d'une popularité croissante dans la communauté. Depuis ma prise de fonction à Angers, l'essentiel de mes travaux concernant l'optimisation multiobjectif se basent sur cette approche de résolution. Après avoir introduit quelques concepts et définitions de base, la notion d'optimisation multiobjectif basée sur les indicateurs de qualité sera détaillée. Ensuite, les différents travaux réalisés seront présentés, l'objectif général étant de mettre au point des métaheuristiques capables d'être facilement adaptables à la résolution d'une large classe de problèmes d'optimisation multiobjectif.

Les travaux présentés dans cette partie sont décrit de manière plus complète dans plusieurs publications : l'algorithme IBMOLS à fait l'objet de publications dans [Basseur et Burke, 2007, Basseur *et al.*, 2012a] ; les travaux autour de HBMOLS ont été publiés dans [Basseur *et al.*, 2012b, Basseur et Liefooghe, 2013] ; les travaux concernant les algorithmes de *path-relinking* sont publiés dans [Zeng *et al.*, 2013b, Zeng *et al.*, 2013a] ; SbLS est publié dans [Basseur *et al.*, 2013]. Une partie des travaux réalisés dans le cadre de l'optimisation multiobjectif n'est pas présentée ici par soucis d'uniformité et pour limiter la longueur du document. Pour plus d'informations sur ces travaux, on pourra se référer par exemple aux articles de revues [RI-2], [RI-4], [RI-6] et [RI-7] (voir CV détaillé à la fin du document).

Chapitre 6

Contexte et définitions de bases

Dans ce chapitre, nous introduisons dans un premier temps quelques concepts et définitions de base de l'optimisation combinatoire multiobjectif, utiles pour comprendre la suite du manuscrit. Ensuite, nous ferons un tour rapide des méthodes de résolution du domaine, en nous concentrant sur l'aspect résolution de problèmes combinatoires multiobjectif à l'aide de métaheuristiques, sans connaissance de préférences du décideur.

6.1 Définitions

Soit $\mathcal{X}$ un espace de recherche (ou espace décisionnel) et x une solution de cet espace. $\mathfrak{F}(x) = (\mathfrak{f}_1(x),\mathfrak{f}_2(x),\ldots,\mathfrak{f}_n(x))$ est le *vecteur objectif* de x composé de $n \geqslant 2$ fonctions objectif. Résoudre un problème d'optimisation multiobjectif $(\mathcal{X},\mathfrak{F})$ consiste à trouver $x^\star \in \operatorname{argmin}_{x\in\mathcal{X}} \mathfrak{F}(x)$. L'espace des vecteurs objectif est appelé *espace objectif*. Comme pour l'optimisation mono-objectif, on peut indifféremment maximiser ou minimiser chacune des composantes de $\mathfrak{F}$. La plupart des problèmes étudiés dans cette partie sont des problèmes de minimisation ; dans ce qui suit, on se placera donc dans un contexte de minimisation.

En optimisation mono-objectif, il existe un ordre total sur $\mathbb{R}$ parmi les solutions réalisables. Dans un contexte multiobjectif, il n'existe généralement pas de solution optimale pour l'ensemble des objectifs étant donnée la nature souvent conflictuelle de ces derniers. En effet, étant données deux solutions de l'espace de recherche, il est courant que l'une soit meilleure que l'autre sur une partie des objectifs et moins bonne sur une autre partie des objectifs. Ainsi, il n'existe qu'une relation d'ordre partielle entre les solutions, s'appuyant sur la notion de *dominance de Pareto* :

Définition 6 (Dominance de Pareto) *Une solution $x \in \mathcal{X}$ domine une solution $x' \in \mathcal{X}$ si et seulement si $\forall i \in \{1,\ldots,n\}$, $\mathfrak{f}_i(x) \leqslant \mathfrak{f}_i(x')$ et $\exists i \in \{1,\ldots,n\}$ tel que $\mathfrak{f}_i(x) < \mathfrak{f}_i(x')$. On note alors $x \succ x'$.*

Cette relation nous permet de définir quels sont les cas où une solution est meilleure qu'une autre quelles que soient les préférences du décideur. Parfois, on utilise également la notion de *dominance faible*, qui inclut dans la relation les solutions équivalentes (telles que $\mathfrak{F}(x) = \mathfrak{F}(y)$) :

Définition 7 (Dominance faible) *Une solution $x \in \mathcal{X}$ domine une solution $y \in \mathcal{X}$ si et seulement*

si $\forall i \in \{1,\ldots,n\}$, $\mathfrak{f}_i(x) \leqslant \mathfrak{f}_i(y)$. *On note alors* $x \succeq y$.

En optimisation multiobjectif, on cherche à déterminer les solutions de l'espace de recherche qui offrent les meilleurs compromis possibles, c'est-à-dire qui ne sont dominées par aucune autre solution de l'espace de recherche. De telles solutions sont appelées solutions *Pareto optimales* :

Définition 8 (Solution Pareto optimale) *Une solution* $x \in \mathcal{X}$ *est dite* Pareto optimale *si et seulement si* $\forall y \in \mathcal{X}, y \nsucc x$.

Ces solutions sont aussi appelées solutions *non dominées*. Par la suite, on utilisera par abus de langage le terme de solution non dominée pour désigner une solution qui n'est pas dominée dans un ensemble $\mathcal{Z} \subseteq \mathcal{X}$.

Définition 9 (Ensemble Pareto optimal) *Soit* $\mathbf{PO} = \{\mathbf{x} \in \mathcal{X} \mid \nexists \mathbf{y} \in \mathcal{X}, \mathbf{y} \succ \mathbf{x}\}$ *l'ensemble des solutions Pareto optimales de l'espace de recherche.* $\mathbf{PO}$ *est appelé* ensemble Pareto optimal.

On cherche donc à déterminer cet ensemble Pareto optimal, ou du moins à en déterminer une approximation. On distingue parfois l'ensemble Pareto optimal maximal de l'ensemble Pareto optimal minimal. L'ensemble Pareto optimal maximal correspond à l'ensemble $\mathbf{PO}$ et contient toutes les solutions équivalentes pour chaque point Pareto optimal. Un ensemble Pareto optimal minimal ne contient qu'une seule solution pour chaque point non dominé dans l'espace des objectifs. Généralement, déterminer un ensemble Pareto optimal minimal est suffisant (sauf dans le cas où certains aspects, comme la robustesse ou l'incertitude des solutions, n'auraient pas été pris en compte dans les fonctions objectif).

Dans mes recherches récentes, je m'intéresse essentiellement aux algorithmes de type recherche locale. De la même manière qu'en optimisation mono-objectif, ces algorithmes se basent sur une notion de voisinage qui consiste à modifier légèrement un ensemble de configurations dans le but de l'améliorer, jusqu'à atteindre éventuellement un *optimum local*. Cette notion d'optimum local, commune en optimisation mono-objectif, a un sens également en optimisation multiobjectif, décrit dans la définition ci-dessous :

Définition 10 (Ensemble optimum local) *Soit une relation de voisinage* $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$. $\mathbf{LO}$ *est un optimum local au sens de Pareto si et seulement si :*

$$\forall x \in \mathbf{PO} \mid \forall \mathbf{x}' \in \mathcal{N}(\mathbf{x}), \mathbf{x} \succeq \mathbf{x}'$$

Les notions de *point idéal* et *point nadir* sont souvent utilisées en optimisation multiobjectif :

Définition 11 (Point idéal) *Le* point idéal $x^* = (x_1^*,\ldots,x_n^*)$ *est un vecteur qui minimise chaque fonction objectif séparément.* $x_i^* = min_{x \in \mathcal{X}} \mathfrak{f}_i(x), \forall i \in \{1,\ldots,n\}$.

Définition 12 (Point nadir) *Le* point nadir $x^n = (x_1^n,\ldots,x_n^n)$ *est un vecteur qui correspond au maximum de chaque fonction objectif parmi les solutions Pareto optimales.* $x_i^n = max_{x \in \mathbf{PO}} \mathfrak{f}_i(x)$, $\forall i \in \{1,\ldots,n\}$.

Lorsque l'on a une population $\mathbf{P}$ de solutions, seules celles étant non dominées sont susceptibles d'intéresser le décideur (en l'absence d'autres éléments à prendre en compte tels que la robustesse des solutions). On parle également d'ensemble de *solutions compromis* :

Définition 13 (Sous-ensemble non dominé) *Soit* $\mathbf{P}$ *un ensemble de solutions.* $ND(\mathbf{P})$ *correspond au sous-ensemble minimal des solutions non dominées de* $\mathbf{P}$ *:*

$$ND(\mathbf{P}) = \{\mathbf{x} \in \mathbf{P} | (\nexists \mathbf{y} \in \mathbf{P}, \mathbf{y} \succ \mathbf{x}) \wedge (\nexists \mathbf{x},\mathbf{y} \in \mathbf{P^2}, \mathfrak{F}(\mathbf{x}) = \mathfrak{F}(\mathbf{y}))\}$$

Les approches de résolution par agrégation des objectifs permettent de trouver uniquement les solutions Pareto optimales *supportées* (voir section 6.2). Dans ce cas il faut faire appel à d'autres mécanismes pour déterminer les solution Pareto optimales *non-supportées*.

Définition 14 (Solution supportée) *Les solutions se trouvant sur l'enveloppe convexe de l'ensemble Pareto optimal sont appelées solutions supportées.*

La figure 6.1 illustre l'ensemble des définitions introduites dans cette section sous la forme d'un exemple dans un cadre biobjectif.

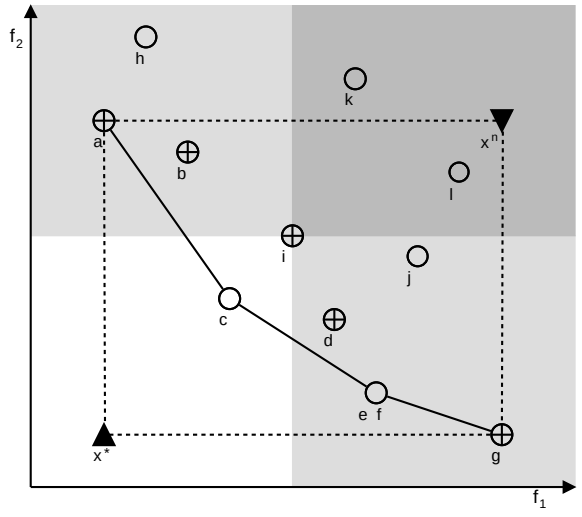


FIGURE 6.1 – Termes utilisés en optimisation multiobjectif : exemple dans le cadre biobjectif. Les cercles représentent les solutions de l'espace réalisable. Dominance de Pareto relativement à la solution i : la solution c, se trouvant sur la zone sur fond blanc, domine i ; les solutions l et k, se trouvant dans la zone gris foncé, sont dominées par i ; les autres solutions, se trouvant sur les zones légèrement grisées, sont incomparables à i. Les solutions a, b, c, d, e, f et g sont Pareto optimales, et forment l'ensemble Pareto optimal (maximal) $\mathbf{PO}$. Les solutions a, b, c, d, e et g forment un ensemble Pareto optimal minimal (les solutions e et f sont équivalentes : $e \succeq f$ et $f \succeq e$). Les solutions Pareto optimales supportées sont a, c, e, f et g et se trouvent sur l'enveloppe convexe des solutions réalisables ; les solutions d et g sont des solutions Pareto optimales non-supportées. Les croix (solutions a, b, d, g et i) sont mutuellement non dominées et forment une approximation de $\mathbf{PO}$. x^n est le point nadir de l'ensemble Pareto optimal, tandis que x^* est le point idéal.

6.2 Résolution de problèmes d'optimisation combinatoire multiobjectif

Dans cette section, nous rappelons brièvement différentes solutions proposées dans la littérature pour répondre aux spécificités des problèmes d'optimisation multiobjectif. Cela concerne trois aspects principaux :

- Approche de résolution : il convient de définir la place du décideur dans le processus d'optimisation. Son rôle est en général bien plus important qu'en optimisation mono-objectif.
- Prise en compte des objectifs : étant donné que le nombre d'objectifs empêche de pouvoir classer les solutions facilement, différentes approches ont été proposées afin d'y parvenir.
- Évaluation des méthodes d'optimisation : encore une fois, l'absence de relation d'ordre entre les solutions empêche de pouvoir évaluer facilement l'efficacité d'une méthode approchée de résolution. De nombreux indicateurs de qualité existent, et nous nous concentrerons sur le plus utilisé. De plus, étant donné le caractère stochastique des métaheuristiques, une validation statistique des résultats est nécessaire.

6.2.1 Approches de résolution

En optimisation (combinatoire) multiobjectif, on cherche à déterminer un ensemble de solutions compromis de la meilleure qualité possible, en essayant d'atteindre ou d'approcher l'ensemble Pareto optimal $\mathbf{PO}$. Cependant, dans un cadre réel, il convient de choisir une seule solution qui sera destinée à être mise en oeuvre. Un décideur doit donc intervenir pour effectuer un choix parmi les compromis possibles. D'une manière générale, celui-ci peut intervenir avant, pendant ou après l'application de la méthode de résolution du problème multiobjectif considéré. (1) *a priori* : les préférences entre les objectifs sont clairement établies avant le processus d'optimisation ; (2) *interactif* : le décideur affine son choix au fur et à mesure du déroulement de la méthode d'optimisation ; (3) *a posteriori* : le décideur choisit la solution de son choix parmi un ensemble de solutions compromis fournies par la méthode d'optimisation.

Les interventions du décideur, avant ou pendant la recherche, peuvent permettre de concentrer la recherche sur certaines zones de l'espace des objectifs à privilégier. Nous ne nous intéresserons pas ici aux possibles interventions du décideur par l'une de ces approches. Notre objectif sera principalement de rechercher un ensemble de solutions compromis le plus large et diversifié possible. Nous nous restreindrons donc à l'optimisation *a posteriori* de problèmes combinatoires multiobjectif.

6.2.2 Prise en compte des objectifs

Comme vu précédemment, la comparaison des solutions entre elles n'est pas triviale. Il convient donc de définir des stratégies de prise en compte des objectifs afin de pouvoir, parmi un ensemble de solutions, décider comment les ordonner en termes de qualité. Cette comparabilité est primordiale afin de pouvoir définir un algorithme de recherche. Nous distinguons ici quatre types d'approches, qui ont chacun fait l'objet d'un certain nombre d'études : les approches non scalaires, les approches scalaires, les approches Pareto et les approches basées sur les indicateurs de qualité.

Approches non scalaires Les méthodes non scalaires considèrent les objectifs individuellement [Ehrgott, 2005]. Par exemple, l'algorithme VEGA (*Vector Evaluated Genetic Algorithm*) [Schaffer, 1985] utilise un principe de *sélection parallèle* où une population de solutions est divisée en sous-populations, chacune étant formée des meilleurs individus pour un objectif donné (une sous-population par objectif à optimiser). Ces sous-populations sont combinées pour créer une nouvelle population qui sera divisée en plusieurs, et ainsi de suite. Une autre approche non scalaire couramment utilisée est la méthode lexicographique [Fourman, 1985], où les objectifs sont classés *a priori* par ordre d'importance par le décideur. Ensuite, les fonctions objectif sont traitées dans cet ordre durant le processus d'optimisation : toute amélioration d'un objectif donné sera préférée à l'ensemble des améliorations possibles des objectifs moins bien classés par le décideur.

Approches scalaires Le principe des méthodes scalaires est de ramener un problème multiobjectif à un problème mono-objectif à l'aide de paramètres. Si le décideur est capable de quantifier à l'avance l'importance relative des objectifs, cela permet alors de transformer le problème multiobjectif en un problème mono-objectif. Ceci induit l'obtention d'une solution optimale unique et donc l'absence de compromis à proposer au décideur. Lorsque l'on ne dispose pas de ces informations de la part du décideur, on peut résoudre alors un ensemble de problèmes mono-objectif, en faisant varier l'importance relative des objectifs.

Parmi ces méthodes scalaires, la *somme pondérée* des objectifs est certainement la plus utilisée. Cela consiste à transformer un problème multiobjectif en un problème qui combine les différentes fonctions objectif du problème $\mathfrak{f}_1, \ldots, \mathfrak{f}_n$ en une seule fonction $\mathfrak{f} : \mathfrak{f}(x) = \sum_{i=1}^{n} \lambda_i \mathfrak{f}_i(x)$. Différents poids fournissent différentes solutions, mais une même solution peut être générée en utilisant des poids différents. L'avantage de cette méthode est sa facilité de conception puisqu'une fois les poids établis, toute méthode d'optimisation mono-objectif peut être utilisée. Cependant, il a été montré que que cette méthode ne permet de trouver que les solutions supportées de l'ensemble Pareto optimal [Das et Dennis, 1996], même si des mécanismes spécifiques peuvent aider à les trouver, souvent *a posteriori*.

La méthode *ϵ-contrainte* optimise un seul objectif, les autres objectifs étant considérés comme des contraintes ($\mathfrak{f}_i \leqslant \epsilon_i$). L'ensemble des solutions de compromis est alors recherché en faisant varier les ϵ_i. On traite ainsi le problème d'optimisation multiobjectif en résolvant un ensemble de problèmes d'optimisation mono-objectif. Cette méthode permet de générer les solutions supportées aussi bien que les solutions non supportées, mais l'ajout de contraintes peut complexifier quelque peu le processus d'optimisation.

Approches Pareto Golberg [Goldberg *et al.*, 1989] a peut-être été le premier à utiliser les approches Pareto qui ont connu un essor remarquable à la fin des années 90. Pendant quelques années, de nombreux algorithmes proposés dans la littérature furent des approches Pareto appliquées sur des algorithmes évolutionnaires. Ces approches consistent à utiliser la notion de dominance Pareto pour comparer les solutions et leur affecter un fitness. Cette notion ne fournissant qu'une relation d'ordre très partielle, diverses méthodes de calcul ont été proposées, utilisant essentiellement la dominance relative des solutions d'un ensemble pour calculer leur fitness. Globalement, le score affecté à une solution est défini en fonction des solutions qu'elle domine et des solutions qui la dominent.

Il existe beaucoup de méthodes employant une approche Pareto [Coello *et al.*, 2007], les plus

populaires étant certainement NSGA-II [Deb *et al.*, 2002] et SPEA2 [Zitzler *et al.*, 2001]. Voici le calcul des fitness tel qu'il est réalisé par ces deux approches :

- NSGA-II : Le meilleur score (rang 1) est attribué aux solutions non dominées de la population. Puis le second meilleur score (rang 2) est attribué aux solutions qui ne sont dominées que par les solutions de rang 1, et ainsi de suite.
- SPEA2 : Pour chaque solution x de la population, on calcule dans un premier temps son poids, correspondant au nombre de solutions de la population qu'elle domine. Ensuite, le score d'une solution est défini en sommant les poids des solutions qui la dominent (un score de 0 affecté à une solution indique alors qu'elle est non dominée).

Approches basées sur les indicateurs de qualité Lorsque l'on souhaite comparer expérimentalement des métaheuristiques pour l'optimisation multiobjectif, des mesures doivent être utilisées afin de valider la performance relative des algorithmes. Mais l'existence d'un ensemble de solutions ainsi que l'absence de relation d'ordre total entre ces solutions rendent difficile la mesure de qualité d'approximations de l'ensemble Pareto optimal. De nombreuses métriques ont étés proposées pour évaluer la qualité d'une approximation ou pour comparer deux approximations entre elles [Zitzler *et al.*, 2008].

Ces indicateurs permettent de quantifier sous forme numérique la qualité d'un ensemble Pareto obtenu par une méthode d'optimisation. Ainsi, cela fourni une relation d'ordre entre les ensembles non dominés calculés par les métaheuristiques et donc de pouvoir déduire l'efficacité des algorithmes. Ces indicateurs doivent d'une part évaluer la qualité des solutions proposées par un algorithme, mais aussi la diversité des solutions proposées en termes de variété de compromis proposés au décideur. Zitzler et Künzli ont proposé d'étendre l'utilisation des indicateurs de qualité aux algorithmes évolutionnaires [Zitzler et Künzli, 2004]. Cela consiste principalement à définir l'opérateur de sélection de telle sorte qu'il tende à maximiser l'indicateur de qualité considéré. Depuis, de nombreuses études ont vu le jour à ce sujet, et une partie importante de mes recherches de ces dernières années ont porté sur cet aspect de l'optimisation multiobjectif. Ces approches feront l'objet du chapitre suivant, où leur principe général sera abordé plus en détails.

6.2.3 Évaluation d'approximations d'ensembles de Pareto

D'une manière générale, nous conservons le protocole d'expérimentation et d'analyse statistique des résultats présenté dans la section 2.3. La différence avec l'optimisation mono-objectif est que la comparaison d'approximations n'est pas directe puisqu'il n'existe pas de relation d'ordre total entre les ensembles non dominés. Il est donc nécessaire d'utiliser un indicateur de qualité permettant de faire correspondre une valeur scalaire à une approximation.

Parmi les nombreux indicateurs de qualité de la littérature, nous avons choisi d'utiliser l'indicateur de différence d'hypervolume (HD) proposé par Zitzler et Thiele [Zitzler et Thiele, 1999]. Cet indicateur se base sur la notion d'hypervolume (Hyp), introduite par les même auteurs. L'indicateur $Hyp(\mathbf{A},Z_{ref})$ correspond au volume de l'espace objectif faiblement dominé par une approximation $\mathbf{A}$. Ce volume est délimité par un point de référence Z_{ref} nécessairement dominé par toutes les solutions $\{x_0,\ldots,x_n\}$ de $\mathbf{A}$ (voir l'équation 6.1 et la figure 6.2).

$$Hyp(\mathbf{A},Z_{ref}) = VOL\left([x_0,Z_{ref}]\bigcup\ldots\bigcup[x_n,Z_{ref}]\right) \tag{6.1}$$

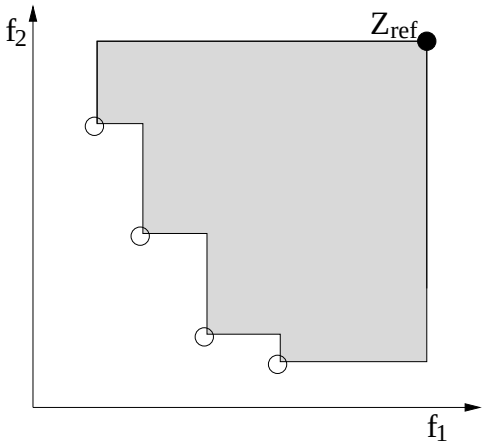


FIGURE 6.2 – Exemple en deux dimensions d'hypervolume entre un ensemble de référence $\mathbf{PO}^*$ et un point de référence Z_{ref} (zone grisée).

VOL correspond à la mesure de Lebesgue classique. Souvent, on confronte les approximations à des ensembles de références. $HD(\mathbf{A},\mathbf{R})$ calcule, en terme d'hypervolume, la différence entre une approximation $\mathbf{A}$ et un ensemble de référence $\mathbf{R}$, ce qui correspond au volume dominé par $\mathbf{R}$ et pas par $\mathbf{A}$ [1]. Idéalement, $\mathbf{R}$ correspond à l'ensemble Pareto optimal $\mathbf{PO}^*$ (voir la figure 6.3 et l'équation 6.2). Si, comme souvent, cet ensemble n'est pas calculable, alors on le définit comme l'ensemble non dominé de l'union des ensembles résultats obtenus sur l'instance de problème. Nous utilisons l'indicateur HD pour plusieurs raisons [Zitzler *et al.*, 2003] : (1) il évalue simultanément la qualité des solutions ainsi que leur répartition dans l'espace des objectifs ; (2) il est l'un des rares indicateurs de qualité à vérifier la propriété de transitivité de la relation d'ordre obtenue entre différents ensembles ; (3) la relation d'ordre entre les ensembles induite par HD respecte la relation d'ordre partielle induite par la dominance de Pareto ; (4) c'est l'opérateur le plus largement utilisé pour évaluer les algorithmes d'approximation des problèmes d'optimisation multiobjectif.

$$HD(\mathbf{A},\mathbf{PO}^*) = Hyp(\mathbf{PO}^*,Z_{ref}) - Hyp(\mathbf{A},Z_{ref}) \tag{6.2}$$

6.3 Métaheuristiques multiobjectif

Mes recherches se concentrent sur les approches de résolution *a posteriori*. Les travaux réalisés tendent à conserver la nature multiobjectif des problèmes étudiés, en excluant en particulier les méthodes d'agrégation des objectifs. Depuis l'essor des approches Pareto à la fin des années 90, de nombreuses métaheuristiques ont été adaptées au contexte qui nous intéresse ici. Dans un premier temps, les algorithmes évolutionnaires furent privilégiés, d'une part car ils étaient très populaires à cette époque, d'autre part car l'utilisation d'une population de solutions semblait adaptée pour la recherche d'un ensemble de solutions compromis, qui peut être alors découvert en une seule recherche.

Peu à peu, d'autres métaheuristiques ont été proposées, la plupart étant issues d'adaptations

1. Toutes les solutions de $\mathbf{A}$ doivent être faiblement dominées par au moins une solution de $\mathbf{R}$.

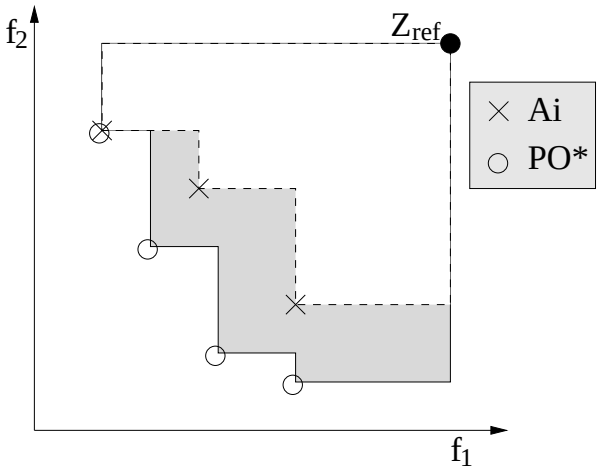


FIGURE 6.3 – Exemple en deux dimensions d'une différence d'hypervolume entre un ensemble de référence $\mathbf{PO}^*$ et une approximation $\mathbf{A_i}$ (zone grisée).

de métaheuristiques qui avaient été initialement proposées dans un cadre mono-objectif. L'adaptation des métaheuristiques classiques au contexte multiobjectif impose de faire des choix qui permettent souvent d'obtenir des métaheuristiques multiobjectif originales. Ces métaheuristiques ont fait l'objet d'un état de l'art publié avec El-Ghazali Talbi, Antonio Nebro et Enrique Alba [Talbi *et al.*, 2012]. Dans ce papier, différents types de métaheuristiques multiobjectif sont recensées et discutées. Diverses métaheuristiques non évolutionnaires sont recensées, mais également les métaheuristiques hybrides, qui sont devenues très étudiées récemment, consistant en la coopération entre une métaheuristique et une autre méthode d'optimisation (souvent une autre métaheuristique, mais peut-être aussi une méthode complète ou une heuristique spécifique). Dans cet état de l'art, les métaheuristiques multiobjectif parallèles sont également discutées, ainsi que la conception de métaheuristiques pour l'optimisation sous incertitude. Notons que durant mes activités de recherche, j'ai pu aborder tous ces aspects de l'optimisation multiobjectif. Dans la section suivante, je détaille le parcours de mes travaux concernant l'optimisation multiobjectif.

Métaheuristiques multiobjectif étudiées dans mes recherches

Une partie de mes travaux en optimisation multiobjectif sera présentée dans les chapitres suivants. Dans cette section, je présente l'évolution de mes recherches en mettant en exergue les liens qui ont permis cette évolution.

Au début de mon doctorat en 2001, je me suis principalement intéressé aux algorithmes évolutionnaires multiobjectif pour la résolution d'un problème d'ordonnancement de type flow shop. À cette époque, la quasi-totalité des travaux du domaine concernaient les algorithmes évolutionnaires. Ce domaine était d'ailleurs relativement récent (la première métaheuristique avec approche Pareto date de 1989 [Goldberg *et al.*, 1989]), et les méthodes qui sont devenues les plus reconnues aujourd'hui allaient à peine être publiées (SPEA2 en 2001 [Zitzler *et al.*, 2001] et NSGA-II en 2002 [Deb *et al.*, 2002]). Mes premiers travaux se concentraient sur la mise au point de mécanismes adaptatifs, conçus pour rendre un algorithme génétique le plus autonome possible. Cela concernait la sélection adaptative d'opérateurs de mutations, ainsi que le paramétrage automatique du mécanisme de diversification de la population de solutions [Basseur *et al.*, 2002]. Dans ces tra-

vaux, j'avais déjà établi un mécanisme de recherche locale itérée en complément de la phase de recherche évolutionnaire afin d'améliorer et d'accélérer la convergence de l'algorithme. Cette recherche locale, appelée aujourd'hui PLS (*Pareto Local Search*) a été depuis utilisée dans divers travaux (voir [Liefooghe *et al.*, 2012] pour plus de détails). Les recherches locales (mono-objectif et multiobjectif) constituent désormais mon sujet principal de recherche.

