Université Lille 1
École Doctorale Sciences Pour l'Ingénieur Université Lille Nord-de-France
Laboratoire d'Informatique Fondamentale de Lille (UMR CNRS 8022)
Centre de Recherche INRIA Lille - Nord Europe

Mémoire de thèse présenté pour obtenir le grade de
Docteur de l'Université Lille 1
Discipline : Informatique

Recherche locale et optimisation combinatoire : De l'analyse structurelle d'un problème à la conception d'algorithmes efficaces

Marie-Éléonore Marmion

Date de soutenance : 9 décembre 2011

Membres du jury

Président :	Laurence Duchien	-	Professeur des Universités, Lille 1
Rapporteurs :	Alexandre Caminada	-	Professeur des Universités, UTBM
	Frédéric Saubion	-	Professeur des Universités, Angers
Examinateurs :	Thomas Stützle	-	Professeur des Universités, ULB
	Sébastien Verel	-	Maître de Conférences, Nice-Sophia Antipolis
Directeurs :	Clarisse Dhaenens	-	Professeur des Universités, Lille 1
	Laetitia Jourdan	-	Professeur des Universités, Lille 1

Numéro d'ordre : 40633 | Année : 2011

Remerciements

Mon parcours en doctorat a débuté, s'est déroulé et se finit par de nombreuses et jolies rencontres professionnelles et personnelles. Il a été facilité par le soutien de ma famille, de mes amis et de mes collègues. Pour tout ceci, je souhaite commencer ce mémoire en remerciant toutes ces personnes.

Tout d'abord, je tiens à remercier Clarisse et Laetitia, mes directrices de thèse, de m'avoir permis d'effectuer ces trois années de doctorat dans de très bonnes conditions. Vos réflexions scientifiques, vos conseils, votre soutien et votre confiance ont été un véritable moteur au bon déroulement de mon doctorat.

Je souhaite également remercier les membres du jury : Laurence Duchien, présidente, Alexandre Caminada et Frédéric Saubion, rapporteurs, et Thomas Stützle et Sébastien Verel, examinateurs de cette thèse, pour l'intérêt porté à mon travail.

Je tiens aussi à remercier l'ensemble des membres de l'équipe DOLPHIN pour votre sympathie, vos remarques et vos conseils tout au long de ces années. Merci à l'ensemble des permanents pour votre accueil : Bilel, François, Luce, Nouredine et El-Ghazali Talbi. Je remercie tout particulièrement, Arnaud (alias *Docteur*) et Seb pour nos nombreuses heures de discussions, de réflexions, d'implémentation, d'analyse et pour m'avoir encouragée en voulant "voir les courbes". Cette collaboration a été très appréciable. Enfin, je pense aussi à tous les autres, actuels ou anciens membres, qui ont égayé ces trois années : Ali, Imen, Ines (alias *Sousse*), Jean-Charles, Jean-Claude, Jérémie (un grand merci), Julie H (alias *Juju*), Julie J, Gaël, Karima (alias *Mamati*), Khedidja, Mahmoud, Martin, Mathieu, Mostepha (alias *La tortue*), Moustapha (alias *Papa Moustapha*), Nadia (alias *Mon p'tit boudin bleu canard*), Salma, Sezin, Tuan (alias *Tuan-Tuan*), Thé Van (alias *Ma p'tite crevette* ou *BG Boy*), Waldo, Yacine (alias *(Mon p'tit) Yaya*).

Mes années de doctorat s'accompagnent de nombreux amis. Sans eux, ces trois années n'auraient pas été aussi joyeuses. Merci à mes "vrais" amis, les dits *parisiens-tourangeaux*, pour ces voyages, ces nombreux week-end parisiens... et d'être venus (si souvent) affronter le climat lillois : Adé, Zéty, Raphou, Poop, Mimi et les autres. Merci à mes "faux" amis, les dits *lillois*, pour votre présence au quotidien : les Forils (Louise, Mélanie, Cécile), les (ex-)INRIA (Alban, Florian, Fred, Greg, Guillaume, Martin, Patrick) et les autres.

Cette page s'achève avec mes remerciements pour ma famille. Tout d'abord, ma *grande* famille : grands-parents, oncles, tantes, cousins et cousines, qui m'ont et me soutiennent par l'affection qu'ils me portent. Ensuite, je remercie mes frérots, Aurel-Aimé et François-Yves, d'être présents et de me soutenir, encore et toujours. Ces quelques week-ends passés tous les trois ont été des moments de joie et de détente en famille vraiment appréciables. Enfin, je souhaite remercier mes parents qui m'ont toujours soutenu, qui ont accepté et assumé ma volonté de faire de longues études, qui n'ont jamais compté les kilomètres. Bref, merci pour tout ce que vous avez fait pour moi, pour votre attention, votre réconfort et pour tout cet amour que je reçois.