J'ai ensuite étudié d'autres types de méthodes durant ma thèse : les algorithmes mimétiques et la coopération parallèle en îles [Basseur *et al.*, 2003], la coopération entre méthodes exactes et approchées [Basseur *et al.*, 2004, Basseur *et al.*, 2005b] et les algorithmes de *path-relinking* [Basseur *et al.*, 2005a].

Mes premières années de recherche m'ont permis de réaliser qu'il y avait de nombreuses manières de rendre les méthodes d'optimisation efficaces sur un problème donné. Or, la plupart du temps les méthodes proposées ne sont pas facilement applicables et efficaces sur d'autres problèmes. De plus, l'approfondissement de l'étude d'un problème spécifique aboutit généralement à des méthodes complexes, hybridant plusieurs métaheuristiques et/ou méthodes complètes, et soumises à de nombreux paramètres.

Depuis, je me suis de plus orienté vers la conception de métaheuristiques simples, génériques, et peu sensibles au paramétrage, dans le but de mettre au point des méthodes facilement adaptables à la résolution de nouveaux problèmes d'optimisation. Je me suis aussi peu à peu orienté vers les recherches locales itérées, car elles se sont avérées particulièrement efficaces dans les différentes études effectuées au fil des années. Enfin, afin d'évaluer la robustesse et la généricité des métaheuristiques mises au point, je me suis de plus en plus efforcé d'étudier divers problèmes d'optimisation, même si cela implique beaucoup plus de travail de conception et d'expérimentations.

De plus, depuis mon post-doctorat réalisé en 2006 à l'ETH de Zurich, je me suis vivement intéressé à l'optimisation multiobjectif par les indicateurs de qualité. Cette méthode d'affectation d'efficacité aux solutions est très récente, et un nombre croissant de chercheurs, dont moi, s'intéressent à ce sujet. En particulier, une bonne partie de mon travail se concentre sur l'indicateur d'hypervolume de dominance, qui sera présenté plus tard. Les avantages des méthodes proposées dans ce domaine sont principalement le faible nombre de paramètres nécessaires, ce qui les rend facilement adaptable à divers problèmes, mais également une très bonne performance globale.

6.4 Étude de paysages multiobjectif

De manière similaire au cadre mono-objectif, un *paysage de recherche multiobjectif* est un triplet $(\mathcal{X},\mathcal{N},\mathfrak{F})$, où $\mathcal{X}$ est un *espace de recherche* contenant un ensemble de *configurations* (ou *solutions candidates*), $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$ est une *relation de voisinage*, et $\mathfrak{F} = (\mathfrak{f}_1(x),\mathfrak{f}_2(x),\ldots,\mathfrak{f}_n(x))$ est un *vecteur de fonctions objectif*. $\mathfrak{F}(x)$ représente le *vecteur objectif* de x.

Il est important de noter que l'étude d'un paysage de recherche dans le cadre multiobjectif est rendue difficile du fait de l'absence de relation d'ordre total entre les solutions. Ce point n'a donc pas encore été abordé en profondeur dans mes travaux en optimisation multiobjectif. Deux approches semblent réalisables, mais pour lesquelles une absence de résultats probants n'est pas à exclure. Premièrement, on peut redéfinir des indicateurs de rugosité et de neutralité dans un cadre multiobjectif, mais il sera certainement difficile de leur donner une réelle justification.

Deuxièmement, on peut utiliser une approche basée sur les ensembles (voir le chapitre 10), qui consiste à considérer un ensemble de solutions de compromis comme un tout, c'est à dire comme une *solution-ensemble* du problème d'optimisation considéré ; ainsi, les indicateurs de neutralité et de rugosité, présentés dans la deuxième partie du document, peuvent être réutilisés, mais leur signification réelle est loin d'être assurée. Bien que l'analyse de paysages multiobjectif semble extrêmement complexe, il existe néanmoins des travaux tentant de caractériser de tels paysages [Garrett et Dasgupta, 2008, Verel *et al.*, 2011a].

D'autres indicateurs caractérisant les paysages sont complètement propres aux problèmes multiobjectif et ne font pas entrer en compte l'opérateur de voisinage utilisé. Ces indicateurs concernent la disposition des solutions dans l'espace objectif, qui a un impact important sur le comportement des méthodes d'optimisation. Les figures 6.4 à 6.9 présentent des paysages dans l'espace objectif aux diverses spécificités. Parmi les particularités des solutions dans l'espace objectif, on peut notamment citer :

- **Corrélation** : La corrélation des fonctions objectif a un impact considérable sur le comportement des métaheuristiques. En particulier, l'allure et la cardinalité du front de Pareto en dépend fortement, selon que les fonctions objectif sont corrélées positivement (figure 6.4), négativement (figure 6.5) ou non corrélées (figure 6.6). En particulier, la frontière de Pareto tend à être particulièrement étendue lorsque les objectifs sont négativement corrélés (il est alors facile de trouver des solutions incomparables). Au contraire, cette frontière peut être très réduite lorsque les objectifs sont positivement corrélés. S'il y a une forte corrélation positive de deux objectifs, il peut être alors préférable de ne pas optimiser l'un de ces objectifs, car optimiser une fonction objectif revient alors quasiment à optimiser la seconde fonction objectif. Les travaux cherchant à réduire le nombre d'objectifs à optimiser s'appuient généralement sur des mesures de corrélation pour sélectionner un ensemble d'objectifs pertinents (non corrélés positivement entre eux).
- **Cardinalité** : Le nombre de solutions de Pareto de l'espace de recherche est également un critère influant sur les recherches réalisées par les métaheuristiques et méthodes complètes. Si ce nombre est trop grand, cela peut poser un problème d'archivage, voire d'énumération. De nombreux problèmes multiobjectif sont dits *impraticables*, c'est-à-dire que l'énumération même des solutions de Pareto est non polynomiale. Au contraire, pour certains problèmes, il est difficile de trouver plusieurs solutions de compromis, et cela peut mettre en péril l'efficacité de certains algorithmes de recherche.
- **Convexité de l'ensemble de Pareto** : Pour la plupart des problèmes d'optimisation combinatoire multiobjectif, l'allure du front de Pareto est globalement convexe (voir figures 6.4, 6.5 et 6.6), selon le niveau de corrélation des objectifs. Dans ce cas, une partie non négligeable des solutions de Pareto se trouvent sur l'enveloppe convexe des solutions dans l'espace objectif, même si cette proportion est souvent relativement basse (comme pour le flow shop [Basseur *et al.*, 2002]). Lorsque l'allure du front de Pareto est concave, il se peut que quasiment toutes les solutions de Pareto ne fassent pas partie de l'enveloppe convexe (figure 6.7). Cette particularité des fronts de Pareto a une importance, surtout si l'on utilise les approches de résolution par agrégation des objectifs pour résoudre le problème sous-jacent.
- **Discontinuité de l'ensemble de Pareto** : Par abus de langage, on dira que la frontière de Pareto est discontinue lorsqu'il existe de grands vides entre différentes zones de la frontière de Pareto. Cette discontinuité peut être conséquente à une discontinuité de la répartition des solutions dans l'espace objectif (figure 6.8) et/ou lorsque la zone de l'espace des objectifs où se trouve la frontière de Pareto est fortement concave (figure 6.9).

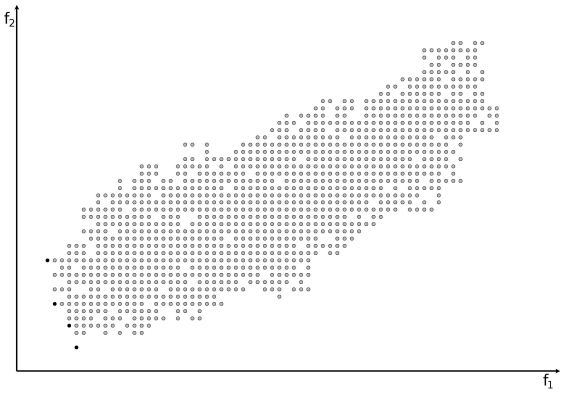


FIGURE 6.4 – Objectifs positivement corrélés.

FIGURE 6.5 – Objectifs négativement corrélés.

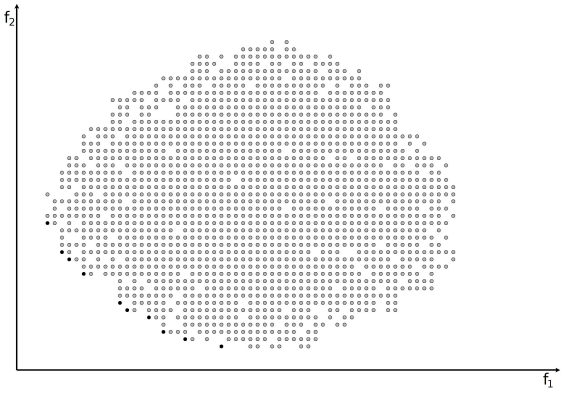


FIGURE 6.6 – Objectifs non corrélés.

FIGURE 6.7 – Frontière de Pareto Concave.

FIGURE 6.8 – Frontière de Pareto convexe et discontinue.

FIGURE 6.9 – Frontière de Pareto concave et discontinue.

L'analyse et la caractérisation de paysages de recherche multiobjectif est clairement une perspective importante de mon travail de recherche. L'objectif serait de réaliser une étude similaire à ce qui est proposé dans la première partie du document, mais pour l'optimisation multiobjectif, dans le but d'analyser par exemple les climbers multiobjectif et étudier leur comportement en

fonction des caractéristiques des paysages considérés. Néanmoins, trouver des indicateurs pertinents permettant de caractériser de manière précise les paysages multiobjectif n'est certainement pas une chose aisée et soulève de nombreuses questions. En particulier, les notions de neutralité et de rugosité se doivent d'être redéfinies dans le contexte multiobjectif : il existe certes des formulations intuitives de ces indicateurs dans le cadre multiobjectif, mais il parait difficile de déterminer dans quelle mesure ils permettent de caractériser de manière efficace des paysages de recherche multiobjectif.

Chapitre 7

Recherche locale multiobjectif basée sur les indicateurs binaires de qualité

Dans ce chapitre, nous présentons l'algorithme IBMOLS (*Indicator-Based MultiObjective Local Search*) qui consiste en un climber multiobjectif où la taille de l'ensemble de compromis manipulé est bornée. IBMOLS a l'avantage d'être particulièrement simple et dépendant de peu de paramètres. De plus les expérimentions menées montrent une performance intéressante et une faible sensibilité au paramétrage.

7.1 Indicateurs de qualité

Les indicateurs de qualité ont dans un premier temps été proposés afin d'évaluer et de comparer les différentes métaheuristiques proposées dans la littérature. Depuis les travaux de Zitzler et Künzli [Zitzler et Künzli, 2004], de nombreuses études ont vu le jour à ce sujet, et la majeure partie de mes derniers travaux en optimisation multiobjectif portent sur cet aspect.

7.1.1 Principe

Un indicateur de qualité $I(\mathbf{A},\mathbf{B})$ consiste à quantifier sous forme de scalaire la différence en termes de qualité entre deux ensembles de vecteurs objectif $\mathbf{A}$ et $\mathbf{B}$ [Zitzler *et al.*, 2003]. Soit $\mathbf{R}$ un ensemble de référence (si possible l'ensemble Pareto optimal), alors l'objectif global du processus d'optimisation peut être formulé ainsi :

$$\text{argmin}_{\mathbf{A}\in\mathcal{M}(\mathcal{X})}\ I(\mathbf{A},\mathbf{R}) \tag{7.1}$$

$\mathcal{M}(\mathcal{X})$ représentant l'espace des ensembles de vecteurs objectifs. $\mathbf{R}$ étant fixé, I devient donc un indicateur unaire qui affecte un scalaire à chaque ensemble de solutions, dont l'objectif est de minimiser sa valeur.

Avec ce type d'évaluation d'ensembles, il est possible de connaître la qualité globale d'une population de solutions et donc de s'en servir dans le processus de sélection des algorithmes évolutionnaires. En effet, il suffit de favoriser la sélection d'ensembles de telle sorte que l'indicateur de qualité tende à diminuer. Par exemple, lors de la sélection d'un sous-ensemble de la population

courante, cela consiste à supprimer les solutions de manière à minimiser la perte de qualité selon l'indicateur de qualité utilisé.

Comme indiqué dans l'article fondateur [Zitzler et Künzli, 2004], il est préférable que l'indicateur choisi soit conforme avec la notion de dominance de Pareto, comme définie ci-dessous (cette définition est valable pour des ensembles de solutions également) :

Définition 15 *Un indicateur binaire I est conforme avec la dominance de Pareto si :*
(1) pour tout x_1,$x_2 \in \mathcal{X}$, $x_1 \succ x_2 \Rightarrow I(x_1,x_2) < I(x_2,x_1)$, et
(2) pour tout x_1,x_2,$x_3 \in \mathcal{X}$, $x_1 \succ x_2 \Rightarrow I(x_3,x_1) \geq I(x_3,x_2)$.

7.1.2 Indicateurs binaires

Nous nous sommes intéressés dans un premier temps aux indicateurs binaires de qualité proposé par Zitzler et Künzli : l'indicateur (additif) I_ϵ (équation 7.2) et l'indicateur d'hypervolume I_{HD} (équation 7.3). Ces indicateurs évaluent la qualité d'une solution par rapport à une autre. Pour évaluer la qualité globale d'une solution par rapport à une population, on combine alors l'ensemble des valeurs calculées.

$$I_\epsilon(x_1,x_2) = max_{i\in\{1,\ldots,n\}}(\mathfrak{f}_i(x_1) - \mathfrak{f}_i(x_2)) \tag{7.2}$$

$I_\epsilon(x_1,x_2)$ ($x_1 \in \mathcal{X}$ et $x_2 \in \mathcal{X}$) représente la translation minimale, dans l'espace objectif, à appliquer à x_1 pour dominer x_2 (voir la figure 7.1). Cette translation peut prendre des valeurs négatives.

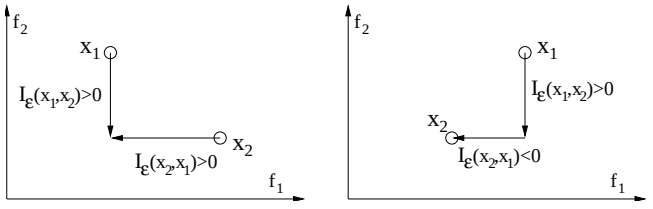


FIGURE 7.1 – Exemple illustratif du calcul de I_ϵ, appliqué à deux solutions x_1 et x_2 (à gauche : cas sans relation de dominance entre x_1 et x_2 ; à droite : $x_2 \succ x_1$).

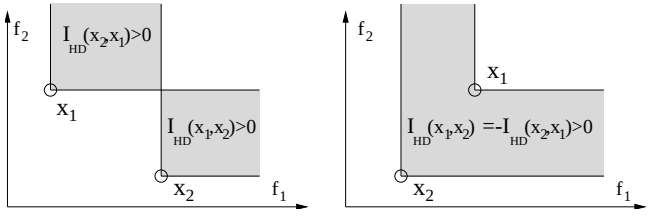


FIGURE 7.2 – Exemple illustratif du calcul de I_H, appliqué à deux solutions x_1 et x_2 (à gauche : cas sans relation de dominance entre x_1 et x_2 ; à droite : $x_2 \succ x_1$).

$$I_{HD}(x_1,x_2) = \begin{cases} H(x_2) - H(x_1) & \text{si } x_2 \succ x_1 \text{ ou } x_1 \succ x_2 \\ H(x_1 + x_2) - H(x_1) & \text{sinon} \end{cases} \tag{7.3}$$

$H(x_1)$ correspond à l'hypervolume de l'espace des objectifs dominé par x_1 (par rapport à un point de référence). $I_{HD}(x_1,x_2)$ représente l'hypervolume de l'espace des objectifs dominé par x_2, mais pas par x_1 (voir la figure 7.2). Le point de référence est fixé à 2 fois la valeur maximale de l'ensemble des solutions sur chaque objectif, comme suggéré par Zitzler et Künzli (cela favorise ainsi la sélection des solutions extrêmes de la population et maintenir ainsi une diversité des solutions dans l'espace des objectifs).

Afin d'évaluer la qualité d'une solution par rapport à une population P de solutions, plusieurs méthodes sont proposées dans [Zitzler et Künzli, 2004]. La somme des indicateurs binaire calculée semble inadaptée puisque cela donne beaucoup d'importance aux valeurs calculées pour des solutions très éloignées les unes des autres dans l'espace objectif ; la valeur minimale semble d'avantage indiquée. La méthode utilisée par Zitzler et Künzli est très proche, amplifiant grandement l'impact des petit indicateurs calculés par rapport aux grandes valeurs :

$$I(\mathbf{P} \setminus \{x\},x) = \sum_{z \in \mathbf{P} \setminus \{x\}} -e^{-I(z,x)/\kappa} \tag{7.4}$$

$\kappa > 0$ représente un facteur d'échelle ; une valeur proche de zéro tend à favoriser les plus petits indicateurs calculés. Lorsque κ tend vers 0, cela correspond au calcul de l'indicateur minimal, mais permet en plus, en cas d'égalité, de favoriser la solution qui a le plus petit deuxième indicateur minimal. Il semble donc approprié de choisir κ proche de 0.

Dans [Basseur *et al.*, 2012a], nous avons transformé en indicateurs binaires des méthodes de *ranking* basées sur la dominance de Pareto. Trois autres indicateurs de qualité ont ainsi été obtenus. Ces indicateurs sont I_{Ben}, I_{Fon} et I_{Sri} et sont respectivement basés sur les méthodes de *ranking* de [Fonseca et Fleming, 1993], [Srinivas et Deb, 1994] et [Bentley et Wakefield, 1997]. Cela nous a permis de valider l'efficacité des indicateurs I_{HD} et I_ϵ proposés par Zitzler et Künzli.

Dans la section suivante, l'algorithme IBMOLS (*Indicator-Based MultiObjective Local Search* [Basseur *et al.*, 2012a]) est présenté. Il se base sur la notion d'indicateur de qualité initialement définie au sein de l'algorithme évolutionnaire IBEA proposé en 2004 [Zitzler et Künzli, 2004].

7.2 IBMOLS

7.2.1 Recherches locales multiobjectif

Les algorithmes de recherche locale multiobjectif peuvent s'apparenter aux climbers en optimisation mono-objectif. L'idée est d'améliorer de manière itérative la qualité globale d'un ensemble de solutions mutuellement non dominées jusqu'à atteindre un optimum local (multiobjectif — voir définition 10). Nous gardons ici le terme *recherche locale* plutôt que *climber*, car améliorer petit à petit un ensemble de solutions non dominées ne correspond pas exactement au fait de grimper sur un paysage de recherche.

Si l'on veut se rapprocher des principes des climbers mono-objectif, un algorithme de recherche locale consiste à explorer le voisinage des solutions, en gardant les meilleures solutions rencontrées et en supprimant les solutions de moins bonne qualité. En d'autres termes, cela consiste à maintenir un ensemble de solutions non dominées, en ajoutant les solutions voisines non dominées et en supprimant systématiquement les solutions qui deviennent dominées au cours de la

recherche. Ceci est l'idée générale de l'algorithme PLS (*Pareto Local Search*). Le critère d'acceptation se base donc sur une relation de dominance arbitraire, telle que la dominance Pareto. Une des particularités de PLS est qu'elle manipule une population de taille variable. Des approches de type PLS ont été proposées dans divers travaux plus ou moins simultanément. On se réfèrera à [Liefooghe *et al.*, 2012] pour une étude de différente variantes de la méthode PLS. À chaque itération, PLS sélectionne une solution non-visitée de l'archive, et explore son voisinage exhaustivement. Il contient une condition d'arrêt naturelle qui est vérifiée lorsque le voisinage de tous les membres de l'archive a été exploré. Une archive non bornée est généralement utilisée, même si un mécanisme de limitation de la taille peut très facilement y être inclus. La description générique de PLS est proposée dans l'algorithme 7.1.

Algorithme 7.1 PLS.

Entrée : $\mathbf{P}$ (ensemble initial de solutions mutuellement non dominées)
Sortie : approximation de l'ensemble de Pareto
Tant que il reste au moins une solution inexplorée
  choisir une solution inexplorée $x \in \mathbf{P}$
  $\mathbf{P} \leftarrow \mathbf{P} \cup \mathcal{N}(x)$ (les solutions de $\mathcal{N}(x)$ sont marquées comme inexplorées)
  $\mathbf{P} \leftarrow ND(\mathbf{P})$
  marquer x comme explorée
retourner $\mathbf{P}$

Le problème principal de PLS est la non-maîtrise de l'explosion possible du nombre de solutions non dominées et donc de la taille de la population. Ceci peut aboutir à une convergence extrêmement lente de l'algorithme à moins que l'on intègre à PLS un mécanisme permettant de contrôler efficacement le nombre de solutions non dominées de l'archive. Le cas extrême inverse peut aussi exister, lorsque PLS n'arrive pas à générer un nombre suffisant de solutions non dominées durant la recherche, ce qui peut entraîner une convergence prématurée de l'algorithme.

Lors de mon séjour en 2005 à Zurich, j'ai découvert l'algorithme évolutionnaire IBEA, et je fus rapidement séduit par sa simplicité et le peu de paramètres entrant en compte, le tout allié à une performance globale très bonne. L'algorithme IBMOLS proposée dans la section suivante est une simple adaptation de l'algorithme IBEA pour le transformer en recherche locale basée sur une population de solutions. L'énorme avantage de IBMOLS par rapport aux recherches locales de type PLS, c'est que la taille de la population reste fixe. Ceci permet d'éviter l'explosion du nombre de solutions non dominées et IBMOLS complète naturellement la population avec des solutions dominées lorsque peu de compromis sont trouvés.

7.2.2 Description de l'algorithme IBMOLS

Une étape de l'algorithme IBMOLS consiste, pour chaque solution x de la population courante, à générer un à un les voisins de x, jusqu'à ce que l'un d'eux soit meilleur que la pire solution de la population suivant l'indicateur binaire de qualité utilisé. IBMOLS est détaillé dans l'algorithme 7.2.

Quelques précisions sont données ci-après. Le voisinage est systématiquement exploré dans un ordre aléatoire, sans remise. On sélectionne alors le premier voisin intéressant rencontré pour remplacer la plus mauvaise solution de la population. De plus, les valeurs des vecteurs objectifs des solutions sont normalisées sur l'intervalle $[0,1]$, pour éviter les problèmes d'échelle pouvant sur-

venir pour beaucoup d'indicateurs binaires de qualité (dont I_ϵ). Les détails de cette normalisation se trouvent dans [Basseur *et al.*, 2012a].

Algorithme 7.2 IBMOLS

Entrée : N (taille de population)
I (indicateur binaire de qualité)
$\mathbf{P}$ (population initiale de taille N)
Sortie : $\mathbf{A}$ (approximation de l'ensemble de Pareto)
étape 1 : $\mathbf{A} \leftarrow ND(\mathbf{P})$
étape 2 - assignation initiale de l'efficacité : $\forall x \in \mathbf{P}$,*Fit*$(x) = I(\mathbf{P} \setminus \{x\},x)$.
étape 3 - pas de recherche locale : $\forall x \in \mathbf{P}$ faire :
normalisation des vecteurs objectif de $\mathbf{P}$ sur $[0,1]^n$
répéter
1) $x^* \leftarrow$ voisin inexploré de x
2) $\mathbf{P} \leftarrow \mathbf{P} \cup x^*$
3) évaluation du fitness de x^* : $I(\mathbf{P} \setminus \{x^*\},x^*)$
4) mise à jour des autres fitness de $z \in \mathbf{P}$: *Fit*$(z) += I(x^*,z)$
5) $\omega \leftarrow$ pire solution de $\mathbf{P}$
6) supprimer ω de $\mathbf{P}$
7) mise à jour des autres fitness de $z \in \mathbf{P}$: *Fit*$(z) -= I(w,z)$
jusqu'à ce que tous les voisins de x soient explorés ou si $\omega \neq x^*$ (*amélioration* de $\mathbf{P}$)
étape 4 - critère d'arrêt : $\mathbf{A} \leftarrow$ ND($\mathbf{A} \bigcup \mathbf{P}$). Si $\mathbf{A}$ n'a pas été modifiée, alors retourner $\mathbf{A}$; sinon retourner à l'étape 3.

L'algorithme IBMOLS possède quelques spécificités intéressantes, qui rendent cette méthode simple, générique et performante :

- L'algorithme est original, puisque la grande majorité des métaheuristiques multiobjectif sont des algorithmes évolutionnaires et très souvent utilisent des fonctions de ranking basées sur la dominance de Pareto ou sur une agrégation des objectifs.
- La diversité des solutions est naturellement préservée. Elle doit être prise en compte par l'indicateur binaire de qualité qui est basé sur une évaluation globale des solutions en termes de qualité et de diversité, par rapport aux autres solutions de la population.
- L'indicateur de qualité utilisé peut prendre en compte certaines préférences du décideur pour guider la recherche.
- La recherche locale utilise une population de solution de taille fixée. Cela permet de trouver un ensemble de solutions non dominées en une seule recherche tout en contrôlant le nombre de solutions non dominées stockées durant la recherche. L'archive, quant à elle, recense néanmoins l'ensemble des solutions non dominées découvertes durant la recherche.

Naturellement, la recherche locale finit par atteindre un optimum local multiobjectif, qui est souvent assez éloigné de l'ensemble Pareto optimal. Pour cela, une version itérée d'IBMOLS a été mise au point, comme cela se fait souvent en optimisation mono-objectif. L'algorithme 7.3 décrit ce mécanisme. Une approximation $\mathbf{PO}$ de l'ensemble Pareto optimal est maintenue et mise à jour régulièrement. Après chaque recherche locale, $\mathbf{PO}$ est mis à jour avec le résultat obtenu par l'algorithme IBMOLS, puis une nouvelle population initiale créée à partir de $\mathbf{PO}$ sera utilisée pour la prochaine exécution de l'algorithme IBMOLS. La création d'une nouvelle population à partir d'une archive se fait via la fonction *genPop*, pour laquelle plusieurs possibilités ont été étudiées dans mes travaux : aléatoire, marche aléatoire, croisement et *path-relinking* (cette dernière méthode sera étudiée dans le chapitre 9).

Algorithme 7.3 Algorithme IBMOLS itéré

Entrée : N (taille de la population)
I (indicateur binaire de qualité)
Output : $\mathbf{PO}$ (approximation de l'ensemble de Pareto)
étape 1 : $\mathbf{PO} \leftarrow \emptyset$
étape 2 : tant que le temps d'exécution maximal n'est pas atteint faire
1) $\mathbf{P} \leftarrow genPop(\mathbf{PO},N)$
2) $\mathbf{A} \leftarrow IBMOLS(N,I,\mathbf{P})$
3) $\mathbf{PO} \leftarrow ND(\mathbf{PO} \cup \mathbf{A})$
étape 3 : Retourner $\mathbf{PO}$

7.3 Résultats expérimentaux

Pour étudier l'applicabilité et la performance globale de l'algorithme IBMOLS, celui-ci à été éprouvé sur différents problèmes dans [Basseur *et al.*, 2012a] : flow shop biobjectif, *ring star* biobjectif et emploi du temps d'infirmières à trois objectifs. Afin de réduire la place des expérimentations dans ce manuscrit, seuls les résultats sur le flow shop sont exposés ici. Notons également que l'algorithme IBMOLS a été comparé avec les algorithmes IBEA [Zitzler et Künzli, 2004], NSGA-II [Deb *et al.*, 2002] et SEAMO-R [Landa-silva et Le, 2008], et sa supériorité globale a été identifiée. Une description plus détaillée des expérimentations réalisées et de l'analyse des résultats peut être trouvée dans [Basseur *et al.*, 2012a].

Dans ce qui suit, nous présentons d'abord succinctement le problème de flow-shop étudié et discuterons des valeurs des paramètres utilisées dans les expérimentations. Ensuite nous présenterons les résultats afin d'en tirer les conclusions en termes de performance, mais aussi en termes de sensibilité au paramétrage.

7.3.1 Problème de flow shop biobjectif

Nous avons étudié dans la partie I problème de flow shop dans un cadre où un seul objectif est optimisé : la date de fin d'ordonnancement. Cependant, ce problème est naturellement multiobjectif, de nombreuses études de la littérature traitent des objectifs différents, qui peuvent être considérés de manière simultanée [Amit *et al.*, 1995, Landa Silva *et al.*, 2004, T'Kindt et Billaut, 2006].

Rappelons que le problème de flow shop (FSP) se présente comme un ensemble de n travaux (jobs) $\{J_1,J_2,\ldots,J_n\}$ à ordonnancer sur m machines $\{M_1,M_2,\ldots,M_m\}$, les machines étant des ressources critiques. On considère souvent que chaque job J_i possède également une date de fin souhaitée d_i (et parfois également une date de disponibilité).

Nous considérons ici deux objectifs très étudiés : la date de fin d'ordonnancement C_{max} (voir le chapitre 2.2.2), et le retard total T. Sachant que s_{ij} est la date à laquelle la $j^{\text{ième}}$ tâche du job i a été terminée. Le retard total peut être formulé de la manière suivante :

$$T = \sum_{i=1}^{n} [\max(0,s_{iM} + t_{iM} - d_i)]$$

Minimiser C_{max} est un problème NP-difficile lorsque $m > 2$, minimiser T est NP-difficile même lorsque $m = 1$. Les instances considérées sont celles proposées dans [Talbi *et al.*, 2001]

et sont une extension au cas biobjectif des instances de Taillard utilisées précédemment (ajout de dates de fin souhaitées).

7.3.2 Paramètres

Pour l'application de l'algorithme IBMOLS au problème de flow shop, nous avons choisi une représentation des individus classique, sous forme de permutation de jobs, comme nous l'avons fait pour la version mono-objectif du problème. De même, l'opérateur de voisinage est l'opérateur d'insertion, connu comme étant le plus efficace pour optimiser l'objectif C_{max} [Taillard, 1993].

La sensibilité au paramétrage d'IBMOLS a été évaluée et a requis de réaliser un grand nombre de tests. Pour limiter ce nombre, la sensibilité de l'algorithme a été étudiée pour chaque paramètre séparément. IBMOLS dépend essentiellement de trois paramètres, ce qui est peu au regard de beaucoup de métaheuristiques multiobjectif avancées : l'indicateur binaire de qualité, la taille de population et la fonction d'initialisation des différentes populations initialisées pour exécuter l'algorithme. Les autres paramètres sont secondaires et doivent seulement être définis de manière cohérente. κ est défini très petit (10^{-3}), et le calcul de I_{HD} requiert un point de référence qui a été fixé à $(2{,}2)$ (valeurs normalisées) comme suggéré dans les travaux de Zitzler et Künzli.

7.3.3 Protocole expérimental

Nous ne cherchons pas a démontrer la supériorité de l'algorithme IBMOLS sur les meilleures méthodes d'optimisation, surtout lorsque celles-ci sont dédiées à un problème particulier. Malgré cela, nous avons montré une certaine supériorité par rapport à des algorithmes classiques [Basseur *et al.*, 2012a]. L'étude porte principalement sur la sensibilité de l'algorithme au paramétrage, dans le but de pouvoir identifier s'il est possible de fixer de manière efficace ces paramètres.

Le protocole expérimental suit globalement la description donnée dans la section 2.3. 20 exécutions ont été réalisées par instance et par méthode, puis la différence d'hypervolume moyenne est calculée par rapport à l'ensemble de référence construit à partir de l'union de tous les résultats. Une étude statistique, basée sur le test de Mann-Whitney à ensuite été réalisée. Dans les tables, les valeurs en gras indiquent les méthodes qui ne sont pas statistiquement dominées par celle obtenant la meilleure (plus petite) différence d'hypervolume moyenne.

Enfin, afin de visualiser graphiquement les résultats de manière globale, nous utilisons le principe d'*attainment function* [Zitzler *et al.*, 2003]. Pour une méthode donnée, cette fonction calcule la probabilité qu'au moins une solution de l'algorithme domine un point donné. En appliquant cela à l'espace des objectifs, on peut visualiser la zone de l'espace de recherche qui est atteinte avec un certain degré de certitude.

7.3.4 Résultats

L'étude est divisée en trois parties, correspondant au paramétrage de l'indicateur de qualité, de la stratégie d'initialisation des populations et de la taille de population. Les résultats sur le problème de flow shop se trouvent respectivement dans les tables 7.1, 7.2 et 7.3. L'analyse fournie ici ne considère que les aspects principaux des résultats, une analyse plus approfondie peut être trouvée dans [Basseur *et al.*, 2012a].

Performance des indicateurs de qualité

Cinq indicateurs binaires ont été testés : I_ϵ et I_{HD} proposés par Zitzler et Künzli, ainsi que I_{Ben}, I_{Sri} et I_{Fon} issus de méthodes de *ranking* classique de la littérature basés sur la dominance de Pareto. Nous avons donc cinq variantes de IBMOLS que nous appellerons respectivement $IBMOLS_\epsilon$, IBMOLS$_{HD}$, IBMOLS$_{Ben}$, IBMOLS$_{Sri}$ et IBMOLS$_{Fon}$.

Très succinctement, la table 7.1 montre que les indicateurs binaires de qualité n'utilisant pas directement la dominance de Pareto sont globalement plus performants, surtout lorsque la taille de l'espace de recherche est grande (factorielle du premier nombre figurant sur les instances). De plus, IBMOLS$_\epsilon$ tend à être meilleur que IBMOLS$_{HD}$. L'avantage principal des indicateurs IBMOLS$_\epsilon$ et IBMOLS$_{HD}$ est qu'ils sont capable de quantifier la dominance au lieu de n'utiliser qu'une valeur binaire comme le font les approches Pareto. Pour illustrer graphiquement les performances des algorithmes, les résultats empiriques de l'*attainment function* pour l'instance $ta\_50\_20\_01$ se trouvent dans la figure 7.3.

Indicateur	I_ϵ	I_{HD}	I_{Ben}	I_{Sri}	I_{Fon}
$ta\_20\_5\_01$	0,005	0,077	0,117	0,009	**0,002**
$ta\_20\_5\_02$	0,070	0,062	0,097	0,020	**0,010**
$ta\_20\_10\_01$	**0,002**	0,004	0,045	0,010	0,004
$ta\_20\_10\_02$	**0,018**	0,021	0,075	0,024	0,022
$ta\_20\_20\_01$	**0,001**	0,011	0,045	0,007	0,004
$ta\_50\_5\_01$	**0,009**	0,059	0,271	0,076	0,034
$ta\_50\_10\_01$	**0,055**	0,089	0,341	0,151	0,099
$ta\_50\_20\_01$	**0,058**	0,077	0,349	0,182	0,111

TABLE 7.1 – Comparaison des indicateurs binaires de qualité : taille de population 10, sauf les 2 dernières instances (20) ; initialisation : marche aléatoire pratiquée sur des solutions de l'archive (nombre de pas : $0{,}3 * n$).

Méthodes d'initialisation des populations

Trois définitions différentes de la fonction $genPop$ utilisée dans l'algorithme 7.3 ont été étudiées :

- *Rand* : Les N solutions initiales sont générées aléatoirement.
- *Cro* : On réalise une opération de recombinaison via $2N$ parents sélectionnés aléatoirement dans l'archive $\mathbf{PO}$, chaque solution étant sélectionnée une fois au maximum. Si la taille de $\mathbf{PO}$ est insuffisante, tout l'ensemble est sélectionné et les parents manquants sont générés aléatoirement. L'opérateur de recombinaison utilisé est un croisement deux points, utilisé dans nos anciens travaux sur les algorithmes évolutionnaires [Basseur *et al.*, 2002].
- *RM* : On sélectionne N solutions aléatoirement dans l'archive de la même manière que pour *Cro*. On applique ensuite sur ces solutions des mouvements aléatoires (nombre de pas : $\alpha\% * n$). Si besoin, l'archive est complétée par des solutions aléatoires.

La table 7.2 résume les résultats obtenus par l'algorithme IBMOLS itéré, dont les populations sont initialisées par chacune de ces trois méthodes. Pour la méthode *RM*, plusieurs taux de mouvements aléatoires ont été testés : 5%, 10%, 20%, 30% et 50% (de la taille de l'instance, en jobs).

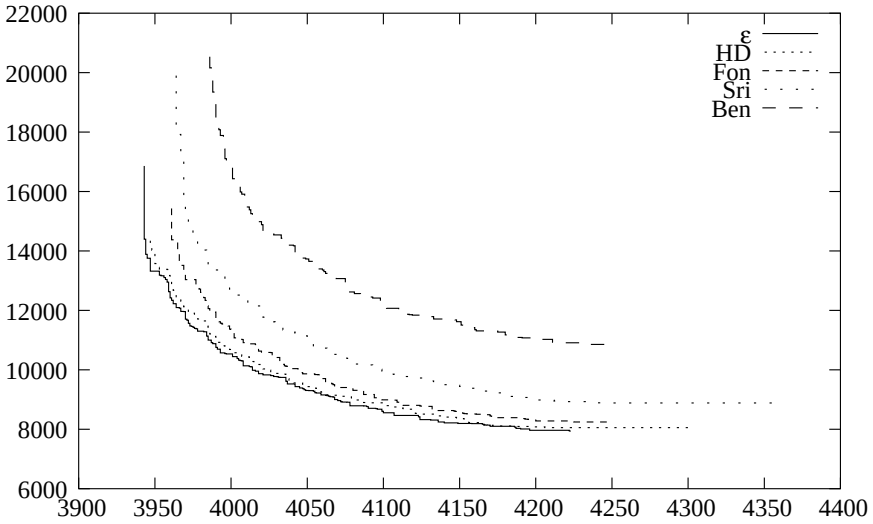


FIGURE 7.3 – Zone de l'espace objectif dominée par au moins 90% des exécutions réalisées pour chaque indicateur de qualité : exemple sur l'instance $ta\_50\_20\_01$ (abscisses : C_{max} ; ordonnées : T).

Les meilleurs résultats moyen sont toujours obtenus par la méthode *RM*, quel que soit le taux de mouvements utilisé (sauf $ta\_20\_5\_01$ et un taux de mouvements de 5%). Malgré tout, un taux de mouvements aléatoires relativement bas (10% par exemple) semble être plus adapté pour atteindre de meilleurs résultats.

Initialisation	RM					Rand	Cro
taux RM	5%	10%	20%	30%	50%		
$ta\_20\_5\_01$	0,063	0,018	**0,007**	0,011	0,009	0,039	0,024
$ta\_20\_5\_02$	0,087	0,039	**0,011**	0,013	0,018	0,129	0,140
$ta\_20\_10\_01$	0,005	0,006	0,004	0,004	**0,003**	0,007	0,014
$ta\_20\_10\_02$	0,024	**0,021**	0,030	0,028	0,028	0,035	0,063
$ta\_20\_20\_01$	0,010	0,003	**0,002**	0,003	0,003	0,013	0,060
$ta\_50\_5\_01$	**0,024**	0,029	0,050	0,061	0,079	0,122	0,117
$ta\_50\_10\_01$	0,080	**0,066**	0,087	0,102	0,114	0,169	0,263
$ta\_50\_20\_01$	0,081	**0,081**	0,107	0,115	0,136	0,174	0,134

TABLE 7.2 – Comparaison des fonctions d'initialisation des populations pour IBMOLS$_\epsilon$: taille de population 10, sauf les 2 dernières instances (20).

Taille de la population

Différentes tailles de populations ont été testées et la table 7.3 résume les résultats obtenus pour des tailles variant de 3 à 50 individus. Ces tailles sont très petites par rapport à ce que l'on trouve généralement dans les métaheuristiques à base de populations de solutions. En effet, comme le montre la table, les meilleurs résultats sont atteints pour des très petites tailles de population : de 3 à 20 individus, ce nombre étant relativement croissant en fonction de la taille de l'espace de recherche considéré. Il semble possible de fixer au préalable une taille en fonction de n et m ($\frac{n.m}{50}$

par exemple).

Taille population	3	5	8	10	15	20	30	50
$ta\_20\_5\_01$	**0,000**	0,001	0,036	0,035	0,055	0,028	0,057	0,128
$ta\_20\_5\_02$	0,014	**0,004**	0,019	0,037	0,046	0,076	0,073	0,106
$ta\_20\_10\_01$	0,009	**0,004**	0,004	0,005	0,005	0,008	0,013	0,016
$ta\_20\_10\_02$	0,031	0,027	**0,021**	0,023	0,027	0,035	0,037	0,051
$ta\_20\_20\_01$	0,005	0,003	**0,002**	0,004	0,026	0,052	0,053	0,063
$ta\_50\_5\_01$	0,017	0,018	**0,015**	0,020	0,033	0,048	0,068	0,086
$ta\_50\_10\_01$	0,147	0,102	**0,081**	0,083	0,087	0,095	0,134	0,184
$ta\_50\_20\_01$	0,165	0,117	0,093	0,085	0,090	0,089	**0,080**	0,103

TABLE 7.3 – Comparaison des tailles de population pour IBMOLS$_\epsilon$. Initialisation : marche aléatoire pratiquée sur des solutions de l'archive (nombre de pas : $0{,}3n$).

Pour conclure, il semble que les trois paramètres puissent être fixés préalablement en fonction des caractéristiques du problème. Cela s'est avéré aussi pour les autres problèmes étudiés. Ce qui fait que l'algorithme IBMOLS peut être facilement paramétrable et applicable à de nouvelles instances. Pour appliquer l'algorithme IBMOLS au traitement de nouveaux problèmes combinatoires multiobjectif, il semble cohérent de choisir prioritairement l'indicateur I_ϵ, une petite taille de population (10 individus par exemple) et d'itérer l'algorithme en initialisant les population par quelques mouvements aléatoires appliqués sur des solutions de l'archive. Sur le problème du flow shop, les résultats obtenus sont comparables à ceux obtenus lors de mon doctorat. Or, les temps d'exécution varient ici de 20 secondes à 20 minutes [Basseur *et al.*, 2012a], tandis que les résultats de mes travaux antérieurs, basés sur une approche Pareto, étaient obtenus par une version parallèle d'un algorithme complexe et plus sensible au paramétrage, dont les temps d'exécution étaient bien plus élevés. Malgré l'avancée de la performance des machines sur cette période, l'algorithme IBMOLS dispose de nombreux avantages : performance, simplicité, généricité et peu voire pas de paramètre à définir. De plus, dans [Basseur *et al.*, 2012a], nous avons montré son efficacité sur divers problèmes d'optimisation combinatoire multiobjectif : ordonnancement, routing+affectation et d'emploi du temps. Pour ces raisons, j'ai donc continué par la suite à m'intéresser aux approches de type recherches locales itérées basées sur les indicateurs de qualité.

Chapitre 8

Recherche locale multiobjectif basée sur l'hypervolume de dominance

Dans le chapitre précédent, nous avons montré l'intérêt de l'algorithme IBMOLS pour résoudre divers problèmes d'optimisation. Dans ce chapitre, nous nous intéressons toujours à l'optimisation basée sur les indicateurs de qualité, mais en nous focalisant sur l'hypervolume de dominance pour évaluer la qualité des solutions d'une approximation. L'indicateur d'hypervolume s'est globalement imposé comme indicateur d'évaluation des algorithmes d'optimisation multiobjectif, c'est donc naturellement que l'algorithme HBMOLS, inspiré de l'algorithme IBMOLS et basé sur l'hypervolume de dominance, a été proposé, tant ce choix parait judicieux pour l'optimisation multiobjectif en général. Dans ce chapitre, l'algorithme HBMOLS est d'abord décrit, avant de discuter des méthodes de calcul exactes et approchées de l'hypervolume de dominance. Enfin, je présenterai quelques résultats expérimentaux.

8.1 HBMOLS

Les expérimentations réalisées sur l'algorithme IBMOLS ont montré une relative supériorité de l'indicateur binaire I_ϵ sur l'indicateur binaire I_{HD}. Cela peut paraître surprenant, car l'hypervolume de dominance est un indicateur plus pertinent que l'ϵ-dominance pour évaluer la qualité d'approximations. Ce résultat vient principalement du fait que la combinaison des indicateurs binaires $I_{HD}(x,y)$ pour chaque $y \in \mathbf{P}$ aboutit à une valeur ne correspondant pas du tout à l'apport de x en termes d'hypervolume (voir figure 8.1). En particulier, I_{HD} dépend énormément du placement du point de référence qui influence en grande partie les valeurs calculées.

En se replaçant dans un contexte de sélection des algorithmes d'optimisation, l'objectif est en réalité de minimiser la perte, en termes de qualité de l'approximation, occasionnée par la suppression d'une solution de cette approximation. Si l'on considère que l'on souhaite maximiser l'hypervolume de dominance formé par l'approximation, on cherche alors à minimiser la perte d'hypervolume engendrée par la suppression d'une solution de l'approximation. Pour cela, il convient de définir $HypC$, correspondant à la contribution de x à l'hypervolume formé par l'ensemble de solutions $\mathbf{P}$, c'est-à-dire l'hypervolume dominé par x et par aucune autre solution de $\mathbf{P}$ (voir la

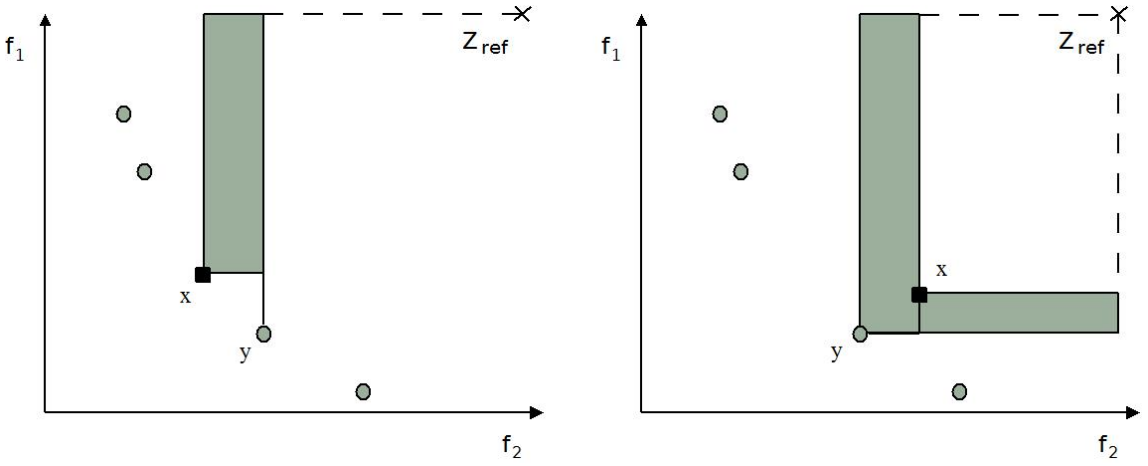


FIGURE 8.1 – $I_{Hyp}(x,\mathbf{P})$: Différence d'hypervolume minimale entre x et y, $y \in \mathbf{P}$ (aire de la partie grisée). A gauche : cas d'une solution non dominée. A droite : cas d'une solution dominée (valeur négative).

figure 8.2 – partie gauche – et l'équation 8.1).

$$HypC(x,\mathbf{P}) = HD(\mathbf{P} \setminus \{x\},\mathbf{P}) = Hyp(\mathbf{P},Z_{ref}) - Hyp(\mathbf{P} \setminus \{x\},Z_{ref}) \tag{8.1}$$

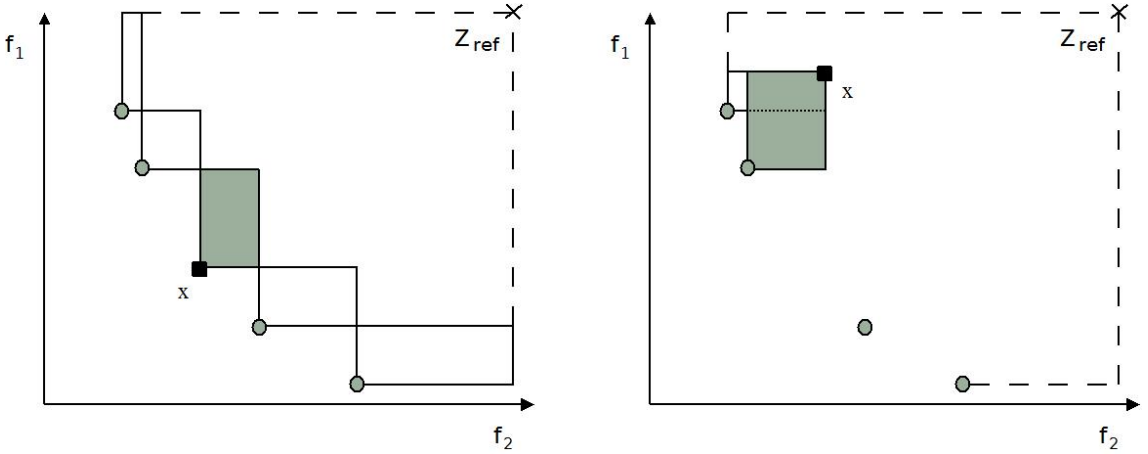


FIGURE 8.2 – $HC(x,\mathbf{P})$: Contribution de x à l'hypervolume de dominance de $\mathbf{P}$ (aire de la partie grisée). À gauche, $HC(x,\mathbf{P}) = HypC(x,\mathbf{P})$. À droite, cas d'une solution dominée (valeur négative, voir équation 8.2).

Lorsque l'ensemble des solutions contient des solutions dominées, alors leur contribution à l'hypervolume global est nulle. $HypC$ ne permet donc pas de comparer les solutions dominées entre elles. Afin de remédier à ce problème et de favoriser les solutions légèrement dominées par rapport aux solutions de très mauvaise qualité, nous avons proposé l'indicateur HC défini dans l'équation 8.2 et illustré dans la figure 8.2.

$$HC(x,\mathbf{P}) = \begin{cases} HypC(x,\mathbf{P}) & \text{si } x \text{ est non dominée} \\ -max_{y \in \mathbf{P}, y \succ x}\,(VOL([y,x])) & \text{sinon} \end{cases} \tag{8.2}$$

Algorithme 8.1 HBMOLS

Entrée : $\mathbf{P}$ (population initiale de taille N)
Sortie : $\mathbf{A}$ (approximation de l'ensemble de Pareto)
étape 1 : $\mathbf{A} \leftarrow ND(\mathbf{P})$
étape 2 - assignation initiale de l'efficacité : $\forall x \in \mathbf{P}$,*Fit*$(x) = HC(x,\mathbf{P})$.
étape 3 - pas de recherche locale : $\forall x \in \mathbf{P}$ faire :
répéter
1) $x^* \leftarrow$ voisin inexploré de x
2) $\mathbf{P} \leftarrow \mathbf{P} \cup x^*$
3) évaluation du fitness de x^* : $HC(x,\mathbf{P})$
4) mise à jour des fitness de $z \in \mathbf{P}$: *Fit*$(z) = HC(z,\mathbf{P})$
5) $\omega \leftarrow$ pire solution de $\mathbf{P}$
6) supprimer ω de $\mathbf{P}$
7) mise à jour des autres fitness de $z \in \mathcal{N}_{Obj}(x,\mathbf{P})$: *Fit*$(z) = HC(z,\mathbf{P})$
jusqu'à ce que tous les voisins de x soient explorés ou si $\omega \neq x^*$ (amélioration de $\mathbf{P}$)
$\mathbf{A} \leftarrow$ ND($\mathbf{A} \bigcup \mathbf{P}$)
étape 4 - critère d'arrêt : Si $\mathbf{P}$ n'a pas été modifiée, alors retourner $\mathbf{A}$; sinon retourner à l'étape 3.

Le fonctionnement de l'algorithme HBMOLS est très similaire au fonctionnement de l'algorithme IBMOLS, les deux ne différant que sur le processus d'affectation de la qualité des solutions. Les étapes de l'algorithme HBMOLS sont détaillées dans l'algorithme 8.1. L'utilisation de l'hypervolume de dominance permet néanmoins de d'améliorer et/ou de simplifier certaines étapes qui étaient réalisées dans l'algorithme IBMOLS. Ces aspects sont discutés dans ce qui suit.

8.1.1 Calcul et mise à jour du fitness des solutions

À la différence des indicateurs binaires de qualité utilisés dans les algorithmes IBMOLS et IBEA (I_ϵ et I_{HD}), pour l'algorithme HBMOLS, l'ajout ou la suppression d'une solution dans une approximation n'implique pas de recalculer l'ensemble des fitness des solutions de l'approximation.

En effet, la contribution à l'hypervolume d'une solution ne dépend que de ses voisines dans l'espace des objectifs. Deux solutions sont voisines dans l'espace des objectifs si aucune autre solution ne se trouve entre elles, c'est-à-dire si aucune autre solution ne se trouve strictement dans l'hypercube formé par ces deux solutions dans l'espace objectif :

Définition 16 (Solutions voisines dans l'espace objectif) *Soit* $\mathbf{P} \in 2^{\mathcal{X}}$, $x,y \in \mathbf{P}^2$ *deux solutions de* $\mathbf{P}$. *Soit* b *le point idéal et* w *le point nadir formé par* $\{x,y\}$. x *et* y *sont* voisines dans l'espace objectif *si et seulement si* $\forall z \in \mathbf{P}, z \not\succ w \wedge b \not\succ z$. *On notera alors* $x \in \mathcal{N}_{obj}(y,\mathbf{P})$ *(cela implique que* $y \in \mathcal{N}_{obj}(x,\mathbf{P})$*).*

Dans le cas de l'optimisation biobjectif, ce voisinage ne contient que deux solutions au maximum (une seule dans le cas d'une solution extrême). À partir de trois objectifs, le nombre de voisins dans l'espace objectif n'est pas borné et peut dans le cas extrême contenir toute l'approximation.

8.1.2 Normalisation des fonctions objectif

L'étape de normalisation des objectifs n'apparaît pas dans l'algorithme IBMOLS. En effet, l'indicateur HC n'est pas sensible à l'échelle utilisée pour les objectifs : si l'on multiplie les valeurs d'un objectif par une valeur α, tous les fitness HC calculés sont également multipliés par α, et la relation d'ordre résultant reste inchangée. Ceci est valable quel que soit le nombre d'objectifs considérés.

La suppression de cette étape n'est pas anodine, car dans l'algorithme IBMOLS, cette normalisation était susceptible d'être recalculée à chaque ajout ou suppression d'une solution de la population, puisque le facteur utilisé dépend du point nadir et du point idéal de l'approximation.

8.1.3 Coordonnées du point de référence

Le mécanisme de sélection de HBMOLS dépend en partie du choix du point de référence Z_{ref}. Plus précisément, les coordonnées de Z_{ref} sont déterminantes pour le fitness des solutions extrêmes de l'approximation. Afin de garder un ensemble de solutions offrant des compromis les plus divers possibles, il est indispensable d'éviter de supprimer les solutions extrêmes, afin de ne pas réduire l'étendue de de l'approximation dans l'espace objectif. Pour cela, il suffit de définir les coordonnées de Z_{ref} à $(+\infty, \ldots, +\infty)$, ce qui donne un fitness maximal à chaque solution ayant la meilleure valeur sur un objectif donné. Cela équivaut à exclure ces solutions du processus de suppression de $HBMOLS$. Par conséquent, Z_{ref} n'est pas un paramètre en soit.

8.1.4 Critère d'arrêt

Pour l'algorithme IBMOLS, l'indicateur utilisé ne vérifie pas nécessairement la propriété de transitivité de la relation d'ordre entre les approximations. La conséquence principale est qu'il est possible de "boucler" entre plusieurs approximations, et ainsi ne jamais atteindre un optimum local (au sens de l'indicateur utilisé) : par exemple un indicateur peut évaluer A_1 meilleure que A_2, A_2 meilleure que A_3 et A_3 meilleure que A_1. Dans ce cas, le mécanisme de sélection peut très bien passer indéfiniment entre A_1, A_2 et A_3. Pour cette raison, l'algorithme IBMOLS s'arrête lorsque un pas complet de l'algorithme n'a pas permis d'ajouter une solutions dans l'archive (voir étape 4 de l'algorithme 7.2). Pour HBMOLS, ce problème n'apparait pas (en particulier parce que l'hypervolume vérifie la relation de transitivité), ce qui permet d'arrêter lorsque l'algorithme atteint effectivement un optimum local (aucune solution n'a pu être modifiée sur un pas complet de recherche).

8.2 Calcul de l'hypervolume de dominance

L'inconvénient majeur d'HBMOLS réside dans la difficulté du calcul de l'hypervolume de dominance ainsi que les indicateurs associés $HypC$ et HC. D'une manière générale, le calcul de l'hypervolume de dominance a été montré comme étant #P-difficile, c'est à dire que tout algorithme le calculant nécessite un temps super-polynomial en fonction du nombre d'objectifs considérés [Bringmann et Friedrich, 2008] (sauf si $P = NP$). De même, ces auteurs ont montré que même l'approximation de la contribution à l'hypervolume d'une solution est NP-difficile

[Bringmann et Friedrich, 2009].

La complexité algorithmique de HBMOLS repose donc en grande partie sur le calcul de l'hypervolume de dominance, d'autant plus lorsque le nombre d'objectifs considéré est élevé. Nous nous sommes donc intéressés aux algorithmes de calcul exact de l'hypervolume de dominance, mais également aux algorithmes d'approximation. Ces aspects ont été abordés lors de la thèse de Rong-Qiang Zeng et le stage de master recherche d'Arthur Chambon.

8.2.1 Calcul exact de l'hypervolume de dominance

Les principaux algorithmes proposés pour calculer rapidement l'hypervolume de dominance sont :

- IEA (*Inclusion–Exclusion Algorithm*) [Wu et Azarm, 2001],
- LebMeasure (*Lebesgues Measure Algorithm*) [Fleischer, 2003],
- HSO (*Hypervolume by Slicing Objectives*) [While *et al.*, 2006],
- HOY (*Hypervolume by Overmars and Yap*) [Beume, 2009],
- IIHSO (*Iterated Incremental HSO*) [Bradstreet *et al.*, 2010],
- WFG (*Walking Fish Group*) [While *et al.*, 2012].

Les expériences réalisées sur des populations de vecteurs objectifs aléatoires mutuellement non dominés, ont montré que l'algorithme HSO était globalement le plus efficace, même s'il n'est pas le plus rapide sur toutes les instances testées et que sa complexité dans le pire des cas n'est pas la meilleure parmi les algorithmes existants. Dans le cadre de la procédure de sélection de l'algorithme HBMOLS, ce n'est pas l'hypervolume de dominance que l'on cherche à calculer, mais la contribution de chaque solution à cet hypervolume. Une méthode simple pour y arriver est de calculer l'hypervolume total, puis de lui ôter l'hypervolume formé par la population moins la solution dont on cherche à calculer la contribution ($HypC(x) = Hyp(\mathbf{P}) - Hyp(\mathbf{P} \setminus x)$).

Durant le stage d'Arthur Chambon, un algorithme exact de calcul de la contribution à l'hypervolume de dominance a été proposé. Cet algorithme, CHSO, s'inspire de l'algorithme HSO mais calcule les contributions de manière plus efficace que la méthode de base. Globalement, l'algorithme s'appuie sur la remarque que l'hypervolume exclusif dominé par un point ne dépend pas toujours de tous les autres points de la population, mais uniquement de ses voisins dans l'espace objectif. On a alors $HypC(x) = Hyp(\mathcal{N}_{obj}(x,\mathbf{P}) \cup \{x\}) - Hyp(\mathcal{N}_{obj}(x,\mathbf{P}))$.

L'algorithme s'appuie sur cette propriété afin de déterminer de manière plus efficace l'hypervolume exclusif des solutions de la population. Des précisions concernant l'algorithme proposé sont données dans le rapport de stage d'Arthur Chambon. Nous avons donc testé CHSO pour évaluer différents ensembles de solutions mutuellement non dominées (si l'ensemble de solutions contient des solutions dominées, la sélection réalisée par HBMOLS est alors beaucoup moins complexe à calculer).

Les résultats pour des ensembles de solutions de 10 à 500 individus et de 3 à 13 objectifs sont donnés dans la table 8.1. Si certains résultats n'apparaissent pas, cela indique que le temps de réponse de l'algorithme était trop long. L'algorithme CHSO est comparé aux algorithmes HSO et WFG, qui sont particulièrement efficaces lorsque le nombre d'objectifs est grand. CHSO est globalement beaucoup plus rapide que HSO et est bien plus performant que WFG dans le cas à 3 objectifs. CHSO est plutôt performant pour 4 et 5 objectifs où les temps de calcul sont comparables avec ceux de WFG. Au delà de 5 objectifs, WFG devient nettement plus rapide.

Nb objectifs	Nb solutions	HSO	WFG	CHSO
3	10	0,2"	0,2"	**0,1"**
3	25	4,2"	1,1"	**0,4"**
3	50	1'4"	9"	**2"**
3	100	19'19"	1'22"	**16"**
3	150	116'1"	5'25"	**55"**
3	200	-	16'12"	**2'16"**
3	500	-	477'59"	**39'44"**
4	10	0,5"	0,3"	**0,2"**
4	25	29"	3"	**2"**
4	50	14'16"	26"	**15"**
4	100	501'27"	3'57"	**2'27"**
4	150	4450'	13'	**10'**
4	200	-	34'	**33'**
4	500	-	-	**1275'**
5	10	1,5"	**0,2"**	0,5"
5	25	2'51,5"	**7,0"**	7,3"
5	50	-	1'49"	**1'44"**
5	100	-	**24'**	28'
5	150	-	**94'**	198'
7	10	8,8"	**0,3"**	3,4"
7	25	60'48"	**35"**	3'7"
7	50	-	**30'**	80'
10	10	1'19,4"	**0,5"**	31,4"
10	25	-	**3'**	196'
10	50	-	**391'**	-
13	10	7'45,3"	**0,8"**	4'19,4"
13	25	-	**13'**	-

TABLE 8.1 – Temps de calcul de contribution à l'hypervolume de l'ensemble d'une population de solutions mutuellement non dominées.

8.2.2 Calcul approché de l'hypervolume de dominance

Le calcul de l'hypervolume exclusif de solutions est très coûteux lorsque le nombre d'objectifs dépasse 3, surtout que ce calcul est réalisé très souvent durant la recherche. Dans le cas où ce temps de calcul devient grand par rapport au coût de la fonction d'évaluation des solutions, il semble cohérent de se restreindre à une approximation de HC afin de ne pas ralentir la recherche.

Nous avons mis au point un algorithme d'approximation de HC consistant à calculer la contribution en se basant uniquement sur un nombre restreint de voisins dans l'espace des objectifs (voir définition 16), ce qui permet de réduire la complexité du calcul. Les résultats, présentés dans [Zeng, 2012], montrent qu'une telle approche peut être bénéfique, la perte de précision quant à la finesse de l'évaluation de la qualité des solutions étant compensée par le gain en temps de calcul. Dans le cadre du master recherche d'Arthur Chambon, des indicateurs de qualité ont été comparés avec HC afin de déterminer s'ils permettaient de classer les solutions entre elles de manière similaire (sans forcément chercher à approximer les *valeurs* calculées par HC) .

8.3 Résultats expérimentaux

Nous avons réalisé de nombreuses expérimentations pour évaluer l'aptitude d'HBMOLS à approximer les ensembles Pareto optimaux de problèmes d'optimisation combinatoire multiobjectif.

Premièrement, nous l'avons naturellement comparé à IBMOLS puisque le principe de base de ces deux algorithmes est le même. Nous avons éprouvé HBMOLS et IBMOLS sur des instances de QAP et de Flow Shop à deux, trois et quatre objectifs, afin d'observer les limites de HBMOLS en termes de temps de calcul — étant donné que le calcul de l'hypervolume devient complexe lorsque le nombre d'objectifs augmente.

Nous avons ensuite comparé l'algorithme HBMOLS à des métaheuristiques multiobjectif classiques de la littérature, sur différents problèmes de flow shop biobjectif où nous avons fait varier les fonctions objectif optimisées.

8.3.1 Comparaison IBMOLS / HBMOLS

Paramètres

Un des intérêts majeurs de ces deux algorithmes est qu'ils nécessitent un petit nombre de paramètres, pouvant être fixés de manière simple sans remettre en jeu l'efficacité de la recherche. Globalement, il s'agit d'ajuster ces paramètres en fonction de la taille de l'instance étudiée.

- Temps d'exécution : Pour une instance à n jobs et m machines et considérant N objectifs optimisés, le temps en T des algorithmes (en secondes) est défini comme suit :

$$T = \frac{n^2 \times m \times N}{100} \tag{8.3}$$

- Taille de la population : Nous avons montré dans le chapitre précédent qu'IBMOLS était particulièrement efficace lorsque la taille de la population est petite, mais que la taille optimale tend à augmenter légèrement avec la taille de l'instance. Au vu de ces observations, nous avons défini les tailles comme suit :

$$|P| = \begin{cases} 10 & \text{si} \quad 0 < n \times m < 500 \\ 20 & \text{si} \quad 500 \leqslant n \times m < 1000 \\ 30 & \text{si} \quad 1000 \leqslant n \times m \end{cases} \tag{8.4}$$

- Initialisation des populations : Nous nous sommes encore une fois appuyés sur les résultats du chapitre précédent. Chaque solution initiale est générée en choisissant une solution non sélectionnée de l'archive A, à laquelle on applique $0{,}3 \times n$ mouvements aléatoires. Si l'archive ne contient pas assez de solutions, on complète la population avec des solutions aléatoires.
- Point de référence Z_{ref} : IBMOLS$_{HD}$ et HBMOLS requièrent la définition d'un point de référence. Celui-ci est défini de telle sorte que les solutions non dominées extrêmes soient systématiquement conservées durant la recherche. Pour cela, il suffit de définir Z_{ref} comme étant le plus *mauvais* possible (idéalement, dans un contexte de minimisation des objectifs, l'ensemble des coordonnées doivent être définies à $+\infty$).

Enfin, comme précédemment, les individus sont codés sous forme de permutations, et l'opérateur de voisinage utilisé est l'opérateur d'insertion pour le flow shop, et l'opérateur d'échange

pour le QAP. Le protocole expérimental utilisé est similaire à celui utilisé lors de nos précédentes expérimentations et est détaillé dans [Basseur *et al.*, 2012b].

Résultats

Les instances de flow shop et de QAP sont celles utilisées auparavant. Afin de réaliser des expérimentations sur plusieurs objectifs, en évitant d'avoir une corrélation trop forte entre les objectifs, nous avons choisi de créer artificiellement des instances multiobjectif en combinant plusieurs instances de mêmes tailles (voir [Basseur *et al.*, 2012b]). Nous avons réalisé des expérimentations sur le Flow Shop et le QAP, pour un nombre d'objectifs variant de deux à quatre. L'algorithme de calcul de l'hypervolume utilisé, en particulier pour trois objectifs ou plus, est détaillé dans l'article [Basseur *et al.*, 2012b].

Instance	IBMOLS$_\epsilon$	IBMOLS$_{HD}$	HBMOLS
20_05_01	0,0019	0,0019	**0,0006**
20_10_01	0,0024	0,0014	**0,0005**
20_15_01	0,0085	0,0094	**0,0023**
20_20_01	0,0010	0,0011	**0,0001**
30_05_01	0,0554	0,0534	**0,0165**
30_10_01	0,0644	0,0724	**0,0350**
30_15_01	0,0396	0,0396	**0,0202**
30_20_01	0,0351	0,0385	**0,0188**

Instance	IBMOLS$_\epsilon$	IBMOLS$_{HD}$	HBMOLS
50_05_01	0,0556	0,0606	**0,0367**
50_10_01	0,0943	0,0910	**0,0567**
50_15_01	0,0940	0,0987	**0,0700**
50_20_01	0,1085	0,1101	**0,0916**
70_05_01	0,1571	0,1611	**0,0756**
70_10_01	0,0885	0,1214	**0,0601**
70_15_01	0,1024	0,1573	**0,0801**
70_20_01	0,1049	0,1510	**0,0736**
100_05_01	0,1685	0,1812	**0,0828**

TABLE 8.2 – Comparaison entre IBMOLS et HBMOLS sur le flow shop à deux objectifs.

Instance	IBMOLS$_\epsilon$	IBMOLS$_{HD}$	HBMOLS
20_05_01+20_05_02	0,0088	0,0171	**0,0064**
20_10_01+20_10_02	0,0449	0,0724	**0,0132**
20_15_01+20_15_02	0,0461	0,0522	**0,0176**
20_20_01+20_20_01	0,0457	0,0470	**0,0144**
30_05_01+30_05_02	0,0492	0,0602	**0,0401**
30_10_01+30_10_02	0,1200	0,1334	**0,0733**
30_15_01+30_15_02	0,1090	0,1147	**0,0486**
30_20_01+30_20_02	0,1133	0,1393	**0,0768**

Instance	IBMOLS$_\epsilon$	IBMOLS$_{HD}$	HBMOLS
50_05_01+50_05_02	0,1336	0,1233	**0,0402**
50_10_01+50_10_02	0,1200	0,1424	**0,1142**
50_15_01+50_15_02	0,1242	0,1476	**0,0923**
50_20_01+50_20_02	0,1442	0,1397	**0,1212**
70_05_01+70_05_02	0,0824	0,0612	**0,0171**
70_10_01+70_10_02	0,1167	0,1467	**0,0803**
70_15_01+70_15_02	0,1495	0,1759	**0,1404**
70_20_01+70_20_02	0,1328	0,1563	**0,1121**
100_05_01+100_05_02	0,1082	0,1169	**0,0511**

TABLE 8.3 – Comparaison entre IBMOLS et HBMOLS sur le flow shop à trois objectifs.

Instance	IBMOLS$_\epsilon$	IBMOLS$_{HD}$	HBMOLS
20_05_01+20_05_02	**0,0319**	0,0441	0,0435
20_10_01+20_10_02	**0,0726**	0,1186	0,0882
20_15_01+20_15_02	**0,0701**	0,0917	0,0729
20_20_01+20_20_01	**0,0853**	0,1043	0,0866
30_05_01+30_05_02	**0,0899**	0,1061	0,1353
30_10_01+30_10_02	**0,1171**	0,1583	0,1463
30_15_01+30_15_02	**0,1213**	0,1351	0,1324
50_05_01+50_05_02	0,1434	0,1199	**0,1160**
70_05_01+70_05_02	0,1946	0,1504	**0,1275**

TABLE 8.4 – Comparaison entre IBMOLS et HBMOLS sur le flow shop à quatre objectifs.

Les tables 8.2, 8.3 et 8.4 donnent les résultats comparatifs pour le flow shop à deux, trois et quatre objectifs.

Les résultats sur les instances à deux objectifs (table 8.2) sont largement en faveur de HBMOLS, qui domine statistiquement IBMOLS$_{hyp}$ et IBMOLS$_{\epsilon}$ sur toutes les instances testées. De plus, la différence d'hypervolume moyenne de HBMOLS par rapport aux deux variantes d'IBMOLS est largement inférieure (parfois deux ou trois fois moindre). Cela montre clairement que l'indicateur de contribution à l'hypervolume de dominance est plus adapté que les indicateurs binaires pour évaluer l'intérêt des solutions pendant la recherche.

Les résultats sur les instances à trois objectifs (table 8.3) confirment les observations faites sur les instances à deux objectifs. HBMOLS domine statistiquement les deux variantes d'IBMOLS sur toutes les instances sauf 50_10_01 + 50_10_02. Les écarts entre les différences d'hypervolume moyennes calculées sont néanmoins moindres que pour les instances à deux objectifs.

La table 8.4 montre que pour les instances à quatre objectifs, HBMOLS ne domine plus statistiquement les variantes d'IBMOLS. HBMOLS est même dominé statistiquement sur cinq instances et est moins bon en moyenne que IBMOLS$_{\epsilon}$ sur les sept plus petites instances. L'efficacité de la sélection réalisée par HBMOLS n'est pas à remettre en cause dans ces résultats. Le problème vient exclusivement du temps de calcul nécessaire pour calculer des hypervolumes pour quatre objectifs. Cela montre les limitations de HBMOLS, qui ne peut être adapté directement pour les problèmes comportant beaucoup d'objectifs, car le temps d'évaluation de la qualité des solutions est clairement un frein à la convergence de l'algorithme.

Instance	IBMOLS$_{\epsilon}$	IBMOLS$_{HD}$	HBMOLS
chr_12_a+chr_12_b	0,0029	0,0030	**0,0022**
chr_15_a+chr_15_b	0,0144	**0,0127**	**0,0100**
chr_20_a+chr_20_b	0,0692	0,0644	**0,0502**
esc_16_a+esc_16_b	**0,0000**	**0,0000**	**0,0000**
esc_32_a+esc_32_b	0,1190	0,1196	**0,0873**
Lipa_30_a+Lipa_30_b	0,2202	0,2462	**0,1675**
Ste_36_a+Ste_36_b	0,8165	0,7260	**0,3235**

TABLE 8.5 – Comparaison entre IBMOLS et HBMOLS sur le QAP à deux objectifs.

Instance	IBMOLS$_{\epsilon}$	IBMOLS$_{HD}$	HBMOLS
chr_12_a+chr_12_b+chr_12_c	**0,0009**	**0,0015**	0,0016
chr_15_a+chr_15_b+chr_15_c	**0,0365**	0,0427	**0,0361**
chr_20_a+chr_20_b+chr_20_c	0,1145	0,1172	**0,1038**
esc_16_a+esc_16_b+esc_16_c	**0,00008**	0,00029	**0,00005**
esc_32_a+esc_32_b+esc_32_c	**0,1482**	**0,1359**	0,1606
Ste_36_a+Ste_36_b+Ste_36_c	0,9303	1,1682	**0,6182**

TABLE 8.6 – Comparaison entre IBMOLS et HBMOLS sur le QAP à trois objectifs.

Instance	IBMOLS$_{\epsilon}$	IBMOLS$_{HD}$	HBMOLS
esc_16_a+esc_16_b+esc_16_c+esc_16_d	**0,0048**	0,0076	0,0094
esc_32_a+esc_32_b+esc_32_c+esc_32_d	**0,1599**	0,2000	0,3246

TABLE 8.7 – Comparaison entre IBMOLS et HBMOLS sur le QAP à quatre objectifs.

Les tables 8.5, 8.6 et 8.7 contiennent les résultats obtenus sur le QAP à deux, trois et quatre objectifs.

Les résultats sur les instances à deux objectifs (table 8.5) sont en faveur de HBMOLS, qui est le

meilleur algorithme en moyenne sur toutes les instances testées. La dominance statistique est claire également, sauf pour l'instance $chr\_15\_a + chr\_15\_b$ où IBMOLS$_{HD}$ n'est pas dominé, ainsi que l'instance $esc\_16\_a + esc\_16\_b$ où les différences d'hypervolume moyennes sont toutes nulles (cela veut dire que les résultats obtenus sur l'ensemble des exécutions sont tous identiques dans l'espace objectif, ce qui semble indiquer qu'un ensemble Pareto optimal est systématiquement atteint).

Comme pour le flow shop, l'augmentation du nombre d'objectifs étudiés diminue l'efficacité de HBMOLS, mais de manière plus importante ici. Pour trois objectifs (table 8.6), les résultats sont très contrastés : IBMOLS$_\epsilon$ est meilleur en moyenne sur une instance, IBMOLS$_{HD}$ sur deux instances et HBMOLS sur quatre instances. En observant les résultats statistiques, cela parait encore moins clair. Pour quatre objectifs (table 8.7) HBMOLS est dominé statistiquement par IBMOLS$_\epsilon$ sur les deux instances testées. Il est d'ailleurs l'algorithme le moins bon en moyenne et de manière assez claire.

D'une manière générale, les résultats montrent que HBMOLS est très efficace, mais qu'il n'est pas adapté pour traiter des problèmes d'optimisation comportant beaucoup d'objectifs. Dans la suite de cette section, nous nous intéressons à la performance d'HBMOLS par rapport à des métaheuristiques multiobjectif classiques de la littérature dans le cas à deux objectifs.

8.3.2 Étude de différents algorithmes d'optimisation multiobjectif sur le problème de flow shop

Les résultats présentés dans cette section sont une version condensée des résultats présentés dans [Basseur et Liefooghe, 2013], qui comporte de plus d'autres informations concernant les méthodes testées et une analyse de la corrélation des objectifs optimisés. Nous nous sommes intéressés à la performance de HBMOLS par rapport à trois méthodes classiques de la littérature : deux algorithmes évolutionnaires (NSGA-II [Deb *et al.*, 2002] et IBEA [Zitzler et Künzli, 2004]) et un algorithme de recherche locale à base de population (PLS [Paquete *et al.*, 2004]). Nous avons étudié le comportement de ces approches de résolution sur différentes variantes de flow shop biobjectif (nous nous restreignons ici aux cas à deux objectifs), en particulier pour différentes paires d'objectifs parmi la date d'achèvement de l'ordonnancement, la somme des dates d'achèvement, le retard maximum, la somme des retards, ou encore le nombre de travaux en retard.

Un jeu d'instances a été proposé et une étude expérimentale sur le degré de corrélation entre ces différents critères a d'abord été menée. Les métaheuristiques étudiées ont été adaptées au problème de flow shop de permutation biobjectif avec dates de fins souhaitées. Une analyse expérimentale sur leur performance respective nous a permis de mettre en avant l'efficacité des approches de recherche locale à base de population et en particulier HBMOLS.

Dans la suite, nous commençons par présenter brièvement deux algorithmes classiques de l'optimisation multiobjectif : NSGA-II, IBEA (PLS à été présenté à la section 7.2.1). Puis, nous présentons les résultats expérimentaux obtenus sur différents problèmes de flow shop.

NSGA-II

NSGA-II [Deb *et al.*, 2002] est probablement l'algorithme évolutionnaire multiobjectif le plus utilisé comme algorithme de référence pour évaluer d'autres algorithmes. À chaque génération,

les solutions de la population courante sont classées en fonction de deux critères. Deux valeurs sont donc affectées à chaque membre de la population. Tout d'abord, la stratégie d'affectation des valeurs de fitness se base sur le compte de dominance et sur la dominance Pareto. Toutes les solutions non dominées de la population se voient affecter une valeur de fitness de 1 (premier front), puis elles sont retirées de la population. Toutes les solutions non dominées de la population se voient affecter une valeur de fitness de 2 (deuxième front), puis elles sont retirées de la population, et ainsi de suite. Ce processus est itéré jusqu'à ce que l'ensemble des solutions dont la valeur de fitness est à évaluer soit vide. Cette première valeur représente la qualité de la solution en termes de convergence. La deuxième valeur consiste à estimer la densité de solutions autour d'un point particulier de l'espace objectif à l'aide de la distance de *crowding*, et représente la qualité de la solution en termes de diversité. La distance de *crowding* est calculée parmi les solutions appartenant à un même front (c'est-à-dire ayant une valeur de fitness identique). Ainsi, une solution est considérée comme plus performante qu'une autre si elle a une meilleure valeur de fitness ou, en cas d'égalité, si elle a la meilleure distance de *crowding*. À chaque itération de l'algorithme évolutionnaire, N solutions *enfant* sont générées, où N est la taille de la population. La stratégie de sélection consiste en un tournoi déterministe réalisé entre deux solutions choisies aléatoirement. La phase de remplacement élitiste fonctionne comme suit. La population de l'itération précédente et la population *enfant* sont fusionnées, et seules les meilleures solutions survivent selon la procédure hiérarchique énoncée ci-dessus. Par ailleurs, dans la version originale de NSGA-II, aucune technique d'archivage n'est considérée, l'approximation courante de l'ensemble Pareto optimal étant contenue au sein de la population principale.

IBEA

L'algorithme IBEA a inspiré la conception de l'algorithme IBMOLS et dans une moindre mesure, l'algorithme HBMOLS. L'idée principale introduite par IBEA [Zitzler et Künzli, 2004] est d'instaurer un ordre total entre les solutions de la population courante en généralisant la relation de dominance au moyen d'un indicateur binaire de qualité arbitraire I. Cet indicateur représente l'objectif à optimiser durant le processus d'optimisation. Afin d'intégrer I au sein d'une stratégie d'affectation des valeurs de fitness, IBEA utilise une approche additive, basée sur une comparaison deux à deux des solutions de la population courante et qui amplifie l'influence des solutions non dominées sur les solutions dominées (voir section 7.1.2) :

$$F(x) = \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/\kappa} \tag{8.5}$$

où $\kappa > 0$ est un facteur d'échelle. La sélection pour la recombinaison se compose d'un tournoi binaire déterministe. La sélection pour le remplacement consiste à supprimer, itérativement, la plus mauvaise solution de la population courante jusqu'à ce que la taille requise soit atteinte ; les valeurs de fitness des individus restants étant mises à jour après chaque suppression.

Ainsi, n'importe quel indicateur binaire de qualité peut potentiellement être utilisé dans l'équation (8.5). À titre d'exemple, prenons l'indicateur ϵ-additif ($I_{\epsilon+}$), défini comme suit :

$$I_{\epsilon+}(z,z') = \max_{i \in \{1,\ldots,n\}} \{z_i - z'_i\} \tag{8.6}$$

Il donne la valeur minimale par laquelle un vecteur objectif $z \in Z$ doit être — ou peut être — translaté dans l'espace objectif pour faiblement dominer une autre solution $z' \in Z$. Notons que

cette translation peut prendre une valeur négative et que les fonctions objectif sont supposées être normalisées.

Résultats expérimentaux sur plusieurs variantes du flow shop

Afin d'évaluer les ordonnancements d'un problème de flow shop, un certain nombre de critères peuvent être définis. Le plus classique est le *makespan* ; la somme (ou moyenne) des retards est souvent étudiée également. Ici, nous nous intéressons aussi à d'autres critères considérés dans la littérature. Voici les cinq objectifs que nous avons considérés :

- C_{max} : Date d'achèvement de l'ordonnancement.
- C_{sum} : Somme des dates d'achèvement des jobs (équivalent à la date de fin d'ordonnancement moyenne des jobs).
- T_{max} : Retard maximum des jobs par rapport à leur date de fin au plus tard souhaitée.
- T_{sum} : Somme des retards des jobs par rapport à leur date de fin au plus tard souhaitée (équivalent au retard moyen des jobs).
- U_{sum} : Nombre de jobs en retard par rapport à leur date de fin au plus tard souhaitée.

Nous nous sommes intéressés ici aux problèmes biobjectif, en combinant de toutes les manières possibles les cinq objectifs présentés ci-dessus.

Nous avons mené une étude de corrélation des objectifs optimisés dont les résultats sont présentés dans [Basseur et Liefooghe, 2013]. Ceux-ci révèlent une corrélation positive entre chaque paire de critères. En particulier, la minimisation de la date d'achèvement de l'ordonnancement (C_{max}) s'avère fortement corrélée avec la minimisation de la date d'achèvement moyenne des jobs (C_{sum}). De façon plus surprenante, il existe également une très forte corrélation entre ces deux critères et la somme des retards des jobs (T_{sum}). Enfin, seule la minimisation conjointe du retard maximum des jobs (T_{max}) et du nombre de jobs en retard (U_{sum}) révèle une corrélation quasiment nulle. En conséquence, de telles formulations du problème de flow shop biobjectif n'engendrent qu'un nombre limité de solutions Pareto optimales, en comparaison à d'autres problèmes où les fonctions objectif sont plus conflictuelles (ce qui est bénéfique à l'algorithme PLS, qui ne peut contrôler la taille de l'ensemble courant des solutions non dominées).

Paramètrage Les approches présentées ici présentent l'avantage non négligeable de nécessiter très peu de paramètres. Ceux-ci sont discutés ci-dessous :

- NSGA-II : La taille de la population a été fixée à 100 individus, la littérature fixant généralement cette taille entre 100 et 200. L'opérateur de recombinaison utilisé est le même que pour le cadre mono-objectif [Basseur *et al.*, 2002].
- IBEA : La taille de la population a été fixée à 100 individus, afin d'avoir des conditions d'exécution similaires à celles de NSGA-II. De plus, la valeur de κ a été fixée à 0,001. L'impact de ce paramètre est limité, à partir du moment où l'on choisit une valeur proche de 0 (en général cette valeur est fixée à 0,01 ou 0,001).
- PLS : Il n'y a pas de paramètre primordial à fixer pour cet algorithme, la taille de la population étant variable. Néanmoins, la population initiale est générée à partir de 100 solutions aléatoires, parmi lesquelles on garde celles étant non dominées. Ce paramètre d'initialisation a un impact très limité sur les résultats.
- HBMOLS : La taille de la population a été fixée à 20 individus.

PLS et HBMOLS étant des recherches locales *itérées*, il convient définir comment initialiser une nouvelle population après une recherche locale. Pour cela, on choisit aléatoirement des solutions dans l'archive des solutions de ces algorithmes, sur lesquelles on applique quelques mutations aléatoires. Ce nombre de mutations a été fixé à 1/5 du nombre de jobs de l'instance testée.

Résultats La table 8.8 recense l'ensemble des résultats obtenus par les quatre algorithmes. La première colonne donne l'instance testée, identifiée par deux nombres référant au nombre de jobs et au nombre de machines. La deuxième colonne donne les deux objectifs optimisés. Enfin les colonnes suivantes indiquent la différence d'hypervolume moyenne obtenue par la méthode correspondante. Comme dans tout le manuscrit, les valeurs sont sur fond grisé lorsque l'algorithme correspondant n'est pas statistiquement moins bon que le meilleur algorithme (au sens de la distribution des valeurs de différence d'hypervolume), avec une valeur p supérieure à 95%. La meilleure valeur moyenne obtenue pour chaque instance apparait en gras.

L'analyse de la table 8.8 nous montre que l'algorithme HBMOLS obtient globalement les meilleurs résultats, et que PLS arrive souvent en deuxième position. Il semble en effet que les algorithmes de type recherche locale à base de population sont adaptés aux problèmes de flow shop étudiés ici. C'est un constat qui est souvent observé en optimisation mono-objectif, même si les algorithmes évolutionnaires peuvent néanmoins être plus performants lorsqu'ils sont hybridés avec des algorithmes de type recherche locale. Une analyse plus fine des résultats peut nous apporter des observations supplémentaires :

- Les résultats généraux sont semblables, quels que soient les objectifs étudiés. Ceci est en partie dû à l'aspect très générique des méthodes proposées, qui ont un comportement similaire dans les différents cas. De plus, les différentes fonctions objectif ont toutes une corrélation positive, ce qui a pour effet d'accentuer la similarité des résultats obtenus.
- PLS est efficace sur les petites instances, ce qui est intéressant vue la simplicité de cette approche. Cependant, sur des instances de plus grande taille, il faut lui préférer HBMOLS qui est globalement plus efficace.
- Les approches évolutionnaires (NSGA-II et IBEA) ne rivalisent que sur l'instance 50_05, qui se caractérise par un espace de recherche assez grand et un très petit nombre de solutions non dominées. On voit que, dans ce cas particulier, PLS et HBMOLS sont mis en difficulté. Dans ce cas, il serait intéressant d'appliquer une méthode mono-objectif (en combinant éventuellement les deux fonctions objectif), étant donné qu'il est peu probable que l'on parvienne à découvrir de nombreuses solutions compromis avec une méthode multiobjectif.

Les résultats obtenus sur différentes variantes du flow shop montrent que HBMOLS est globalement supérieur à trois algorithmes classiques de la littérature : PLS, NSGA-II et IBEA. Plus précisément, les algorithmes de recherche locale itérée, tels que PLS et HBMOLS, semblent particulièrement adaptés à la résolution des problèmes de flow shop. De plus, HBMOLS obtient globalement les meilleurs résultats, car à la différence de PLS, il permet d'une part de comparer les solutions dominées et non dominées entre elles ; de plus la taille variable de la population de PLS peut engendrer des problèmes de convergence lorsque celle-ci est trop grande ou trop petite. Par ailleurs, nous avons montré que les critères classiques du flow shop s'avèrent relativement corrélés, ce qui limite le nombre de solutions non dominées et explique en partie les bonnes performances de PLS.

Instance	Objectifs	NSGA	IBEA	PLS	HBMOLS
ta_20_05_01	(C_{Max},C_{Sum})	0,1703	0,2290	**0,0874**	0,1148
ta_20_10_01		0,1289	0,1220	**0,0435**	0,0436
ta_20_20_01		0,2149	0,2230	**0,0402**	0,0984
ta_50_05_01		0,0960	0,1046	0,1524	**0,0917**
ta_50_10_01		0,3673	0,3410	0,3336	**0,1859**
ta_50_20_01		0,3403	0,3281	0,3047	**0,1168**
ta_20_05_01	(C_{Max},T_{Max})	0,0353	0,0210	0,0201	**0,0113**
ta_20_10_01		0,0761	0,0723	0,0221	**0,0156**
ta_20_20_01		0,1292	0,1112	0,0403	**0,0177**
ta_50_05_01		0,3905	0,0173	0,0297	**0,0083**
ta_50_10_01		0,1119	0,1203	0,1240	**0,0825**
ta_50_20_01		0,1426	0,1523	0,1442	**0,0853**
ta_20_05_01	(C_{Max},T_{Sum})	0,1325	0,1149	**0,0755**	0,0620
ta_20_10_01		0,0905	0,0777	**0,0298**	0,0304
ta_20_20_01		0,1823	0,1691	**0,0225**	0,0405
ta_50_05_01		0,0992	0,0886	0,1286	**0,0731**
ta_50_10_01		0,1730	0,1785	0,1856	**0,1060**
ta_50_20_01		0,2326	0,2121	0,2008	**0,1217**
ta_20_05_01	(C_{Max},U_{Sum})	0,2118	0,2277	0,1912	**0,1511**
ta_20_10_01		0,1413	0,1378	0,0790	**0,0643**
ta_20_20_01		0,1706	0,1809	**0,0599**	0,0662
ta_50_05_01		**0,1925**	0,2480	0,5347	0,2861
ta_50_10_01		0,2882	0,2298	0,3229	**0,1517**
ta_50_20_01		0,2729	0,2645	0,3686	**0,1965**
ta_20_05_01	(C_{Sum},T_{Max})	0,0197	0,0188	0,0077	**0,0063**
ta_20_10_01		0,0540	0,0634	0,0163	**0,0122**
ta_20_20_01		0,0616	0,0522	**0,0073**	0,0083
ta_50_05_01		0,0771	0,0734	0,1536	**0,0540**
ta_50_10_01		0,1531	0,1700	0,1990	**0,0907**
ta_50_20_01		0,2623	0,2480	0,2360	**0,1248**
ta_20_05_01	(C_{Sum},T_{Sum})	0,0424	0,0394	0,0220	**0,0184**
ta_20_10_01		0,1228	0,1165	**0,0304**	0,0378
ta_20_20_01		0,1781	0,1556	**0,0219**	0,0501
ta_50_05_01		0,1872	0,1157	0,2463	**0,0976**
ta_50_10_01		0,3247	0,3032	0,2981	**0,1624**
ta_50_20_01		0,3725	0,3798	0,3160	**0,1793**
ta_20_05_01	(C_{Sum},U_{Sum})	0,1206	0,1088	0,0494	**0,0276**
ta_20_10_01		0,1136	0,0996	0,0171	**0,0107**
ta_20_20_01		0,1858	0,2118	0,0516	**0,0357**
ta_50_05_01		0,2691	0,2380	0,3591	**0,1945**
ta_50_10_01		0,4006	0,3707	0,3989	**0,1918**
ta_50_20_01		0,4471	0,4221	0,4024	**0,1513**
ta_20_05_01	(T_{Max},T_{Sum})	0,0469	0,0386	0,0211	**0,0192**
ta_20_10_01		0,0670	0,0850	0,0056	**0,0053**
ta_20_20_01		0,0566	0,0412	**0,0085**	0,0087
ta_50_05_01		0,1309	0,1179	0,2224	**0,1108**
ta_50_10_01		0,1827	0,1997	0,2230	**0,1095**
ta_50_20_01		0,3763	0,3421	0,3033	**0,1387**
ta_20_05_01	(T_{Max},U_{Sum})	0,0678	0,0641	0,0649	**0,0396**
ta_20_10_01		0,0806	0,0634	0,0309	**0,0110**
ta_20_20_01		0,0692	0,0567	0,0130	**0,0055**
ta_50_05_01		0,1154	**0,0801**	0,1782	0,0998
ta_50_10_01		0,1345	0,1011	0,1511	**0,0783**
ta_50_20_01		0,1555	0,1215	0,1445	**0,0566**
ta_20_05_01	(T_{Sum},U_{Sum})	0,1234	0,1076	0,0915	**0,0707**
ta_20_10_01		0,0707	0,0626	0,0060	**0,0008**
ta_20_20_01		0,1175	0,1030	0,0134	**0,0091**
ta_50_05_01		0,1314	0,1113	0,1596	**0,0997**
ta_50_10_01		0,1822	0,1614	0,1906	**0,0907**
ta_50_20_01		0,2618	0,2812	0,2445	**0,1102**

TABLE 8.8 – Résultats de NSGA-II, IBEA, PLS et HBMOLS sur différentes instances et différents objectifs du flow shop.

8.3.3 Discussion

Un des intérêts majeurs de HBMOLS est sa généricité, ce qui en fait un algorithme facilement adaptable à la résolution de nombreux problèmes d'optimisation multiobjectif. En effet, si l'on occulte les paramètres spécifiques au problème traité (représentation des solutions et structure de voisinage), l'adaptation d'HBMOLS à la résolution d'un nouveau problème nécessite peu d'efforts. Le point de référence Z_{ref} n'est pas un paramètre en soit puisqu'il suffit de le définir comme étant le plus mauvais possible. Il reste alors principalement à définir une taille de population pour l'algorithme. Les différentes expérimentations réalisées montrent de plus qu'utiliser une taille de population de l'ordre d'une dizaine de solutions est un choix acceptable, même si l'on peut souvent affiner ce paramètre pour améliorer légèrement les résultats. Le seul paramètre réel est la fonction d'initialisation des solutions de HBMOLS itéré, ceci étant un problème récurrent pour les recherches locales itérées. Cet aspect est d'ailleurs étudié partiellement dans le chapitre suivant, où nous proposons d'utiliser un algorithme de *path-relinking* comme fonction d'initialisation des populations.

Les expérimentations ont montré que HBMOLS est globalement très performant pour les problèmes à deux voire trois objectifs. Malheureusement, la complexité du calcul de l'hypervolume de dominance est telle que HBMOLS ne peut être efficace lorsque le nombre d'objectifs augmente. Dans ce cas, il est préférable d'utiliser IBMOLS$_\epsilon$ par exemple, ou tout autre indicateur efficace. L'utilisation d'indicateurs approximant la contribution à l'hypervolume de solutions peut être envisagée afin de trouver un compromis entre qualité de la sélection et temps d'évaluation de la qualité des solutions. D'une manière générale, il semble cohérent de privilégier tout indicateur qui se *rapproche* de la contribution à l'hypervolume, l'objectif étant de maximiser le nombre de fois où la sélection aboutit à un résultat identique, tout en minimisant le coût en temps de calcul, sans forcément chercher à approximer l'indicateur de contribution à l'hypervolume.

L'efficacité et la robustesse d'IBMOLS et de HBMOLS ont été éprouvées sur plusieurs problèmes combinatoires multiobjectif. Dans ce document nous avons en particulier exposé les résultats pour différentes variantes des problèmes de flow shop et d'assignement quadratique. Des résultats sur le *ring-star* ainsi que sur un problème d'emploi du temps d'infirmières ont été également publiés dans [Basseur *et al.*, 2012a]. Nous avons également obtenu des résultats très intéressants sur un problème de sac-à-dos multiobjectif ainsi que sur un problème de programmation quadratique binaire. Les travaux en cours sur ces problèmes montrent néanmoins certaines limites de ces algorithmes lorsque l'on cherche à résoudre des instances de très grande taille. En effet, la convergence peut s'avérer longue puisque ces algorithmes acceptent toute amélioration, si minime soit-elle, durant toute la recherche. Dans le cas d'étude de paysages de recherche très grands, il peut s'avérer nécessaire de favoriser une convergence rapide en cherchant à améliorer le plus possible la qualité globale de la population à chaque pas de la recherche. Un piste intéressante pourrait être de maximiser le gain de qualité en parcourant le voisinage complet d'une solution, ce qui équivaudrait à une approche de type `BEST` en optimisation mono-objectif. Une autre piste pourrait être de baser la recherche locale uniquement sur la notion de dominance de Pareto dans un premier temps, jusqu'à ce que cette recherche n'évolue plus. Ensuite, une recherche de type IBMOLS ou HBMOLS prendrait le relais pour affiner la qualité de l'approximation.

Chapitre 9

Path-relinking multiobjectif

Dans les chapitres précédents, nous avons présenté les algorithmes IBMOLS et HBMOLS, qui sont des recherches locales à base de population pour l'optimisation combinatoire multiobjectif. Évidemment, comme tous les algorithmes de type climber, il convient de définir une méthode qui permet de relancer la recherche lorsqu'un ensemble non dominé atteint un optimum local. Dans la section 7.2, nous avons réalisé quelques expérimentations sur trois approches classiques et les résultats ont montré que l'efficacité de la recherche dépendait de manière non négligeable de la fonction de relance de la recherche. Ces trois approches sont : (1) générer un ensemble de solutions aléatoires ; (2) sélectionner des solutions de l'approximation courante et leur appliquer une marche aléatoire ; (3) appliquer un opérateur de recombinaison sur des paires de solutions de l'approximation courante. Les résultats ont montré, comme dans de nombreuses autres études, qu'utiliser des solutions de l'archive pour relancer la recherche est plus efficace que de relancer la recherche *à l'aveugle*. Intuitivement, cela permet d'intensifier l'effort de recherche autour des meilleures solutions rencontrées et en même temps de réduire le temps nécessaire pour atteindre un nouvel optimum local, car on redémarre la recherche à partir de solutions de qualité souvent assez bonnes. Dans ce chapitre, nous nous intéressons à un autre moyen de générer des nouvelles solutions pour la recherche locale : la recomposition de chemin ou *path-relinking* (dans la suite, le terme anglais a été préféré). Nous verrons que le *path-relinking* est efficace et il s'avère de surcroît que l'on peut éventuellement éviter l'ajout de paramètre, ce qui est difficilement le cas pour la marche aléatoire.

9.1 Path-relinking multiobjectif

9.1.1 *Path-relinking*

Les algorithmes évolutionnaires de recherche par dispersion (*Scatter Search*) [Glover, 1977], et leur forme généralisée, les algorithmes de *path-relinking* [Glover et Laguna, 2000], ont eu un succès important ces dernières années.

Les algorithmes de recherche par dispersion se différencient des algorithmes évolutionnaires classiques principalement parce qu'ils accordent peu de place à l'aléatoire dans leur déroulement. Ils ont de surcroît prouvé leur efficacité en particulier en optimisation mono-objectif.

La recherche par dispersion opère sur un ensemble de solutions appelé ensemble de référence, en les combinant pour en créer de nouvelles. À la différence de nombreux algorithmes évolutionnaires, l'algorithme opère généralement sur un petit ensemble de solutions. Deux pièces essentielles de ces algorithmes sont : (1) Une méthode d'amélioration d'une solution initiale en une ou plusieurs solutions de meilleures qualité — cette méthode peut très bien être un climber par exemple ; (2) Une méthode de combinaison de solutions qui permet de transformer un ensemble de solutions donné en un ensemble de solutions combinant les caractéristiques d'au moins deux solutions.

Les algorithmes de *path-relinking* généralisent la combinaison de solutions en générant des chemins complets entre ces solutions (voire au-delà de ces solutions), via une relation de voisinage. Les solutions intermédiaires peuvent alors servir de base pour être améliorées ou même pour générer de nouveaux chemins vers d'autres solutions. Ce type d'algorithme est intuitivement intéressant, puisque générer un chemin permet de créer de nouvelles solutions qui partagent une partie des caractéristiques de chaque solution "parente", le taux d'héritage des caractéristiques de chaque parent dépendant de la position de la solution sur le chemin.

Pour générer un chemin d'une solution *initiale* vers une solution *de guidage*, il est nécessaire de définir comment choisir les mouvements de manière à introduire progressivement les caractéristiques de la solution de guidage. Ce processus peut être vu comme un opérateur de recombinaison généralisé, permettant de générer un ensemble de solutions dont les éléments héritent à des niveaux différents de chacun des parents (la solution initiale et la solution de guidage).

Dans cette section, nous proposons d'adapter ce concept à l'optimisation multiobjectif. Durant ma thèse, une étude combinant *path-relinking* et PLS avait été proposée [Basseur *et al.*, 2005a]. Il existe un petit nombre de travaux adaptant le concept de *path-relinking* au cadre multiobjectif, lesquels sont relativement récents [Pasia *et al.*, 2007], [Beausoleil *et al.*, 2008], [Seridi *et al.*, 2013], [Jaszkiewicz et Zielniewicz, 2009], [Drugan et Thierens, 2010] et [Jia et Hu, 2014].

L'impact de l'aspect multiobjectif des problèmes d'optimisation sur les algorithmes de *path-relinking* est en fait assez limité puisque cet algorithme se base essentiellement sur les variables de décision sans nécessairement prendre en compte la ou les fonctions d'évaluations. La figure 9.1 schématise simplement le *path-relinking* dans l'espace de décision et l'espace objectif. La mise au point d'un algorithme de *path-relinking* impose des choix dont certains sont très spécifiques à l'aspect multiobjectif des problèmes considérés :

- Structure du voisinage utilisée pour la génération de chemin.
- Opérateur de calcul de la distance entre les solutions (corrélée ou non, avec le voisinage).
- Critère de sélection de la solution initiale et de guidage.
- Critère de sélection du (des) chemin(s) à parcourir.
- Choix des solutions pour l'application du mécanisme d'intensification.

Les trois derniers aspects peuvent prendre en compte la qualité des solutions et des adaptations au cadre multiobjectif peuvent donc être proposés. La sélection des solutions initiales et de guidage peut être corrélée à la qualité générale des solutions (en favorisant les meilleures, par exemple). Mais, dans un cadre multiobjectif, on peut également vouloir favoriser les solutions proches dans l'espace objectif (dans le but d'intensifier la recherche dans cette zone), ou au contraire choisir des solutions éloignées afin de pouvoir découvrir des solutions de compromis intermédiaires grâce au *path-relinking*, comme proposé dans [Gandibleux *et al.*, 2003]. On peut se poser des questions similaires pour sélectionner un chemin parmi l'ensemble des chemins possibles, ainsi que pour

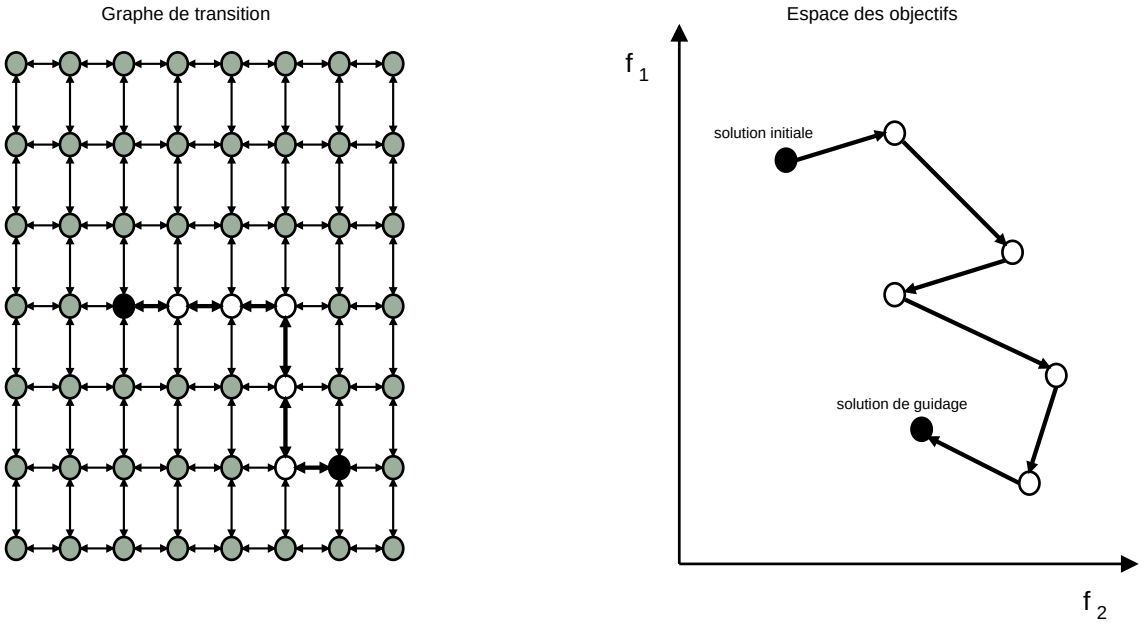


FIGURE 9.1 – Principe du *path-relinking*, du point de vue de l'espace de décision (à gauche) et de l'espace des objectifs (à droite).

choisir les solutions sur lesquelles on va appliquer l'algorithme d'intensification de la recherche. Dans la suite, nous nous proposons d'utiliser le *path-relinking* multiobjectif en collaboration avec HBMOLS, qui servira donc de méthode d'amélioration des solutions des chemins générés. L'objectif est d'une part d'évaluer l'efficacité du *path-relinking* pour initialiser les populations d'HBMOLS, d'autre part d'étudier le comportement de l'algorithme selon les choix effectués pour son implémentation.

9.1.2 Méthodologie

Durant ma thèse, nous avions proposé une première adaptation au cadre multiobjectif des algorithmes de *path-relinking* [Basseur *et al.*, 2005a]. L'algorithme proposé était une hybridation entre un algorithme génétique adaptatif et une combinaison de recherche locale Pareto (PLS) avec un algorithme de *path-relinking*. Voici les choix qui étaient effectués (détails dans l'algorithme) :

- Voisinage : le même que pour le mécanisme d'intensification (l'opérateur d'insertion, pour le problème de flow shop).
- Calcul de distance : corrélée avec le voisinage.
- Solution initiale et de guidage sélectionnées aléatoirement.
- Choix du chemin : on cherche un chemin minimal (la distance doit exactement diminuer de 1 à chaque pas du path-relinking). À chaque étape, parmi les solutions candidates, on choisit la meilleure, l'évaluation étant réalisée via une agrégation aléatoire des objectifs.
- Initialisation du mécanisme d'intensification : on prend les solutions non dominées du chemin généré.

Les résultats obtenus dans [Basseur *et al.*, 2005a] étaient très encourageants. Malgré cela, il nous a semblé intéressant d'étudier le comportement du *path-relinking* dans le cadre d'une col-

Algorithme 9.1 Algorithme de path-relinking proposé dans [Basseur *et al.*, 2005a].

Entrée : $\mathbf{A}$ (Population initiale obtenue par un algorithme génétique adaptatif)
Sortie : $\mathbf{A}$ (approximation de l'ensemble de Pareto)
$\mathbf{A} \leftarrow \emptyset$
$\mathbf{P} \leftarrow N$ solutions aléatoires.
tant que *critère d'arrêt non atteint* faire
  1) choisir aléatoirement la solution initiale $x_i \in \mathbf{A}$ et la solution de guidage $x_g \in \mathbf{A} \setminus \{x_i\}$
  2) $k \leftarrow 0$, $x_k \leftarrow x_i$
  3) tant que $x_k \neq x_g$ faire
    Définir aléatoirement des poids positifs $\lambda_1, \cdots, \lambda_N$
    $x_{k+1} \leftarrow \text{argmax}_{y \in \mathcal{N}(x_k), d(y,x_g)=d(x_k,x_g)-1}(\lambda_1 \times f_1(y) + \cdots + \lambda_N \times f_N(y))$
  4) $\mathbf{P} \leftarrow ND(\{x_0, x_1, \cdots, x_{d(x_i,x_g)-1}\})$
  5) $\mathbf{A} \leftarrow PLS(\mathbf{P})$
retourner $\mathbf{A}$

laboration avec HBMOLS, mais surtout d'étudier les résultats selon les choix réalisés pour son l'implémentation. Nous avons donc proposé HBMOLS$_{Path}$, consistant en une coopération entre HBMOLS et un algorithme de *path-relinking*, dont la version générique est détaillée dans l'algorithme 9.2.

Algorithme 9.2 HPR.

Entrée : T (Taille de la population)
Sortie : $\mathbf{A}$ (approximation de l'ensemble de Pareto)
$\mathbf{A} \leftarrow \emptyset$
$\mathbf{P} \leftarrow T$ solutions aléatoires.
tant que *critère d'arrêt non atteint* faire
  1) $\mathbf{A} \leftarrow \mathbf{A} \bigcup HBMOLS(\mathbf{P})$
  2) choisir aléatoirement la solution initiale $x_i \in \mathbf{A}$ et la solution de guidage $x_g \in \mathbf{A} \setminus \{x_i\}$
  3) $k \leftarrow 0$, $x_k \leftarrow x_i$
  4) tant que $x_k \neq x_g$ faire
    choisir $x_{k+1} \in \mathcal{N}(x_k)$ tel que $d(x_{k+1},x_g) = d(x_k,x_g) - 1$
  5) $\mathbf{P} \leftarrow$ sous-ensemble de $\{x_0, x_1, \cdots, x_{d(x_i,x_g)-1}\}$
retourner $\mathbf{A}$

Nous avons voulu étudier différentes versions de HBMOLS$_{Path}$. Pour cela nous nous sommes concentrés sur le choix du chemin à générer et sur le choix des solutions pour l'initialisation du mécanisme d'intensification. Pour les autres aspects, nous avons opté pour la conservation des choix réalisés dans [Basseur *et al.*, 2005a]. La question centrale qui nous a guidés est de savoir si la qualité des solutions était un critère pertinent pour réaliser ces choix.

Choix du chemin Soit x_k la solution courante et $y_0, \ldots, y_C$ l'ensemble des solutions candidates (on cherche un chemin minimal, donc la distance doit exactement diminuer de 1 à chaque pas du path-relinking). Voici les approches proposées :

1. Choix aléatoire de $x_{k+1} \in \{y_0, \ldots, y_C\}$.
2. Premier mouvement possible : $x_{k+1} \leftarrow y_0$ (spécifique au flow shop et testé uniquement sur ce problème).
3. Dernier mouvement possible : $x_{k+1} \leftarrow y_C$ (spécifique au flow shop et testé uniquement sur ce problème).

4. Choix aléatoire parmi les solutions candidates non dominées : $x_{k+1} \in ND(\{y_0,\ldots,y_C\})$.
5. Sélection basée sur l'hypervolume : $x_{k+1} \in \text{argmax}_{y\in\{y_0,\ldots,y_C\}}(HC(y,\{y_0,\ldots,y_C\}))$.

Initialisation du mécanisme d'intensification La sélection est réalisée sur l'ensemble des solutions du chemin $x_1,\cdots,x_{d(x_i,x_g)-1}$:

1. *All* : La totalité des solutions du chemin ($\{x_1,\cdots,x_{d(x_i,x_g)-1}\}$).
2. *Middle* : Une seule solution, celle se situant au milieu du chemin ($x_{\lceil d(x_i,x_g)/2\rceil}$).
3. *k-Middle* : Un ensemble de k solutions situées au milieu du chemin ($\{x_{\lceil (d(x_i,x_g)-k)/2\rceil},\cdots, x_{\lceil (d(x_i,x_g)-k)/2+k-1\rceil}\}$).
4. *Best* : L'ensemble des solutions non dominées du chemin ($ND(\{x_1,\cdots,x_{d(x_i,x_g)-1}\})$).

Les versions *Middle* et *k-Middle* ont pour but d'intensifier la recherche sur des solutions suffisamment éloignées de la solution initiales et de la solution de guidage, ceci afin d'éviter d'atteindre des optima locaux déjà explorés via HBMOLS.

9.2 Expérimentations

9.2.1 Implémentation

L'implémentation de HBMOLS$_{Path}$ est fortement liée à la représentation des solutions ainsi qu'à l'opérateur de voisinage utilisé, c'est-à-dire de la structure du graphe de transition considéré. En effet, la difficulté réside dans la mise au point d'une méthode permettant de relier les solutions entre elles. Pour cela, il est préférable de savoir comment calculer le nombre minimal de mouvements à réaliser pour relier deux solutions, ce qui correspond alors à la distance entre ces solutions (il semble cohérent de vouloir limiter la recherche aux chemins minimaux afin de ne pas considérer trop de chemins possibles). Heureusement, cela se calcule relativement facilement pour la plupart des graphes de transitions classiques issus de problèmes d'optimisation combinatoire.

Nous avons réalisé des expérimentations sur des instances de flow shop biobjectif ainsi que sur des instances de QAP biobjectif. Après avoir donné quelques précisions au niveau de l'implémentation, nous présenterons les résultats expérimentaux pour chacun des cas .

Application au flow shop

Comme précédemment, les solutions d'un problème de flow shop sont des permutations et nous utiliserons l'opérateur d'insertion. Nous considérons donc le graphe de transition FSP_{ins}. La distance associée au graphe de transition correspond au nombre minimal d'applications de l'opérateur d'insertion nécessaire pour relier deux solutions dans l'espace décisionnel. Soit $s_{max}(x_1,x_2)$ la plus longue séquence de jobs commune à x_1 et x_2, la distance $d_{perm}(x_1,x_2)$ entre les deux solutions x_1 et x_2 vaut (voir [Basseur *et al.*, 2005a] pour plus de détails) :

$$d_{ins}(x_1,x_2) = n - s_{max} \tag{9.1}$$

Le calcul de la plus longue séquence commune entre deux permutations se résout par programmation dynamique de complexité en $O(n^2)$.

À chaque pas de la génération d'un chemin, il est nécessaire de pouvoir envisager toutes les possibilités, que ce soit dans l'optique de choisir ensuite aléatoirement ou de manière déterministe le mouvement à réaliser. La figure 9.2 donne un exemple de génération de chemin entre deux permutations. Les jobs à déplacer sont ceux ne faisant pas partie de la plus longue sous-chaîne commune aux deux solutions considérées ; il convient donc d'envisager le déplacement des jobs restants, sachant qu'il y a souvent plusieurs positions acceptables permettant de se rapprocher de la solution de guidage. Cela se fait néanmoins assez facilement est n'est pas plus coûteux que le calcul de la distance entre deux solutions. De nombreux détails concernant le fonctionnement de cet algorithme peuvent être trouvés dans [Basseur *et al.*, 2005a] et [Zeng *et al.*, 2013b].

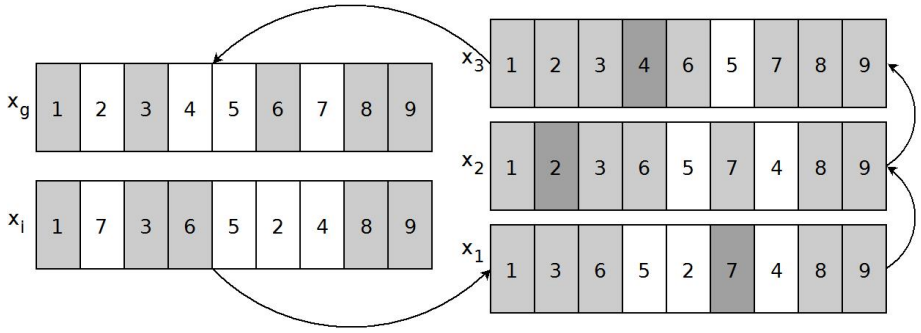


FIGURE 9.2 – Génération de chemin pour les paysages $\mathrm{FSP}_{\mathrm{ins}}$: la plus longue sous-chaîne commune à x_i et x_g apparait en grisé. Les autres jobs sont donc à déplacer (par exemple, à la première étape on peut déplacer le job 7 à toute position de la permutation située entre les jobs 6 et 8, ce qui donne quatre possibilités de déplacement pour ce job). Chemin de l'exemple : $x_i \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_g$.

Application au QAP

Pour le QAP nous considérons le graphe de transition $\mathrm{QAP}_{\mathrm{swap}}$, qui correspond à l'espace des permutations muni de l'opérateur de voisinage d'échange.

La détermination des chemins possibles est facile puisqu'il suffit de déplacer les unités mal placées à la bonne position dans la permutation. Cependant le calcul de la distance entre deux solutions dans le graphe de transition $\mathrm{QAP}_{\mathrm{swap}}$ est moins aisé. Ce calcul se base sur la notion de k-cycle, qui est un ensemble minimal de sites pour lequel les unités affectées dans x_i et x_g sont les mêmes d'un point de vue ensembliste. La distance entre x_i et x_g vaut alors le nombre d'unités moins le nombre de k-cycles de $\{x_i,x_g\}$ plus 1 (plus de détails peuvent être trouvés dans [Thierens, 2006, Schiavinotto et Stützle, 2011]). La figure 9.3 donne un exemple de génération de chemin et de calcul de distance pour le $\mathrm{QAP}_{\mathrm{swap}}$.

9.2.2 Résultats

Le protocole expérimental est similaire à ceux mis au point dans le reste de ce document, de plus amples précisions pouvant être trouvées dans [Zeng *et al.*, 2013b]. Les différentes versions

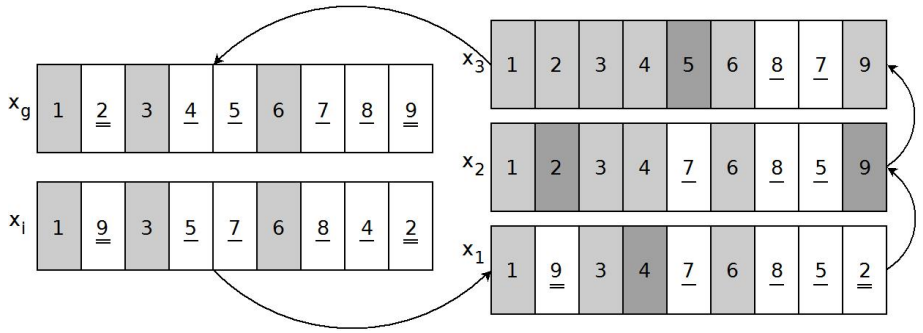


FIGURE 9.3 – Génération de chemin pour le paysages $\mathrm{QAP}_{\mathrm{swap}}$: Les unités ne se trouvant pas à la bonne position sont à déplacer, il suffit de les déplacer une à une sur le bon site dans la permutation. Les 3 cases grisées sont des 1-cycles, les unités 2 et 9 forment un 2-cycle et les unités 4, 5, 7, 8 forment un 4-cycle. Il y a 9 unités et 5 k-cycles, donc la distance entre x_u et x_g est de $9 - 5 = 4$. Chemin de l'exemple : $x_i \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_g$.

de l'algorithme de *path-relinking* ne nécessitent pas de définir de paramètre, hormis pour la version *k-Middle* pour laquelle le paramètre k doit être fixé. *k-Middle* permet de lancer HBMOLS à partir de plusieurs solutions du chemin alors que pour *Middle* HBMOLS est initialisé avec une seule solution du chemin (la population étant complétée avec des solutions aléatoires). Il n'est pas nécessaire de fixer k de manière très précise, il suffit de s'assurer que quelques solutions centrales du chemin seront sélectionnées. Nous avons donc fixé k de la manière suivante :

$$k = \left\lfloor \sqrt{d(x_i,x_g) - 1} \right\rfloor$$

Les résultats concernant les stratégies de choix de chemin ne sont pas détaillés ici, mais peuvent être trouvés dans [Zeng *et al.*, 2013b] pour le problème de flow shop. Les résultats nous ont permis de montrer que la différence de performance entre les cinq méthodes de choix de chemin minimaux était globalement peu significative. En particulier, ce n'est pas particulièrement profitable de chercher à générer un chemin en favorisant les solutions intermédiaire de bonne qualité. Nous nous sommes donc concentrés sur les méthodes de sélection de solution des chemins générés pour initier la recherche HBMOLS, en considérant une génération aléatoire de chemins (minimaux).

La table 9.1 résume les résultats obtenus pour le flow shop. Concentrons-nous dans un premier temps sur la comparaison relative des variantes de *path-relinking*. Globalement, *k-Middle* obtient les meilleurs résultats : il domine en moyenne les autres méthodes sur l'ensemble des instances, sauf sur les plus petites (à 20 jobs) où il est moins bon en moyenne que *All*. Notons que, quasi systématiquement, le choix le moins efficace est celui de lancer la recherche sur les meilleures solutions du chemin (*Best*).

En comparant la variante *k-Middle* de HBMOLS_{Path} avec les deux autres versions itérées classiques de HBMOLS (HBMOLS_{Walk} pour le bruit aléatoire et HBMOLS_{Cross} pour la recombinaison de solutions), on observe que cette version de *path-relinking* est très compétitive. De manière assez globale, Les résultats obtenus sont meilleurs que les deux autres variantes pour les grandes instances, alors que la performance est plutôt limitée pour les petites instances (surtout par rapport à HBMOLS_{Walk} qui la domine statistiquement sur l'ensemble des instances à 20 et 30

Instance	$HBMOLS_{Path}$				$HBMOLS_{Walk}$	$HBMOLS_{Cross}$
	All	Best	Middle	k-middle		
20_05_01	0,0505	0,0766	0,0938	0,0670	**0,0003**	0,0052
20_10_01	0,0234	0,0555	0,0483	0,0346	**0,0007**	0,0274
20_15_01	0,0324	0,0732	0,0704	0,0377	**0,0023**	0,0371
20_20_01	0,0097	0,0345	0,0248	0,0101	**0,0001**	0,0448
30_05_01	0,0493	0,0812	0,0997	0,0406	**0,0118**	0,0620
30_10_01	0,1001	0,2010	0,1764	0,0888	**0,0418**	0,1166
30_15_01	0,0525	0,0962	0,1053	0,0482	**0,0282**	0,0540
30_20_01	0,0484	0,0648	0,0712	0,0406	**0,0358**	0,0510
50_05_01	0,0312	0,0835	0,0903	**0,0226**	0,0410	0,0566
50_10_01	0,1039	0,1499	0,1322	**0,0795**	0,0897	0,1161
50_15_01	0,1316	0,1736	0,1570	**0,0916**	0,1149	0,1315
50_20_01	0,1297	0,1765	0,1464	**0,0935**	0,1171	0,1417
70_05_01	0,1106	0,1915	0,1521	0,0961	**0,0840**	0,1467
70_10_01	0,1312	0,1779	0,1574	**0,1191**	0,1464	0,1723
70_15_01	0,1498	0,1745	0,1642	**0,1346**	0,1570	0,1788
70_20_01	0,1394	0,1839	0,1476	**0,1021**	0,1355	0,1377
100_05_01	0,1993	0.3590	0,2361	**0,1578**	0,1698	0,1752
100_10_01	0,0939	0,1217	0,1041	**0,0711**	0,0803	0,0866
100_15_01	0,1873	0,2059	0,1759	**0,1289**	0,1633	0,1748
100_20_01	0,2059	0,2209	0,1873	**0,1318**	0,1372	0,1804

TABLE 9.1 – Initialisation des recherches locales HBMOLS : comparaison de quatre versions basées sur le *path-relinking* avec deux méthodes d'initialisation classiques (marches aléatoires et recombinaisons). Résultats sur des instances de flow shop biobjectif.

machines).

La comparaison des différents algorithmes sur le QAP apparait dans la table 9.2. Bien que les résultats ne soient pas particulièrement en faveur du *path-relinking*, la version *k-middle* obtient est tout à fait compétitive, même si comme pour le flow shop, $HBMOLS_{Walk}$ est plus efficace sur les petites instances. D'une manière générale, sur le flow shop et pour le QAP, HBMOLS initialisé grâce au *path-relinking* permet d'obtenir des résultats plus intéressants que lorsque il est initialisé à l'aide de recombinaisons classiques.

9.3 Discussion

Dans le chapitre 8, nous avons montré au travers d'expérimentations que HBMOLS est performant pour approximer les fronts de Pareto des problèmes combinatoires lorsque le nombre d'objectifs étudiés est limité. La simplicité de l'algorithme qui est en fait une simple recherche locale multiobjectif sur un ensemble de taille fixée rend l'algorithme attrayant. De plus, si on considère un paysage de recherche multiobjectif défini, HBMOLS ne dépend que d'un seul paramètre : la taille de la population. De plus, diverses expérimentations ont montré que même ce paramètre est peu influent du moment où on le définit de manière raisonnable (de l'ordre d'une dizaine).

Malheureusement, la mise en place d'une version itérée de HBMOLS, peut nécessiter de définir de nouveaux paramètres (quantité de bruit, opérateur de recombinaison, etc.). Dans ce chapitre,

Instance	$HBMOLS_{Path}$				$HBMOLS_{Walk}$	$HBMOLS_{Cross}$
	All	Best	Middle	k-middle		
chr_12_a+chr_12_b	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**	0,0134
chr_15_a+chr_15_b	0,0030	0,01100	**0,0000**	0,0023	**0,0000**	0,0265
chr_20_a+chr_20_b	0,0140	0,0253	0,0048	0,0056	**0,0019**	0,0179
esc_16_a+esc_16_b	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
esc_32_a+esc_32_b	0,0063	0,0088	0,0026	0,0030	**0,0024**	0,0079
Lipa_30_a+Lipa_30_b	0,0020	0,0022	0,0014	**0,0013**	0,0014	0,0030
Ste_36_a+Ste_36_b	0,3041	0,3567	0,6696	0,3640	0,2153	**0,2038**
tai_40_a+tai_40_b	0,0375	0,0419	0,0320	**0,0271**	0,0461	0,0805
tai_50_a+tai_50_b	0,0386	0,0305	0,0406	**0,0271**	0,0484	0,0462

TABLE 9.2 – Initialisation des recherches locales HBMOLS : comparaison de quatre versions basées sur le *path-relinking* avec deux méthodes d'initialisation classiques (marches aléatoires et recombinaisons). Résultats sur des instances de QAP biobjectif.

nous avons étudié l'utilisation du *path-relinking* pour relancer la recherche de HBMOLS$_{Path}$. La méthode de génération de chemins entre solutions est impliquée par l'opérateur de voisinage utilisé (en se concentrant éventuellement sur les chemins minimaux). Nous avons montré expérimentalement que choisir aléatoirement le chemin à générer parmi les chemins possibles était une solution acceptable. De plus, il est souhaitable de relancer HBMOLS à partir de solutions se trouvant au milieu du chemin généré, cela permet d'éviter de relancer la recherche sur des solutions proches de solutions existantes de l'archive. Par conséquent, l'utilisation du *path-relinking* ne nécessite pas forcément de fixer de nouveaux paramètres. Au final, HBMOLS$_{Path}$ est dépendant d'une taille de population et du paramètre k, qui peut éventuellement être calculé automatiquement sans utiliser de calculs complexes. De plus, les résultats obtenus par HBMOLS$_{Path}$ sont très encourageants et poussent à continuer l'étude de cette méthode d'initialisation des ILS.

Chapitre 10

Recherche locale multiobjectif basée sur des ensembles

Le problème d'approximation de l'ensemble Pareto optimal d'un problème d'optimisation multiobjectif a récemment été redéfini sous forme de problème d'ensembles, dans lequel l'espace de recherche est constitué de l'ensemble des ensembles de solutions du problème optimisé. Cette définition, munie de l'évaluation par un indicateur unaire de qualité, permet de ramener les problèmes d'optimisation multiobjectif à un cadre mono-objectif. Dans ce contexte, nous introduisons SbLS, un algorithme de recherche locale générique basée sur les ensembles. Ensuite, nous définissons plusieurs classes de structures de voisinages basées sur des ensembles de solutions, qui permettent la conception de différentes variantes de recherches locales basées sur les ensembles. Des expérimentations sont réalisées sur les paysages NK multiobjectif et valident ce type d'approche qui ouvre la voie à des perspectives en termes de conception d'algorithmes d'optimisation multiobjectif.

10.1 Optimisation multiobjectif basée sur les ensembles

10.1.1 Notations et motivations

Le travail réalisé ici consiste en une extension du concept d'optimisation multiobjectif basée sur les ensembles (*set-based multiobjective optimisation*), introduit dans [Zitzler *et al.*, 2010]. L'objectif est de définir et formaliser les recherches locales basées sur les ensembles (*Set-based Local Search* — SbLS). Avec une telle formulation, la plupart des métaheuristiques évolutionnaires multiobjectif peuvent être vues comme des recherches locales appliquées à des ensembles de solutions.

Dans la suite, on appellera solution-élément une solution réalisable $x \in \mathcal{X}$ et $\mathcal{Z} = \mathfrak{F}(\mathcal{X}) \subset \mathbb{R}^N$ représentera l'ensemble des vecteurs objectif atteignables dans l'espace des objectifs. On rappelle que $\mathfrak{f}_1, \ldots, \mathfrak{f}_n$ sont les composantes du vecteur $\mathfrak{F}$.

Approximer l'ensemble Pareto optimal d'un problème d'optimisation multiobjectif peut être formulé comme un problème d'optimisation mono-objectif où l'on cherche à maximiser la qualité d'un ensemble de solutions-éléments. On appellera par la suite *solution-ensemble* un ensemble de

Var.	Signification
	Domaine des solutions
$\mathcal{X}$	ensemble des solutions-éléments de l'espace de recherche
x	*solution-élément*, c'est-à-dire une solution réalisable du problème d'optimisation d'origine ($x \in \mathcal{X}$)
$\mathcal{Z}$	ensemble des vecteurs objectif réalisable de l'espace de recherche
z	un vecteur objectif ($z \in \mathcal{Z}$)
N	nombre de fonctions objectif
$\mathfrak{F}$	vecteur de fonctions objectif ($\mathfrak{F} : \mathcal{X} \to \mathcal{Z}$)
$\mathcal{N}$	relation de voisinage dans le domaine des solutions-éléments ($\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$)
	Domaine des ensembles de solutions
Σ	ensemble des solutions-ensembles réalisables (Σ est l'ensemble des parties de $\mathcal{X}$)
σ	une *solution-ensemble* ($\sigma \in \Sigma$)
$\mathtt{I}$	fonction d'évaluation (indicateur unaire de qualité) dans le domaine des ensembles ($\mathtt{I} : \Sigma \to \mathbb{R}$)
$\mathtt{I}_H$	indicateur unaire de qualité d'hypervolume de dominance
$\mathtt{N}$	relation de voisinage sur les solutions-ensembles ($\mathtt{N} : \Sigma \to 2^{\Sigma}$)
$\mathcal{C}$	classe de voisinage sur les solutions-ensembles conforme avec le voisinage des solutions-éléments $\mathcal{N}$

TABLE 10.1 – Notations.

solutions-éléments. La table 10.1 recense les notations principales utilisées dans ce chapitre.

En considérant l'objectif d'approximer l'ensemble Pareto optimal, on se place donc dans un espace de recherche constitué de solutions-ensembles. Comme en optimisation mono-objectif, la fonction d'évaluation des solutions ainsi que la relation de voisinage utilisée jouent un rôle primordial dans la conception d'algorithmes de recherches locales, l'efficacité de la recherche étant bien sûr étroitement liée à ces deux composants de base.

Considérant un indicateur de qualité faisant office de fonction d'évaluation et une structure de voisinage basée sur les solutions-ensembles, toute métaheuristique mono-objectif peut potentiellement être appliquée pour identifier une approximation de l'ensemble Pareto optimal.

Dans la suite, nous proposons l'algorithme générique SbLS permettant la reformulation des approches multiobjectif classiques sous forme d'algorithme de recherche locale mono-objectif, ainsi que leur analyse comparative unifiée avec ce qui se fait habituellement en optimisation mono-objectif.

10.1.2 Espace de recherche des solutions-ensembles

L'espace de recherche $\Sigma \subset 2^{\mathcal{X}}$ se définit comme l'ensemble des ensembles de solutions réalisables ou ensemble des solutions-ensembles. Intuitivement, il semble cohérent d'imposer une cardinalité maximale à ces solutions-ensembles, de la forme $|\sigma| \leqslant \mu$ pour tout $\sigma \in \Sigma$. $x \in \mathcal{X}$ représente alors une solution-élément tandis que $\sigma \in \Sigma$ représente une solution-ensemble.

Voici des exemples de domaines de recherche des solutions-ensembles que l'on peut considérer [Verel *et al.*, 2011a] :

- L'espace de recherche des algorithmes à base de populations de solutions-éléments peut se définir comme $\Sigma_{=\mu} = \{\sigma \in 2^{\mathcal{X}} \ : \ |\sigma| = \mu\}$, où μ est la taille de population.
- L'espace de recherche d'algorithmes utilisant une taille bornée d'archive de solutions-éléments sera défini comme $\Sigma_{\leqslant\mu} = \{\sigma \in 2^{\mathcal{X}} \ : \ |\sigma| \leqslant \mu\}$, où μ est la taille maximale de l'archive.

- L'espace de recherche des approches basées sur la dominance de Pareto où l'on ne manipule que des solutions-éléments mutuellement non dominées se défini comme : $\Sigma^{\not\prec} = \{\sigma \in 2^{\mathcal{X}} : \forall x,x' \in \sigma, x \not\prec x'\}$.
- L'espace de recherche combinant les deux restrictions précédentes sera défini de la manière suivante : $\Sigma^{\not\prec}_{\leqslant\mu} = \{\sigma \in 2^{\mathcal{X}} : |\sigma| \leqslant \mu$ et $\forall x,x' \in \sigma, x \not\prec x'\}$ (seul un nombre limité de solutions mutuellement non dominées est autorisé).
- Un espace de recherche sans restriction sera défini comme $\Sigma^{\star} = 2^{\mathcal{X}}$.

10.1.3 Fonction d'évaluation de solutions-ensembles

En optimisation multiobjectif, il y a différentes manières de définir ce qui est une bonne approximation ; cela dépend principalement des préférences du décideur. Comme dans le reste de cette partie, on se place dans le contexte où l'on a aucune connaissance des préférences du décideur et l'on cherche donc à approximer l'ensemble Pareto optimal dans son ensemble. Il existe deux classes de relations sur Σ pour comparer deux solutions-ensembles : les relations de dominance, comme par exemple la dominance de Pareto étendue aux solutions-ensembles ou l'utilisation d'indicateurs de qualité. Afin de pouvoir conformer la recherche sur des solutions-ensembles aux recherches locales mono-objectif, il est approprié de s'appuyer sur des indicateurs unaires de qualité. En effet, en optimisation mono-objectif, la recherche est guidée par la comparaison de solutions ayant été évaluées sous forme de scalaires, nous nous imposons donc de garder cet aspect fondamental qui permet de guider la recherche. Les indicateurs unaires de qualité, et en particulier l'hypervolume de dominance permettent d'introduire une relation d'ordre total entre les solutions-ensembles. D'une manière générale, la relation de préférence entre solutions-ensembles sera donc donnée ici sous forme d'un indicateur unaire de qualité $\mathtt{I} : \Sigma \rightarrow \mathbb{R}$.

Nous nous concentrons ici sur l'indicateur d'hypervolume de dominance $\mathtt{I}_H$, approprié pour évaluer des approximations sans connaissance de préférences du décideur. Cependant, la suite peut être adaptée aisément en modifiant cet indicateur de qualité en fonction de connaissances éventuelles sur ces préférences.

Notons qu'une solution-ensemble de taille bornée par μ maximisant la valeur $\mathtt{I}_H$ est nécéssairement un sous-ensemble de l'ensemble Pareto optimal.

10.1.4 Objectif du problème d'optimisation

Les approches de résolution des problèmes multiobjectif classiques consistent donc à trouver un ensemble de solutions-éléments compromis dans l'espace des solutions-éléments réalisables. Ici, l'espace de recherche est composé de l'ensemble des ensembles de solutions-éléments. On cherche donc ici une solution-ensemble optimale en considérant l'espace de recherche des solutions-ensembles et un indicateur de qualité permettant de les évaluer. L'objectif de la recherche est alors de trouver ou d'approximer la solution-ensemble $\sigma \in \Sigma$ qui maximise l'indicateur considéré. Un *problème d'optimisation basé sur les ensembles* peut être formulé de la manière suivante :

$$\arg\max_{\sigma \in \Sigma} \mathtt{I}(\sigma) \tag{10.1}$$

Par conséquent, $\mathtt{I}$ est une fonction qui affecte à chaque solution-ensemble une valeur scalaire reflétant sa qualité par rapport à l'indicateur unaire utilisé (equation 10.1), c'est-à-dire une fonction

Algorithme 10.1 SbLS

Entrée : $\sigma \in \Sigma$
évaluer σ par rapport à $\mathtt{I}$
répéter
  choisir σ' tel que $\sigma' \in \mathtt{N}(\sigma)$
  évaluer σ' par rapport à $\mathtt{I}$
  si accepte(σ,σ') **alors**
    $\sigma \leftarrow \sigma'$
  fin si
jusqu'à critère d'arrêt atteint
retourne le meilleur σ trouvé

d'évaluation définie sur les solutions-ensembles. Notons que ce problème peut être lié au problème de trouver un sous-ensemble de μ solutions-éléments maximisant $\mathtt{I}$, c'est-à-dire la distribution optimale de μ solutions-éléments par rapport à $\mathtt{I}$ (voir [Auger *et al.*, 2012]).

10.1.5 Limites

Dans l'article fondateur [Zitzler *et al.*, 2010] de l'optimisation multiobjectif basée sur les ensembles, une définition de ce qu'est un problème d'optimisation basé sur les ensembles est donnée, mais pas pour ce qu'est une recherche locale basée sur les ensembles. En particulier, il n'est pas précisé clairement ce que peut être un opérateur de voisinage dans ce contexte, l'approche étant restreinte à la définition de *mutations aléatoires de solutions-ensembles* ou de *heuristique de mutation de solutions-ensembles*. Pourtant, la définition d'une structure de voisinage sur les solutions-ensembles permet de pouvoir distinguer différentes propriétés de l'espace de recherche et des heuristiques à employer pour explorer ces voisinages. Les différentes définitions possibles de voisinages de solutions-ensembles représentent l'aspect principal permettant de différencier plusieurs types de recherches locales basées sur les ensembles ayant des comportements très différents. Nous nous intéresserons donc à cet aspect dans ce qui suit.

10.2 Recherche locale basée sur des ensembles

En optimisation mono-objectif, une recherche locale est basée sur une stratégie de mouvement qui s'appuie sur la définition d'un triplet $(\mathcal{X},\mathcal{N},\mathfrak{f})$ (espace de recherche, relation de voisinage et fonction d'évaluation). De la même manière, la conception d'une recherche locale basée sur les ensembles nécessite de définir adéquatement un espace de recherche, une structure de voisinage et une fonction d'évaluation. Une recherche locale basée sur les ensembles sera donc définie ici par le triplet $(\Sigma,\mathtt{N},\mathtt{I})$ tel que :

- $\Sigma \subset 2^{\mathcal{X}}$ est l'ensemble des solution admissibles,
- $\mathtt{N} : \Sigma \to 2^{\Sigma}$ est une relation de voisinage entre les solutions-ensembles, et
- $\mathtt{I} : \Sigma \to \mathbb{R}$ est un indicateur unaire de qualité, c'est-à-dire une fonction d'évaluation mesurant la qualité des solutions-ensembles. Σ, $\mathtt{N}$ et $\mathtt{I}$ se doivent donc d'être définis en fonction du problème considéré, ce qui est également le cas en optimisation mono-objectif.

L'algorithme 10.1 donne le pseudo-code d'une recherche locale générique basée sur les ensembles (SbLS — *Set-based Local Search*). Si la stratégie de sélection n'accepte que des solutions-ensembles voisines améliorantes, SbLS est une généralisation des climbers pour l'optimisation multiobjectif basée sur les ensembles. Dans un tel contexte et sachant que l'indicateur de qualité fournit une relation d'ordre total entre les solutions-ensembles, il est clair que le climber de type SbLS se termine naturellement en atteignant un optimum local (dans le domaine des solutions-ensembles défini, voir définition 17).

Définition 17 *Une solution-ensemble $\sigma \in \Sigma$ est dite **optimum local au sens des ensembles**, considérant une relation de voisinage d'ensembles* $\mathtt{N}$ *et une fonction d'évaluation d'ensembles* $\mathtt{I}$*, si et seulement si* $\forall \sigma' \in \mathtt{N}(\sigma)$, $\mathtt{I}(\sigma') \leqslant \mathtt{I}(\sigma)$.

Il est bien sûr possible d'étendre l'algorithme SbLS de telle sorte qu'il puisse intégrer le principe des recherches locales avancées comme la recherche Tabu, le recuit simulé ou la recherche locale itérée. Ces algorithmes permettent en particulier de pouvoir s'échapper d'optima locaux atteints prématurément afin de relancer la recherche pour atteindre de meilleurs optima locaux. Cependant, on ne sait pas dans quelle mesure les optima locaux d'ensembles empêchent d'atteindre des ensembles de bonne qualité de manière aussi claire qu'en optimisation mono-objectif.

10.3 Relations de voisinage sur les solutions-ensembles

Dans cette section, nous introduisons plusieurs définitions possibles de relations de voisinage entre solutions-ensembles $\mathtt{N} : \Sigma \rightarrow 2^{\Sigma}$. Afin de simplifier ces définitions, nous ne considérons que les voisinages de solutions-ensembles basés sur un opérateur de voisinage ad hoc sur les solutions-éléments. Nous proposons dans un premier temps des classes de voisinages sur les solutions-ensembles, puis nous discutons des problèmes posés par ces différentes classes de voisinage en ce qui concerne la méthode d'exploration et de sélection des voisins améliorants. Enfin, nous ferons un parallèle entre les algorithmes classiques de la littérature et les approches de type SbLS.

10.3.1 Classes de voisinages

Introduisons dans un premier temps le concept de voisinage d'ensembles $\mathtt{N} : \Sigma \rightarrow 2^{\Sigma}$ conforme avec un voisinage de solutions-éléments $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$:

Définition 18 *Une relation de voisinage de solutions-ensembles* $\mathtt{N}$ *est **conforme** avec une relation de voisinage sur les solutions-éléments $\mathcal{N}$ ($\mathcal{N}$-conforme pour simplifier) si et seulement si :*

$$\forall \sigma \in \Sigma, \forall \sigma' \in \mathtt{N}(\sigma), \forall x' \in (\sigma' \setminus \sigma), \exists x \in \sigma \text{ tel que } x' \in \mathcal{N}(x) \tag{10.2}$$

Soit une solution-ensemble $\sigma \in \Sigma$ et une solution-ensemble de son voisinage $\sigma' \in \mathtt{N}(\sigma)$. Toute solution-élément $x \in \sigma'$ se trouve soit dans la solution-ensemble d'origine σ ou dans le voisinage $\mathcal{N}(x')$ d'une solution-élément de la solution-ensemble d'origine $x' \in \sigma$. Cette relation n'est pas inversible dans le cas général, puisqu'il n'est pas nécessaire que toute solution-élément de σ soit connectée avec une solution-élément de σ' (des solutions-éléments peuvent être supprimées). En particulier, la solution-ensemble vide $\{\}$ est voisine de toute solution-ensemble pour certaines

définitions de l'espace de recherche. Voici quatre relations de voisinages génériques basées sur les solutions-ensembles et $\mathcal{N}$-conformes :

1. $\sigma' \in \mathtt{N}^{(1,1)}(\sigma) \Rightarrow |\sigma' \setminus \sigma| \leqslant 1$ et $\forall x' \in \sigma' \setminus \sigma, \exists x \in \sigma, x' \in \mathcal{N}(x)$.
2. $\sigma' \in \mathtt{N}^{(1,\star)}(\sigma) \Rightarrow \exists x \in \sigma, \forall x' \in \sigma' \setminus \sigma, x' \in \mathcal{N}(x)$.
3. $\sigma' \in \mathtt{N}^{(\star,1)}(\sigma) \Rightarrow \exists \sigma_0 \subset \sigma, \exists \phi : \sigma_0 \rightarrow \sigma' \setminus \sigma$ avec $\forall \sigma' \in \sigma' \setminus \sigma, \exists!\ x \in \sigma_0$ tel que $\phi(x) = x'$ et $x' \in \mathcal{N}(x)$.
4. $\sigma' \in \mathtt{N}^{(\star,\star)}(\sigma) \Rightarrow \forall x' \in \sigma' \setminus \sigma, \exists x \in \sigma$ tel que $x' \in \mathcal{N}(x)$.

Ces quatre voisinages génériques décrivent différentes manières de naviguer dans un espace de recherche de solutions-ensembles. Notons que pour toute solution-ensemble $\sigma \in \Sigma$, $\mathtt{N}^{(1,1)}(\sigma) \subset (\mathtt{N}^{(1,\star)}(\sigma) \cup \mathtt{N}^{(\star,1)}(\sigma)) \subset \mathtt{N}^{(\star,\star)}(\sigma)$.

Par extension, le concept de classe de voisinage basée sur les solutions-ensembles peut être défini comme suit :

Définition 19 *Une relation de voisinage de solutions-ensembles* N *est de classe* $\mathcal{C}(\xi_1,\xi_2)$ *si et seulement si* $\forall \sigma \in \Sigma, \forall \sigma' \in \mathtt{N}(\sigma), \sigma' \in \mathtt{N}^{(\xi_1,\xi_2)}(\sigma)$ *avec* $\xi_i \in \{1,\star\}$.

En d'autres termes, une relation de voisinage N de classe $\mathcal{C}(\xi_1,\xi_2)$ implique que pour toute solution-ensemble, $\sigma \in \Sigma$, $\mathtt{N}(\sigma) \subset \mathtt{N}^{(\xi_1,\xi_2)}(\sigma)$.

Considérant une relation de voisinage de solution-ensemble de classe $\mathcal{C}(1,1)$, une seule solution-élément au maximum peut-être ajoutée. Pour un voisinage de classe $\mathcal{C}(1,\star)$, plusieurs solutions-éléments peuvent être ajoutées, mais toutes doivent appartenir au voisinage d'une unique solution-élément de la solution-ensemble d'origine. Un voisinage de classe $\mathcal{C}(\star,1)$ permet d'ajouter plusieurs solutions-éléments, mais les solutions-éléments ajoutées doivent appartenir au voisinage de solutions-éléments distinctes de la solution-ensemble d'origine. Enfin, la relation de voisinage $\mathtt{N}^{(\star,\star)}$ de classe $\mathcal{C}(\star,\star)$ correspond à la définition non contrainte d'une relation de voisinage $\mathcal{N}$-conforme. Pour les quatre classes de voisinage, un nombre quelconque de solutions-éléments de la solution-ensemble d'origine peuvent être potentiellement supprimées.

Notons que, étant donné que les solutions-ensembles peuvent être vues comme des populations de solutions-éléments, il est possible d'étendre les définitions précédentes afin d'inclure d'autres opérateurs sur les populations tels que la recombinaison.

10.3.2 Règle pivot

En optimisation multiobjectif, le coût calculatoire de l'exécution d'un climber est sans conteste bien plus important qu'en optimisation mono-objectif, en particulier parce que cela impose d'améliorer un ensemble de solutions compromis dans un espace des objectifs au moins bidimensionnel. Il n'est d'ailleurs pas rare qu'exécuter un climber soit trop coûteux pour être effectué expérimentalement. Afin de converger au plus vite vers un optimum local, il peut être souvent préférable de chercher à réaliser les plus grandes améliorations à chaque pas de la recherche (à la manière de la stratégie `BEST`). Le problème de trouver le meilleur voisin d'une solution-ensemble $\sigma \in \Sigma$ peut être formalisé de la manière suivante :

$$\arg \max_{\sigma' \in \mathtt{N}(\sigma)} \mathtt{I}(\sigma') \tag{10.3}$$

L'objectif est alors de trouver une solution-ensemble (en général de taille bornée par μ) qui a la meilleure qualité au sens de l'indicateur unaire utilisé, parmi le voisinage de la solution d'origine (nous nommerons ceci *problème de sélection de voisinage de solution-ensemble* (SbNS)). Il est clair que les structures de voisinages $\mathtt{N}^{(1,\star)}$, $\mathtt{N}^{(\star,1)}$ et $\mathtt{N}^{(\star,\star)}$ contiennent énormément de solutions et ne peuvent être énumérées dans la majeure partie des cas.

Par exemple, en restreignant l'espace de recherche à des solutions-ensembles de tailles bornées ($\Sigma_{\leqslant\mu}$), la taille du voisinage engendré n'est pas polynomiale par rapport à la taille du voisinage des solutions-éléments et par rapport à μ. Pire, même $\mathtt{N}^{(1,1)}$ a une cardinalité non polynomiale puisque le voisinage d'une solution-ensemble σ contient au moins toutes ses parties. L'énumération exhaustive du voisinage est d'autant plus difficile pour un espace de recherche non contraint ($\Sigma^\star$), puisque les solutions-ensembles d'un voisinage peuvent être de taille non polynomiale par rapport à la taille de la solution-ensemble d'origine.

Pour ces raisons, il est nécessaire de trouver un compromis entre la taille et la qualité de la relation de voisinage et le coût calculatoire de son exploration.

10.3.3 Exploration du voisinage

Afin de réduire le coût d'évaluation d'un voisinage de solution-ensemble, on peut s'inspirer de ce qui se fait en optimisation mono-objectif, c'est à dire effectuer une évaluation incrémentale des solutions. En effet, le calcul complet de l'hypervolume d'une solution-ensemble est $\#P$-complet [Bringmann et Friedrich, 2008], mais cette complexité peut être réduite en se concentrant sur le calcul de la contribution à l'hypervolume des solutions-éléments formant une solution-ensemble, en s'inspirant de [Beume *et al.*, 2007].

Malheureusement, au vu de la taille des voisinages à explorer, cela n'est pas suffisant pour pouvoir énumérer et évaluer de manière exhaustive le voisinage de solutions-ensembles. Par conséquent, il semble indiqué d'éviter d'adopter une stratégie du meilleur améliorant classique, en particulier si l'on considère ceux des classes peu restreintes. Il faut donc se concentrer sur des stratégies alternatives, comme ce qui se fait en optimisation mono-objectif pour les recherches à base de voisinages larges [Ahuja *et al.*, 2002]. Ce type d'approche cherche à identifier le meilleur voisin ou tout du moins un voisin améliorant en générant le moins de solutions possibles. La stratégie du premier améliorant en est un exemple simple, mais de nombreuses méthodes spécifiques au problème optimisé existent.

Considérant un algorithme de type SbLS dont l'évaluation est basée sur l'indicateur d'hypervolume, une approche classique, appelée *hypervolume-based greedy heuristic* (hgh) est proposée dans [Bader et Zitzler, 2011, Zitzler *et al.*, 2010]. Soit un ensemble de solutions-éléments dont la taille doit être réduite à μ. hgh consiste à supprimer la solution ayant la plus petite contribution à l'hypervolume total, puis de réitérer cette sélection jusqu'à ce que la taille de l'ensemble ne dépasse plus μ. Ceci n'assure pas d'avoir au final la solution-ensemble de taille μ de la meilleure qualité possible. Cette heuristique peut être facilement adaptée au problème SbNS. Dans un premier temps, on génère une solution-ensemble composée des solutions-éléments de la solution-ensemble d'origine ainsi que d'un certain nombre de solutions solutions-éléments de leur voisinage. Ensuite, on ramène la taille de la solution-ensemble à μ grâce à l'heuristique hgh.

Algo.	Référence	Σ	N	I
(1+1)-PAES	[Knowles et Corne, 2000]	$\Sigma^{\not\preccurlyeq}_{\leqslant_\mu}$	$\mathcal{N}$-conforme, $\mathcal{C}(1,1)$	$\mathtt{I}_H$ (parmi d'autres variantes)
SEMO	[Laumanns *et al.*, 2002]	$\Sigma^{\not\prec}$	$\mathcal{N}$-conforme, $\mathcal{C}(1,1)$	aucun (conforme avec $\mathtt{I}_H$)
PLS	[Paquete *et al.*, 2007]	$\Sigma^{\not\prec}$	$\mathcal{N}$-conforme, $\mathcal{C}(1,\star)$	aucun (conforme avec $\mathtt{I}_H$)
SMS-EMOA	[Beume *et al.*, 2007]	$\Sigma_{=_\mu}$	non nécessairement $\mathcal{N}$-conforme	$\mathtt{I}_H$ (profondeur de dominance)

TABLE 10.2 – Algorithmes classiques de la littérature formulés d'un point de vue SbLS.

10.3.4 Parallèle avec des approches existantes

Un certain nombre d'algorithmes d'optimisation multiobjectif classiques de la littérature sont reformulables de manière à s'intégrer comme variante de l'algorithme SbLS. La table 10.2 donne les composants du modèle qui permettent de réécrire sous forme de variante de SbLS les algorithmes PAES [Knowles et Corne, 2000], SEMO [Laumanns *et al.*, 2002], PLS [Paquete *et al.*, 2007] et SMS-EMOA [Beume *et al.*, 2007] (une description plus précise de ce parallèle se trouve dans [Basseur *et al.*, 2013]). D'autres algorithmes de la littérature peuvent être adaptés de la sorte, mais il est bien sûr possible de concevoir de nombreuses méthodes originales s'intégrant dans l'algorithme générique SbLS.

10.4 Expérimentations

Dans cette section, nous présentons les résultats expérimentaux obtenus par différentes versions de SbLS sur des paysages ρMNK, qui sont une variante multiobjectif des paysages NK.

10.4.1 Paysages ρMNK

Des paysages NK multiobjectif, avec n objectifs indépendants, ont été introduits dans un premier temps dans [Aguirre et Tanaka, 2007]. Plus récemment, des paysages NK multiobjectif où les fonctions objectif sont corrélées ont été proposés [Verel *et al.*, 2011b]. Chaque élément de la matrice C des contributions est un vecteur composé de n réels. Une matrice de corrélation permet de définir le degré de corrélation entre les différentes composantes des vecteurs de C. Le degré de corrélation est défini par ρ et est fixe pour toute paire de fonctions objectif des paysages ρMNK. Le degré d'épistasie de chaque composante de C reste fixe également quel que soit l'objectif considéré. M désigne le nombre d'objectifs, N la taille de la chaîne de bits et K le degré d'épistasie des composantes de C. Plus de détails concernant ces instances se trouvent dans [Verel *et al.*, 2011b].

10.4.2 Protocole expérimental

Nous cherchons a déterminer l'influence des différents paramètres des paysages ρMNK sur la performance d'algorithmes de type SbLS. On se restreindra aux instances biobjectif ($M = 2$), mais en faisant varier la taille de l'instance (N), le niveau d'épistasie des variables (K) et le niveau de corrélation des objectifs (ρ).

Instances	Domaine des solutions-éléments	Domaine des solutions-ensembles
$\rho \in \{-0.5,0.0,+0.5\}$	$\mathcal{X} = \{0,1\}^N$	$\Sigma = \Sigma_{\leqslant\mu}$
$N \in \{256,512\}$	$\mathfrak{F}$ fonctions objectif	$\mathtt{I} = \mathtt{I}_H$
$K \in \{2,4\}$	$\mathcal{N}$ voisinage 1-flip	$\mathtt{N} \in \{\hat{\mathtt{N}}^{(1,1)}_{hgh}, \hat{\mathtt{N}}^{(1,\star)}_{hgh}, \hat{\mathtt{N}}^{(\star,1)}_{hgh}, \hat{\mathtt{N}}^{(\star,\star)}_{hgh}\}$

TABLE 10.3 – Paramètres des expérimentations.

Les paramétrages testés se trouvent dans la table 10.3. Les instances testées peuvent être trouvées à l'adresse suivante : `http://mocolib.sf.net/`.

Nous nous sommes concentrés sur le climber de type SbLS le plus simple, dérivé de l'algorithme 10.1. Les variantes diffèrent principalement sur la classe de voisinage (sur les solutions-ensembles) utilisée. Voici le fonctionnement générique des différentes versions testées. À chaque pas de la recherche, une solution-ensemble est générée et évaluée. Elle est acceptée seulement si l'hypervolume associé à cet ensemble est strictement meilleur que celui de la solution-ensemble courante. Chaque variante ($\hat{\mathtt{N}}^{(1,1)}_{hgh}, \hat{\mathtt{N}}^{(1,\star)}_{hgh}, \hat{\mathtt{N}}^{(\star,1)}_{hgh}, \hat{\mathtt{N}}^{(\star,\star)}_{hgh}$) ajoute d'abord un certain nombre de solutions-éléments avant de réduire la taille de la solution-ensemble à μ en utilisant l'heuristique hgh (voir section 10.3.3). Afin d'éviter d'effectuer des calculs inutilement, les solutions-éléments dominées sont systématiquement supprimées des solutions-ensembles (elles ne peuvent contribuer à améliorer la valeur de l'hypervolume). Les quatre variantes testées sont dérivées des classes définies dans la section 10.3.1. $\forall \sigma \in \Sigma$:

1. $\hat{\mathtt{N}}^{(1,1)}_{hgh}(\sigma) = \{hgh(\sigma \cup \{x'\}) \ : \ x \in \sigma, x' \in \mathcal{N}(x)\}$,
2. $\hat{\mathtt{N}}^{(1,\star)}_{hgh}(\sigma) = \{hgh(\sigma \cup \mathcal{N}(x)) \ : \ x \in \sigma\}$,
3. $\hat{\mathtt{N}}^{(\star,1)}_{hgh}(\sigma) = \{hgh(\sigma \cup_{x\in\sigma} \{x'_x\}) \ : \ \forall x \in \sigma, x'_x \in \mathcal{N}(x)\}$,
4. $\hat{\mathtt{N}}^{(\star,\star)}_{hgh}(\sigma) = \{hgh(\sigma \cup_{x\in\sigma} \mathcal{N}(x))\}$.

Notons que la relation de voisinage sur les ensembles $\hat{\mathtt{N}}^{(\xi_1,\xi_2)}_{hgh}$ appartient à la classe de voisinages $\mathcal{C}(\xi_1,\xi_2)$. Chacune de ces relations de voisinage ajoute 1 ou N voisins de solution-élément(s) pour 1 ou μ solution-élément(s) en fonction de la classe de voisinage considérée. Comme discuté dans [Basseur *et al.*, 2013], ces différentes variantes de SbLS partagent certains aspects avec des algorithmes classiques tels que SEMO, PAES, PLS et SMS-EMOA.

Afin de situer les variantes de SbLS en termes de qualité générale, les résultats ont été comparés avec ceux obtenus par NSGA-II [Deb *et al.*, 2002]. L'espace de recherche des solutions-ensembles est $\Sigma_{\leqslant 100}$, ce qui limite la taille des solutions-ensembles à 100 solutions-éléments. Cela permet de comparer SbLS et NSGA-II dans des contextes similaires, 100 individus étant une taille appropriée pour NSGA-II [Deb *et al.*, 2002]. Les algorithmes ont été stoppés après 10^4 secondes, c'est-à-dire suffisamment pour laisser converger le plus possible les différentes variantes.

10.4.3 Résultats

La table 10.4 résume les résultats obtenus de la même manière que pour le reste du manuscrit (sur 30 exécutions par instance, test statistique de Mann-Whitney avec $p < 0{,}05$). Rappelons que les algorithmes de type SbLS sont quasiment identiques ; ils ne différent que par le voisinage des

ρ	-0,5			
N	256		512	
K	2	4	2	4
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,1)}$	0,5068	0,5189	0,4915	0,5186
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,\star)}$	0,5019	0,5124	0,4880	0,5106
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,1)}$	**0,5074**	0,5203	0,4897	0,5180
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,\star)}$	**0,5074**	**0,5231**	**0,4905**	**0,5187**
NSGA-II	0,4438	0,4806	0,4012	0,4440

ρ	0,0			
N	256		512	
K	2	4	2	4
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,1)}$	0,5161	**0,5453**	0,5155	0,5417
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,\star)}$	0,5129	0,5395	0,5095	0,5396
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,1)}$	0,5169	0,5444	0,5156	**0,5441**
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,\star)}$	**0,5193**	0,5433	**0,5159**	0,5430
NSGA-II	0,5032	0,5373	0,4791	0,5201

ρ	+0,5			
N	256		512	
K	2	4	2	4
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,1)}$	0,5264	0,5518	0,5213	0,5526
SbLS-$\hat{\mathtt{N}}_{hgh}^{(1,\star)}$	0,5231	0,5494	0,5161	0,5444
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,1)}$	0,5276	0,5539	0,5208	0,5528
SbLS-$\hat{\mathtt{N}}_{hgh}^{(\star,\star)}$	0,5296	0,5535	**0,5228**	0,5538
NSGA-II	**0,5333**	**0,5593**	0,5184	**0,5589**

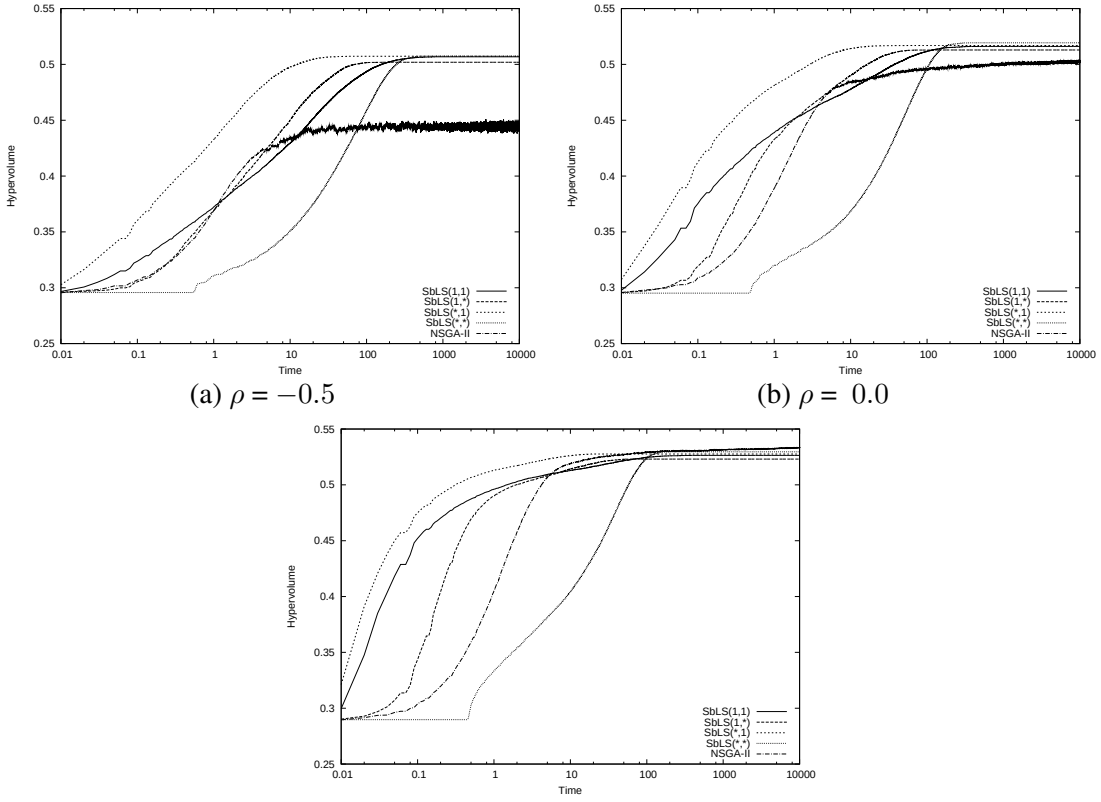
TABLE 10.4 – $\mathtt{I}_H$ moyen obtenu par les quatre variantes de SbLS et NSGA-II.

solutions-ensembles utilisé. De plus, les voisinages de solutions-ensembles définis sont tous basés sur le même voisinage de solutions-éléments (le 1-flip).

Globalement, les résultats obtenus par les variantes de SbLS sont assez proches les uns des autres en comparaison des résultats obtenus par NSGA-II. Leurs résultats sont d'ailleurs globalement compétitifs, surtout lorsque les objectifs sont corrélés négativement ou non corrélés. Lorsque les objectifs sont corrélés positivement, NSGA-II se montre plus performant (une corrélation fortement positive des objectifs indique que le problème considéré se rapproche d'un problème mono-objectif).

En se concentrant sur les différentes variantes de SbLS, qui diffèrent par la classe de voisinage de solutions-ensembles considérée, il est remarquable de noter que $\mathcal{C}(\star,\star)$ n'est jamais dominé statistiquement par une autre classe de voisinage. Ce résultat montre l'intérêt de définir des voisinages de solutions-ensembles larges, bien que le voisinage de solutions-éléments considéré reste fixe. La classe de voisinage $\mathcal{C}(1,\star)$ semble être globalement moins performante car elle est statistiquement dominée par au moins une des autres variantes pour l'ensemble des instances testées. Cela indique que focaliser la recherche de voisinage sur une seule solution-élément est pénalisant pour atteindre de bons optima locaux pour les paysages ρMNK.

Il est intéressant d'observer l'évolution de la convergence des quatre approches SbLS. La figure 10.1 montre l'évolution moyenne de l'hypervolume durant l'exécution des quatre variantes, pour les instances $N = 256$ et $K = 2$ et différentes valeurs de corrélation (-0,5, 0 et 0,5). Sans surprise, l'algorithme SbLS avec un voisinage de type $\mathcal{C}(\star,\star)$ est très coûteux en temps de calcul, ce qui fait que la convergence est très lente. En utilisant un voisinage de classe $\mathcal{C}(\star,1)$, SbLS converge très rapidement en atteignant de bons optima locaux, bien que ceux-ci soient en général très légèrement inférieurs à ceux obtenus par le voisinage $\mathcal{C}(\star,\star)$.



(a) $\rho = -0.5$ (b) $\rho = 0.0$

(c) $\rho = +0.5$

FIGURE 10.1 – Évolution moyenne de l'indicateur $\mathtt{I}_H$ au cours de l'exécution (en secondes), pour $N = 256$, $K = 2$ et $\rho \in \{-0.5, 0, +0.5\}$.

10.5 Conclusions

Ce chapitre propose une extension de l'étude de l'optimisation multiobjectif basée sur les ensembles introduite dans [Zitzler *et al.*, 2010]. En considérant que la solution attendue d'un problème d'optimisation multiobjectif est un ensemble de solutions compromis, on se rapproche alors des définitions usuelles utilisées en optimisation mono-objectif, la fonction d'évaluation étant alors un indicateur de qualité unaire sur les ensembles de compromis.

Nous avons défini l'algorithme générique SbLS qui est une recherche locale générique dépendant d'un indicateur unaire de qualité et d'une relation de voisinage sur les ensembles. Nous avons proposé plusieurs implémentation de SbLS basées sur l'indicateur d'hypervolume de dominance, ces variantes diffèrent par la classe de voisinage utilisée. Les résultats sur les paysages ρMNK montrent la relative performance de tels algorithmes et également que l'approche sur les solutions-ensembles permet d'imaginer de nouvelles métaheuristiques efficaces baséees sur des structures de voisinages peu explorées jusque-là.

Chapitre 11

Conclusions de la partie II

Au fil des années, je me suis de plus en plus intéressé à la conception de métaheuristiques multiobjecttif simples et génériques. Cela s'est traduit dans un premier temps par le réglage automatique de paramètres [Basseur *et al.*, 2002] au sein de métaheuristiques hybrides complexes, puis je me suis de plus en plus efforcé de simplifier les métaheuristiques elles-mêmes ainsi que le nombre de paramètres auquels elles sont soumises. Les approches basées sur les indicateurs de qualité m'ont permis de réaliser un grand pas dans cette direction grâce aux algorithmes IBMOLS et surtout HBMOLS. Leur simplicité a permis de pouvoir les éprouver facilement sur différents problèmes, sans avoir besoin de passer par une phase de paramètrage complexe.

Puis nous avons proposé l'algorithme SbLS, qui permet de ramener un problème d'optimisation multiobjectif en un problème d'optimisation mono-objectif sur des solutions-ensembles. Considérant un indicateur unaire de qualité éventuellement dépendant d'informations fournies par un décideur, SbLS est un algorithme simple, peu paramétré et qui permet de s'abstraire des difficultés liées à la nature multibjectif des problèmes étudiés. Considérant SbLS muni de l'indicateur unaire d'hypervolume s'avère en fait extrêmement proche de HBMOLS et est en fait une généralisation de l'optimisation multiobjectif basée sur l'hypervolume de dominance. Les travaux présentés dans cette partie nous amènent à formuler quelques perspectives de recherche.

Dans un premier temps, il serait intéressant d'étendre l'étude des recherches locales multiobjectif pour la résolution de problèmes de très grandes tailles. En effet, bien qu'IBMOLS et HBMOLS soient globalement performants, il atteignent leurs limites lorsque l'espace de recherche et le voisinage utilisés sont très grands, car l'algorithme peut prendre un temps considérable pour converger une seule fois vers un optimum local. C'est d'ailleurs le cas de beaucoup de métaheuristiques multiobjectif cherchant à découvrir un ensemble de solutions de compromis le plus large possible. En s'appuyant sur des travaux comme [Moalic *et al.*, 2013, Liefooghe *et al.*, 2014], l'objectif serait de modifier HBMOLS de telle sorte que la convergence soit rapide, surtout en début de recherche. Par exemple, il n'est peut être pas absolument nécessaire d'avoir un ensemble de solutions compromis très étendu dès le début de la recherche.

Il serait également intéressant d'approfondir le rapprochement réalisé entre l'optimisation multiobjectif et mono-objectif via l'optimisation multiobjectif basée sur des solutions-ensembles. Nous avons vu que les problèmes multiobjectif peuvent être formulés comme des problèmes mono-objectif où l'on cherche à maximiser la valeur d'un indicateur de qualité. Nous pouvons donc potentiellement appliquer toute métaheuristique mono-objectif à l'optimisation multiobjectif sur

les ensembles. Il serait intéressant d'analyser de telles extensions, tout en sachant qu'a priori, le comportement de telles métaheuristiques devrait être radicalement différent. En effet, les voisinages explorés sont en général très grands, et les paysages multiobjectif, formulés sous forme mono-objectif, comportent un haut taux de neutralité : considérant l'hypervolume comme fonction objectif, tout ajout ou suppression de solution-élément dominée laisse le fitness de la solution inchangé.

D'ailleurs, l'étude de paysages multiobjectif est un domaine où il reste certainement beaucoup de progrès à faire. En gardant l'aspect multiobjectif des paysages de recherche, les notions de neutralité et de rugosité ne sont pas triviales et se doivent d'être redéfinies dans ce contexte afin qu'elles gardent toute leur signification. Une autre approche possible est de baser l'analyse de paysage en se concentrant sur la transformation du problème multiobjectif initial en un problème d'optimisation multiobjectif basé sur les ensembles. Ainsi on peut potentiellement utiliser les indicateurs de caractérisation des paysages de recherche présentés dans la partie I. Malheureusement, en utilisant une évaluation basée sur l'hypervolume de dominance, les paysages engendrés sont extrêmement neutres. Cela indique peut-être qu'il pourrait être intéressant de proposer d'autres indicateurs de qualité qui permettent d'engendrer des paysages moins neutres, tout en gardant une rugosité la plus basse possible.

L'ensemble de mes travaux en optimisation multiobjectif partent du principe que nous n'avons aucune information concernant les préférences du décideur. La conséquence contraignante est qu'il est alors nécessaire de chercher à approximer la totalité du front de Pareto. Pourtant, dans le cadre réel, il est souvent possible d'avoir une idée plus ou moins précise de ces préférences. La connaissance d'informations provenant du décideur peut alors permettre de restreindre le champ des recherches à une zone réduite de l'espace des objectifs. Ce type d'étude a d'ailleurs été réalisé pour des métaheuristiques basées sur l'hypervolume de dominance, dans [Wagner et Trautmann, 2010] par exemple. De même, il serait intéressant d'étudier l'extensibilité de nos méthodes à d'autres définitions de la dominance, comme la dominance de Lorentz [Perny *et al.*, 2006]. Les métaheuristiques basées sur la dominance de Pareto comme NSGA-II [Deb *et al.*, 2002] peuvent a priori être adaptées assez facilement à d'autres définitions de la notion de dominance. De même, les approches basées sur les indicateurs de qualité devraient pouvoir être adaptées assez facilement, l'essentiel du travail consistant alors à proposer de nouveaux indicateurs permettant d'évaluer la qualité globale d'une approximation dans un contexte basé sur la dominance de Lorenz, par exemple.

D'une manière plus générale mes travaux futurs porterons essentiellement sur la conception de métaheuristiques génériques et autonomes, que ce soit en optimisation à un ou plusieurs objectifs. Ces dernières années, cette voie de recherche est devenue assez populaire grâce au développement du *paramétrage automatique* et des *hyperheuristiques* par exemple. Le paramétrage automatique [Hamadi *et al.*, 2012] consiste souvent à affiner les paramètres d'heuristiques durant l'exécution tandis que les hyperheuristiques [Burke *et al.*, 2003] consistent à automatiser le processus de sélection et de génération d'heuristiques en fonction des performances qu'elles obtiennent. Malheureusement le comportement à court terme des heuristiques n'est pas toujours représentatif de sa performance à long terme, comme nous l'avons vu pour la stratégie du pire améliorant des climbers. C'est pourquoi je pense que des études expérimentales alliées à l'analyse de paysages, de manière a priori ou dynamique, peuvent être un indicateur pertinent pour déterminer comment aborder un problème d'optimisation donné. Deux étapes principales sont à considérer. (1) La transformation du problème d'optimisation en paysage de recherche — Une analyse a priori

peut aider à définir le paysage à optimiser (espace de recherche, fonction d'évaluation, voisinage) en minimisant par exemple sa taille et sa rugosité, de manière à minimiser la complexité de son exploration ; (2) Explorer un paysage de recherche donné — Une fois le paysage défini, les indicateurs peuvent dynamiquement aider à déterminer le type d'approche de résolution à appliquer (quel type de recherche, avec quel paramètrage). Bien évidemment, les travaux présentés ici ne représentent qu'une petite étape vers une meilleure compréhension de ce domaine de recherche.

Bibliographie

[Aguirre et Tanaka, 2007] AGUIRRE, H. E. et TANAKA, K. (2007). Working principles, behavior, and performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, 181(3):1670–1690. *Cité page 146*

[Ahuja *et al.*, 2002] AHUJA, R. K., ERGUN, O., ORLIN, J. B. et PUNNEN, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102. *Cité page 145*

[Amit *et al.*, 1995] AMIT, N., JORGE, H. et SUNDERESH, H. (1995). Multiple and bicriteria scheduling : A literature survey. *European Journal of Operational Research*, 81(1):88–104. *Cité page 108*

[Auger *et al.*, 2012] AUGER, A., BADER, J., BROCKHOFF, D. et ZITZLER, E. (2012). Hypervolume-based multiobjective optimization : Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103. *Cité page 142*

[Bader et Zitzler, 2011] BADER, J. et ZITZLER, E. (2011). HypE : An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76. *Cité page 145*

[Barnett, 1998] BARNETT, L. (1998). Ruggedness and neutrality - the NKp family of fitness landscapes. *In Alive VI : Sixth International Conference on Articial Life*, pages 18–27. MIT Press. *3 citations pages 24, 28 et 29*

[Basseur et Burke, 2007] BASSEUR, M. et BURKE, E. K. (2007). Indicator-based multi-objective local search. *In Congress on Evolutionary Computation CEC'07*, Singapore. *Cité page 89*

[Basseur et Goëffon, 2013] BASSEUR, M. et GOËFFON, A. (2013). Hill-climbing strategies on various landscapes : an empirical comparison. *In* BLUM, C. et ALBA, E., éditeurs : *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 479–486. ACM. *2 citations pages 15 et 28*

[Basseur et Goëffon, 2014] BASSEUR, M. et GOËFFON, A. (2014). On the efficiency of worst improvement for climbing NK-landscapes. *In* ARNOLD, D. V., éditeur : *GECCO*, pages 413–420. ACM. *Cité page 15*

[Basseur *et al.*, 2013] BASSEUR, M., GOËFFON, A., LIEFOOGHE, A. et VEREL, S. (2013). On set-based local search for multiobjective combinatorial optimization. *In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 471–478. ACM. *3 citations pages 89, 146 et 147*

[Basseur *et al.*, 2004] BASSEUR, M., LEMESRE, J., DHAENENS, C. et TALBI, E.-G. (2004). Cooperation between branch and bound and evolutionary approaches to solve a biobjective flow

shop problem. *In Worshop on Evolutionary Algorithms (WEA'04)*, volume 3059, pages 72–86. ISBN : 3-540-22067-4. *Cité page 99*

[Basseur et Liefooghe, 2013] BASSEUR, M. et LIEFOOGHE, A. (2013). *Metaheuristics for production scheduling*, chapitre 9 : Metaheuristics for biobjective Flow Shop scheduling, pages 225–252. Wiley. *3 citations pages 89, 122 et 124*

[Basseur *et al.*, 2012a] BASSEUR, M., LIEFOOGHE, A., LE, K. et BURKE, E. K. (2012a). The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems. *Journal of Heuristics*, 18(2):263–296. *7 citations pages 89, 105, 107, 108, 109, 112 et 127*

[Basseur *et al.*, 2002] BASSEUR, M., SEYNHAEVE, F. et TALBI, E.-G. (2002). Design of multi-objective evolutionary algorithms : Application to the flow-shop scheduling problem. *In Congress on Evolutionary Computation CEC'02*, volume 2, pages 1151–1156, Honolulu, USA. *5 citations pages 98, 100, 110, 124 et 151*

[Basseur *et al.*, 2003] BASSEUR, M., SEYNHAEVE, F. et TALBI, E.-G. (2003). Adaptive mechanisms for multi-objective evolutionary algorithms. *In Congress on Engineering in System Application CESA'03*, numéro S3-R-00-222, pages 72–86, Lille, France. *Cité page 99*

[Basseur *et al.*, 2005a] BASSEUR, M., SEYNHAEVE, F. et TALBI, E.-G. (2005a). Path relinking in pareto multi-objective genetic algorithms. *In* COELLO COELLO, C. A., AGUIRRE, A. H. et ZITZLER, E., éditeurs : *Evolutionary Multi-Criterion Optimization, EMO'2005*, volume 3410 de *Lecture Notes in Computer Science*, pages 120–134, Guanajuato, Mexico. Springer-Verlag. ISBN : 3-540-24983-4. *6 citations pages 99, 130, 131, 132, 133 et 134*

[Basseur *et al.*, 2005b] BASSEUR, M., SEYNHAEVE, F. et TALBI, E.-G. (2005b). *Real-world Multi-objective System Engineering*, chapitre 6 : A Cooperative Metaheuristic Applied to Multi-Objective Flow-Shop Scheduling Problem. Nova Science. ISBN : 1-59454-390-9. *Cité page 99*

[Basseur *et al.*, 2012b] BASSEUR, M., ZENG, R.-Q. et HAO, J.-K. (2012b). Hypervolume-based multi-objective local search. *Neural Computing and Applications*, 21(8):1917–1929. *2 citations pages 89 et 120*

[Bateson, 1909] BATESON, W. (1909). *Mendel's Principles of Heredity*. Cambridge University Press. *Cité page 24*

[Beausoleil *et al.*, 2008] BEAUSOLEIL, R. P., BALDOQUIN, G. et MONTEJO, R. A. (2008). Multi-start and path relinking methods to deal with multiobjective knapsack problems. *Annals of Operations Research*, 157(1):105–133. *Cité page 130*

[Ben-Daya et Al-Fawzan, 1998] BEN-DAYA, M. et AL-FAWZAN, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, 109(1):88–95. *Cité page 52*

[Bentley et Wakefield, 1997] BENTLEY, P. J. et WAKEFIELD, J. P. (1997). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. *Soft Computing in Engineering Design and Manufacturing*, 5:231–340. *Cité page 105*

[Beume, 2009] BEUME, N. (2009). S-metric calculation by considering dominated hypervolume as klee's measure problem. *Evolutionary Computation*, 17(4):477–492. *Cité page 117*

[Beume *et al.*, 2007] BEUME, N., NAUJOKS, B. et EMMERICH, M. (2007). SMS-EMOA : Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669. *2 citations pages 145 et 146*

[Bradstreet *et al.*, 2010] BRADSTREET, L., WHILE, L. et BARONE, L. (2010). A fast many-objective hypervolume algorithm using iterated incremental calculations. *In 2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. *Cité page 117*

[Bringmann et Friedrich, 2008] BRINGMANN, K. et FRIEDRICH, T. (2008). Approximating the volume of unions and intersections of high-dimensional geometric objects. *In Proceedings of the 19th International Symposium Algorithms and Computation (ISAAC)*, volume 5369 de *Lecture Notes in Computer Science*, pages 436–447. Springer. *2 citations pages 116 et 145*

[Bringmann et Friedrich, 2009] BRINGMANN, K. et FRIEDRICH, T. (2009). Approximating the least hypervolume contributor : NP-hard in general, but fast in practice. *In Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, EMO'09, pages 6–20. Springer-Verlag. *Cité page 117*

[Burke *et al.*, 2003] BURKE, E., KENDALL, G., NEWALL, J., HART, E., ROSS, P. et SCHULENBURG, S. (2003). Hyper-heuristics : An emerging direction in modern search technology. *In Handbook of metaheuristics*, pages 457–474. Springer US. *Cité page 152*

[Cahon *et al.*, 2004] CAHON, S., MELAB, N. et TALBI, E. (2004). Paradiseo : A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380. *Cité page 86*

[Coello *et al.*, 2007] COELLO, C. C., LAMONT, G. B. et VAN VELDHUIZEN, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer. *Cité page 95*

[Conover, 1999] CONOVER, W. (1999). *Practical nonparametric statistics*. Wiley series in probability and statistics. Wiley, 3rd ed. édition. *Cité page 44*

[Cook, 1971] COOK, S. A. (1971). The complexity of theorem-proving procedures. *In Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158. ACM. *Cité page 34*

[Das et Dennis, 1996] DAS, I. et DENNIS, J. (1996). A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural and Multidisciplinary Optimization*. *Cité page 95*

[Deb, 2001] DEB, K. (2001). Multi-objective optimization. *Multi-objective optimization using evolutionary algorithms*. *Cité page 89*

[Deb *et al.*, 2002] DEB, K., PRATAP, A., AGARWAL, S. et MEYARIVAN, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197. *6 citations pages 96, 98, 108, 122, 147 et 152*

[Dorigo, 1992] DORIGO, M. (1992). *Optimization, Learning and Natural Algorithms*. Thèse de doctorat, Politecnico di Milano, Italy. *Cité page 8*

[Dréo *et al.*, 2006] DRÉO, J., PETROWSKI, A., Éric TAILLARD et SIARRY, P. (2006). *Metaheuristics for Hard Optimization*. Springer. *Cité page 19*

[Drugan et Thierens, 2010] DRUGAN, M. M. et THIERENS, D. (2010). Path-guided mutation for stochastic pareto local search algorithms. *In Parallel Problem Solving from Nature*, volume 6238 de *Lecture Notes in Computer Science*, pages 485–495. *Cité page 130*

[Edgeworth, 1881] EDGEWORTH, F. Y. (1881). *Mathematical Physics*. P. Keagan, London, England. *Cité page 89*

[Ehrgott, 2005] EHRGOTT, M. (2005). *Multicriteria Optimization*, volume 491 de *Lecture Notes in Economics and Mathematical Systems*. Springer. *Cité page 95*

[Fleischer, 2003] FLEISCHER, M. (2003). The measure of Pareto optima applications to multi-objective metaheuristics. *In* FONSECA, C. M., FLEMING, P. J., ZITZLER, E., THIELE, L. et DEB, K., éditeurs : *Evolutionary Multi-Criterion Optimization*, volume 2632 de *Lecture Notes in Computer Science*, pages 519–533. Springer Berlin Heidelberg. *Cité page 117*

[Fogel *et al.*, 1966] FOGEL, L., OWENS, A. et WALSH, M. (1966). *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK. *Cité page 7*

[Fonseca et Fleming, 1993] FONSECA, C. M. et FLEMING, P. J. (1993). Genetic algorithms for multiobjective optimization : Formulation discussion and generalization. *In Fifth International Conference on Genetic Algorithms (ICGA'93)*, pages 416–423, San Mateo, USA. *Cité page 105*

[Fourman, 1985] FOURMAN, M. P. (1985). Compaction of symbolic layout using genetic algorithms. *In First International Conference on Genetic Algorithm (ICGA)*, pages 141–153. Lawrence Erlbaum Associates. *Cité page 95*

[Gandibleux *et al.*, 2003] GANDIBLEUX, X., MORITA, H. et KATOH, N. (2003). Impact of clusters, path-relinking and mutation operators on the heuristic using a genetic heritage for solving assignment problems with two objectives. *Proceedings of The Fifth Metaheuristics International Conference MIC'03*. *Cité page 130*

[Garrett et Dasgupta, 2008] GARRETT, D. et DASGUPTA, D. (2008). Multiobjective landscape analysis and the generalized assignment problem. *In* MANIEZZO, V., BATTITI, R. et WATSON, J.-P., éditeurs : *Learning and Intelligent Optimization*, volume 5313 de *Lecture Notes in Computer Science*, pages 110–124. Springer Berlin Heidelberg. *Cité page 100*

[Gent *et al.*, 1999] GENT, I. P., HOOS, H. H., PROSSER, P. et WALSH, T. (1999). Morphing : Combining structure and randomness. *In* HENDLER, J. et SUBRAMANIAN, D., éditeurs : *AAAI/IAAI*, pages 654–660. AAAI Press / The MIT Press. *Cité page 63*

[Glover, 1977] GLOVER, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166. *Cité page 129*

[Glover, 1986] GLOVER, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549. *2 citations pages 7 et 8*

[Glover et Laguna, 2000] GLOVER, F. et LAGUNA, M. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684. *Cité page 129*

[Goldberg *et al.*, 1989] GOLDBERG, D. E. *et al.* (1989). *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park. *3 citations pages 8, 95 et 98*

[Hamadi *et al.*, 2012] HAMADI, Y., MONFROY, E. et SAUBION, F. (2012). *Autonomous Search*. Springer. *2 citations pages 9 et 152*

[Hansen et Mladenovic, 2006] HANSEN, P. et MLADENOVIC, N. (2006). First vs. best improvement : An empirical study. *Discrete Applied Mathematics*, 154(5):802–817. *Cité page 56*

[Holland, 1975] HOLLAND, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. *Cité page 8*

[Hoos et Stützle, 2004] HOOS, H. et STÜTZLE, T. (2004). *Stochastic Local Search : Foundations & Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. *Cité page 46*

[Jaszkiewicz et Zielniewicz, 2009] JASZKIEWICZ, A. et ZIELNIEWICZ, P. (2009). Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research*, 193(3):885–890. *Cité page 130*

[Jia et Hu, 2014] JIA, S. et HU, Z.-H. (2014). Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. *Computers & Operations Research*, 47(0):11–26. *Cité page 130*

[Jourdan *et al.*, 2009] JOURDAN, L., BASSEUR, M. et TALBI, E.-G. (2009). Hybridizing exact methods and metaheuristics : A taxonomy. *European Journal of Operation Research*, 199(3): 600–629. *Cité page 85*

[Juan *et al.*, 2013] JUAN, A. A., LOURENÇO, H. R., MATEO, M., LUO, R. et CASTELLA, Q. (2013). Using iterated local search for solving the flow-shop problem : Parallelization, parametrization, and randomization issues. *International Transactions in Operational Research*. *Cité page 52*

[Kauffman, 1993] KAUFFMAN, S. A. (1993). *The Origins of Order : Self-Organization and Selection in Evolution*. Oxford University Press, USA, 1 édition. *Cité page 28*

[Kirkpatrick *et al.*, 1983] KIRKPATRICK, S., GELATT, C. D. et VECCHI, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. *Cité page 8*

[Knowles et Corne, 2000] KNOWLES, J. D. et CORNE, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172. *Cité page 146*

[Landa Silva *et al.*, 2004] LANDA SILVA, D., BURKE, E. K. et PETROVIC, S. (2004). An introduction to multiobjective metaheuristics for scheduling and timetabling. *In* GANDIBLEUX, X., SEVAUX, M., SÖRENSEN, K. et T'KINDT, V., éditeurs : *Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, pages 91–129. Springer Berlin Heidelberg. *Cité page 108*

[Landa-silva et Le, 2008] LANDA-SILVA, D. et LE, K. N. (2008). A simple evolutionary algorithm with self-adaptation for multi-objective nurse scheduling. *In* COTTA, C., SEVAUX, M. et SÖRENSEN, K., éditeurs : *Adaptive and Multilevel Metaheuristics*, volume 136 de *Studies in Computational Intelligence*, pages 133–155. Springer Berlin Heidelberg. *Cité page 108*

[Laumanns *et al.*, 2002] LAUMANNS, M., THIELE, L., ZITZLER, E., WELZL, E. et DEB, K. (2002). Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. *In Conference on Parallel Problem Solving From Nature (PPSN VII)*, volume 2439 de *Lecture Notes in Computer Science*, pages 44–53, Granada, Spain. Springer. *Cité page 146*

[Lenstra *et al.*, 1977] LENSTRA, J. K., KAN, A. H. G. R. et BRUCKER, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362. *Cité page 32*

[Liefooghe *et al.*, 2012] LIEFOOGHE, A., HUMEAU, J., MESMOUDI, S., JOURDAN, L. et TALBI, E.-G. (2012). On dominance-based multiobjective local search : design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2):317–352. *2 citations pages 99 et 106*

[Liefooghe *et al.*, 2014] LIEFOOGHE, A., VEREL, S. et HAO, J.-K. (2014). A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming. *Applied Soft Computing*, 16(0):10–19. *Cité page 151*

[Malan et Engelbrecht, 2013] MALAN, K. et ENGELBRECHT, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163. *3 citations pages 22, 23 et 24*

[Manukyan *et al.*, 2014] MANUKYAN, N., EPPSTEIN, M. J. et BUZAS, J. S. (2014). NM landscapes : beyond NK. *In GECCO (Companion)*, pages 203–204. *Cité page 55*

[Merz et Freisleben, 1999] MERZ, P. et FREISLEBEN, B. (1999). Fitness landscapes and memetic algorithm design. *New ideas in optimization*, pages 245–260. *Cité page 19*

[Mitchell *et al.*, 1992] MITCHELL, M., FORREST, S. et HOLLAND, J. H. (1992). The royal road for genetic algorithms : Fitness landscapes and GA performance. *In Proceedings of the first european conference on artificial life*, pages 245–254. Cambridge : The MIT Press. *Cité page 19*

[Mladenović et Hansen, 1997] MLADENOVIĆ, N. et HANSEN, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100. *Cité page 73*

[Moalic *et al.*, 2013] MOALIC, L., CAMINADA, A. et LAMROUS, S. (2013). A fast local search approach for multiobjective problems. *In Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 294–298. Springer Berlin Heidelberg. *Cité page 151*

[Newman et Engelhardt, 1998] NEWMAN, M. E. J. et ENGELHARDT, R. (1998). Effects of selective neutrality on the evolution of molecular species. *Proceedings of the Royal Society B*, 265(1403):1333–1338. *2 citations pages 28 et 29*

[Ochoa *et al.*, 2010] OCHOA, G., VÉREL, S. et TOMASSINI, M. (2010). First-improvement vs. best-improvement local optima networks of NK landscapes. *In* SCHAEFER, R., COTTA, C., KOLODZIEJ, J. et RUDOLPH, G., éditeurs : *PPSN (1)*, volume 6238 de *Lecture Notes in Computer Science*, pages 104–113. Springer. *4 citations pages 36, 39, 55 et 64*

[Paquete *et al.*, 2004] PAQUETE, L., CHIARANDINI, M. et STÜTZLE, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem : An experimental study. *In* GANDIBLEUX, X., SEVAUX, M., SÖRENSEN, K. et T'KINDT, V., éditeurs : *Metaheuristics for Multiobjective Optimisation*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, pages 177–199. Springer Berlin Heidelberg. *Cité page 122*

[Paquete *et al.*, 2007] PAQUETE, L., SCHIAVINOTTO, T. et STÜTZLE, T. (2007). On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 156(1): 83–97. *Cité page 146*

[Pardalos et Wolkowicz, 1994] PARDALOS, P. P. M. et WOLKOWICZ, H. (1994). *Quadratic Assignment and Related Problems : Dimacs Workshop May 20-21, 1993*, volume 16. American Mathematical Soc. *Cité page 32*

[Pareto, 1896] PARETO, V. (1896). *Cours D'Economie Politique, volume I and II*. F. Rouge, Lausanne. *Cité page 89*

[Pasia *et al.*, 2007] PASIA, M., GANDIBLEUX, X., DOERNER, F. et HARTL, F. (2007). Local search guided by path relinking and heuristic bounds. *In 4th International Conference on Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 de *LNCS*, pages 501–515. Springer. *Cité page 130*

[Perny *et al.*, 2006] PERNY, P., SPANJAARD, O. et STORME, L.-X. (2006). A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147(1): 317–341. *Cité page 152*

[Poelwijk *et al.*, 2007] POELWIJK, F., KIVIET, D., WEINREICH, D. et TANS, S. (2007). Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445:383–386. *Cité page 24*

[Poelwijk *et al.*, 2011] POELWIJK, F., SORIN, T.-N., KIVIET, D. et TANS, S. (2011). Reciprocal sign epistasis is a necessary condition for multi-peaked fitness landscapes. *Journal of Theoretical Biology*, 272:141–144. *Cité page 54*

[Porumbel *et al.*, 2008] PORUMBEL, D., HAO, J.-K. et KUNTZ, P. (2008). A study of evaluation functions for the graph k-coloring problem. *In* MONMARCHÉ, N., TALBI, E.-G., COLLET, P., SCHOENAUER, M. et LUTTON, E., éditeurs : *Artificial Evolution*, volume 4926 de *Lecture Notes in Computer Science*, pages 124–135. Springer Berlin Heidelberg. *Cité page 84*

[Ratle, 2001] RATLE, A. (2001). Kriging as a surrogate fitness landscape in evolutionary optimization. *AI EDAM*, 15(01):37–49. *Cité page 19*

[Rechenberg, 1965] RECHENBERG, I. (1965). Cybernetic solution path of an experimental problem. Rapport technique, Royal Aircraft Establishment Library Translation 1122, Farnborough. *Cité page 7*

[Reidys *et al.*, 1998] REIDYS, C. M., PETER et STADLER, P. F. (1998). Neutrality in fitness landscapes. *Applied Mathematics and Computation*, 117:187–207. *Cité page 23*

[Sahni et Gonzalez, 1976] SAHNI, S. et GONZALEZ, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565. *Cité page 32*

[Schaffer, 1985] SCHAFFER, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *In First International Conference on Genetic Algorithms (ICGA)*, pages 93–100. Lawrence Erlbaum Associates. *Cité page 95*

[Schiavinotto et Stützle, 2011] SCHIAVINOTTO, T. et STÜTZLE, T. (2011). A review of metrics on permutations for search landscape analysis. *Computers and Operations Research*, 34(10):3143–3153. *Cité page 134*

[Seridi *et al.*, 2013] SERIDI, K., JOURDAN, L. et TALBI, E.-G. (2013). Multi-objective path relinking for biclustering : application to microarray data. *In EMO'2013 Evolutionary Multi-objective Optimization*, pages 200–214. *Cité page 130*

[Smith *et al.*, 2002] SMITH, T., HUSBANDS, P., LAYZELL, P. et O'SHEA, M. (2002). Fitness landscapes and evolvability. *Evolutionary Computation*, 10:1–34. *2 citations pages 23 et 27*

[Sörensen, 2013] SÖRENSEN, K. (2013). Metaheuristics — the metaphor exposed. *International Transactions in Operational Research*, pages n/a–n/a. *2 citations pages 8 et 9*

[Srinivas et Deb, 1994] SRINIVAS, N. et DEB, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248. *Cité page 105*

[Steuer, 1989] STEUER, R. E. (1989). *Multiple criteria optimization : theory, computation, and application*. Krieger Malabar. *Cité page 89*

[Taillard, 1990] TAILLARD, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74. *Cité page 52*

[Taillard, 1993] TAILLARD, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285. *2 citations pages 32 et 109*

[Talbi, 2009] TALBI, E.-G. (2009). *Metaheuristics : From Design to Implementation*. Wiley Publishing. *Cité page 19*

[Talbi *et al.*, 2012] TALBI, E.-G., BASSEUR, M., NEBRO, A. J. et ALBA, E. (2012). Multi-objective optimization using metaheuristics : non-standard algorithms. *International Transactions in Operational Research*, 19(1-2):283–305. *Cité page 98*

[Talbi *et al.*, 2001] TALBI, E.-G., RAHOUAL, M., MABED, H. et DHAENENS, C. (2001). A hybrid evolutionary approach for multicriteria optimization problems : Application to the flow shop. *In 1st International Conference on Evolutionary Multi-Criterion Optimization - EMO*

2001, volume 1993 de *Lecture Notes in Computer Science*, pages 416–428, Zurich, Switzerland. *Cité page 108*

[Thierens, 2006] THIERENS, D. (2006). Exploration and exploitation bias of crossover and path relinking for permutation problems. *In Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, volume 4193 de *Lecture Notes in Computer Science*, pages 1028–1037. Springer. *Cité page 134*

[T'Kindt et Billaut, 2006] T'KINDT, V. et BILLAUT, J.-C. (2006). *Multicriteria Scheduling - Theory, Models and Algorithms (2. ed.).* Springer. *Cité page 108*

[Traverson, 2014] TRAVERSON, H. (2014). Recherche locale et neutralité dans les paysages de fitness (master thesis report). Rapport technique, Université d'Angers. *3 citations pages 78, 80 et 82*

[Vanneschi *et al.*, 2006] VANNESCHI, L., PIROLA, Y. et COLLARD, P. (2006). A quantitative study of neutrality in GP boolean landscapes. *In Genetic and Evolutionary Computation Conference (GECCO)*, pages 895–902. ACM. *Cité page 23*

[Verel *et al.*, 2011a] VEREL, S., LIEFOOGHE, A. et DHAENENS, C. (2011a). Set-based multiobjective fitness landscapes : a preliminary study. *In 13th conference on Genetic and Evolutionary Computation Conference (GECCO 2011)*, pages 769–776. ACM. *2 citations pages 100 et 140*

[Verel *et al.*, 2011b] VEREL, S., LIEFOOGHE, A., JOURDAN, L. et DHAENENS, C. (2011b). Analyzing the effect of objective correlation on the efficient set of MNK-landscapes. *In Learning and Intelligent OptimizatioN (LION 5)*, volume 6683 de *Lecture Notes in Computer Science*, pages 238–252. Springer. *Cité page 146*

[Vigneron *et al.*, 2014] VIGNERON, V., BASSEUR, M., GOËFFON, A., LARDEUX, F. et SAUBION, F. (2014). On the attainability of nk landscapes global optima. *In* HELMERT, M. et RÖGER, G., éditeurs : *SOCS*. AAAI Press. *3 citations pages 15, 37 et 38*

[Wagner et Trautmann, 2010] WAGNER, T. et TRAUTMANN, H. (2010). Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation*, 14(5):688–701. *Cité page 152*

[While *et al.*, 2012] WHILE, L., BRADSTREET, L. et BARONE, L. (2012). A fast way of calculating exact hypervolumes. *Transactions on Evolutionary Computation*, 16(1):86–95. *Cité page 117*

[While *et al.*, 2006] WHILE, L., HINGSTON, P., BARONE, L. et HUBAND, S. (2006). A faster algorithm for calculating hypervolume. *Transactions on Evolutionary Computation*, 10(1):29–38. *Cité page 117*

[Whitley *et al.*, 2013] WHITLEY, D., HOWE, A. E. et HAINS, D. (2013). Greedy or not ? best improving versus first improving stochastic local search for maxsat. *In AAAI*. *Cité page 54*

[Wolpert et Macready, 1997] WOLPERT, D. H. et MACREADY, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82. *Cité page 8*

[Wright, 1932] WRIGHT, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the VI International Congress of Genetics*, 1:356–366. *Cité page 19*

[Wu et Azarm, 2001] WU, J. et AZARM, S. (2001). Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set. *Journal of Mechanical Design*, 123(1):18–25. *Cité page 117*

[Zeng, 2012] ZENG, R.-Q. (2012). *Métaheuristiques multiobjectif basées sur des voisinages pour l'approximation d'ensembles de Pareto*. Thèse de doctorat. *Cité page 118*

[Zeng *et al.*, 2013a] ZENG, R.-Q., BASSEUR, M. et HAO, J.-K. (2013a). Hypervolume-based multi-objective path relinking algorithm. *In Evolutionary Multi-criterion Optimization (EMO 2013)*, volume 4403 de *Lecture Notes in Computer Science*, pages 185–199. Springer-Verlag. *Cité page 89*

[Zeng *et al.*, 2013b] ZENG, R.-Q., BASSEUR, M. et HAO, J.-K. (2013b). Solving bi-objective flow shop problem with hybrid path relinking algorithm. *Applied Soft Computing*, 13(10):4118–4132. *3 citations pages 89, 134 et 135*

[Zhang et Stickel, 2000] ZHANG, H. et STICKEL, M. (2000). Implementing the davis–putnam method. *Journal of Automated Reasoning*, 24(1-2):277–296. *Cité page 63*

[Zitzler *et al.*, 2008] ZITZLER, E., KNOWLES, J. et THIELE, L. (2008). Quality Assessment of Pareto Set Approximations. *In* BRANKE, J., DEB, K., MIETTINEN, K. et SLOWINSKI, R., éditeurs : *Multiobjective Optimization : Interactive and Evolutionary Approaches*, pages 373–404. Springer. *Cité page 96*

[Zitzler et Künzli, 2004] ZITZLER, E. et KÜNZLI, S. (2004). Indicator-based selection in multiobjective search. *In Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer. *8 citations pages 89, 96, 103, 104, 105, 108, 122 et 123*

[Zitzler *et al.*, 2001] ZITZLER, E., LAUMANNS, M. et THIELE, L. (2001). SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. Rapport technique 103, Eidgenössische Technische Hochschule Zürich (ETH). *2 citations pages 96 et 98*

[Zitzler et Thiele, 1999] ZITZLER, E. et THIELE, L. (1999). Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271. *Cité page 96*

[Zitzler *et al.*, 2010] ZITZLER, E., THIELE, L. et BADER, J. (2010). On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79. *4 citations pages 139, 142, 145 et 149*

[Zitzler *et al.*, 2003] ZITZLER, E., THIELE, L., LAUMANNS, M., FONSECA, C. M. et GRUNERT DA FONSECA, V. (2003). Performance assessment of multiobjective optimizers : An analysis and review. *IEEE Trans. on Evolutionary Computation*, 7(2):117–132. *3 citations pages 97, 103 et 109*