Table des matières

Introduction

Cette thèse s'inscrit dans le domaine de la recherche opérationnelle et de l'optimisation combinatoire. L'objectif de cette thèse est d'étudier la structure d'un problème d'optimisation afin de concevoir des méthodes de résolution adaptées et efficaces. Cette thèse a été menée au sein de l'équipe DOLPHIN [1] du laboratoire d'informatique fondamentale de Lille (LIFL, CNRS, Université Lille 1), commune à l'équipe-projet du même nom du centre de recherche INRIA Lille-Nord Europe. Cette équipe a pour but de modéliser et de résoudre des problèmes d'optimisation combinatoires (multiobjectif) de grande taille, en utilisant des techniques hybrides, de parallélisation et de calcul distribué. Les thématiques de l'équipe portent sur la mise en œuvre conjointe des méthodes avancées issues de l'optimisation combinatoire en recherche opérationnelle, de la décision en intelligence artificielle et du calcul parallèle.

De nombreux problèmes réels peuvent s'exprimer sous la forme d'un problème d'optimisation combinatoire. Un problème d'optimisation combinatoire consiste, dans un espace discret de solutions réalisables, à trouver la meilleure solution possible (ou l'ensemble des meilleures solutions). La notion de meilleur est donnée par un critère de qualité via une fonction objectif. Ces problèmes sont généralement NP-difficiles et les méthodes exactes demeurent inefficaces pour les résoudre rapidement lorsqu'ils sont de grandes tailles.
Les métaheuristiques sont des méthodes génériques de résolution de problèmes d'optimisation, connues et utilisées pour leur efficacité. Il en existe deux types : les recherches locales, méthodes à solution unique, et les méthodes à base de population. Ces métaheuristiques possèdent généralement plusieurs paramètres difficiles à régler. En effet, une même configuration d'une métaheuristique ne peut être efficace pour toutes les instances d'un même problème. Le travail de réglage des paramètres est fastidieux et certains travaux portent sur la manière dont il serait possible de rendre ce travail plus évident, voir de l'automatiser.

Le paysage d'un problème d'optimisation combinatoire est défini à partir d'une notion de voisinage entre solutions réalisables, donnée par une relation binaire, et d'un critère de qualité de ces solutions, donné par la fonction objectif. Le paysage correspond alors à un modèle de structure du problème traité. L'opérateur de voisinage est similaire à celui utilisé par les métaheuristiques. De plus en plus, l'étude du paysage est vue comme un moyen de suivre la dynamique d'une métaheuristique au cours du processus de recherche. Dans la littérature, on trouve principalement des analyses de paysages de problèmes académiques. Malgré encore peu de travaux, l'analyse conjointe du paysage et des performances d'une métaheuristique est en plein développement. Dans ce mémoire, nous proposons d'apporter une contribution à cette thématique en étudiant le paysage de problèmes d'optimisation proches du monde réel et le lien étroit avec certaines classes de métaheuristiques.
Ainsi, nous basant sur quelques indicateurs de paysage de la littérature, nous avons étudié l'impact de la structure de paysage d'un problème d'optimisation sur la dynamique d'une

1. Optimisation multi-critère parallèle coopérative.

métaheuristique, pour deux problèmes issus de la logistique. Ayant conscience de ce lien, nous nous sommes alors demandés comment l'analyse de paysage pouvait permettre d'obtenir de bonnes performances et donc de paramétrer et/ou concevoir des métaheuristiques efficaces.

Une caractéristique importante et qui va jouer un rôle prépondérant dans cette thèse est que certains problèmes d'optimisation possèdent de nombreuses solutions de qualité identique. La structure de paysage est alors qualifiée de neutre. Ces paysages neutres présentent de nombreux plateaux formés par des solutions voisines de même qualité. Cependant, ces plateaux représentent des barrières bloquant la progression d'une recherche locale. L'analyse de la structure neutre nous a incité à tirer parti de ces plateaux pour améliorer la recherche. Nous avons alors, dans cette thèse, proposé deux schémas de méthodes exploitant la neutralité d'un problème d'optimisation combinatoire. Le premier modèle est une recherche locale simple à utiliser et très efficace. Celle-ci a pour but d'exploiter la neutralité des optima locaux du paysage. Ainsi, lorsqu'elle se trouve bloquée sur un optimum local, elle tente de s'en échapper en profitant de son plateau.

Nous nous sommes ensuite intéressés à la neutralité présente à tous les niveaux du paysage. Les plateaux étant généralement larges, nous nous sommes demandés comment guider efficacement la recherche sur les plateaux. Cette stratégie de guidage a pour objectif de se diriger rapidement vers les solutions d'un même plateau ayant au moins un voisin améliorant. Nous avons utilisé la mesure d'évolvabilité comme une mesure pouvant traduire la capacité d'une solution du plateau à avoir un voisin améliorant.
À partir d'une mesure d'évolvabilité, nous avons pu définir une méthode de sélection, tenant compte de l'évolvabilité de chacune des solutions du plateau pour choisir la plus prometteuse. Ainsi, nous avons conçu une deuxième recherche locale efficace pour une mesure d'évolvabilité bien choisie.

Dans cette thèse, nous prouvons l'intérêt d'étudier les structures de paysage d'un problème d'optimisation combinatoire pour le résoudre efficacement. En effet, les caractéristiques d'une structure permettent de comprendre la dynamique de recherche d'une métaheuristique et peuvent être exploiter pour concevoir et paramétrer des métaheuristiques efficaces.

Plan du mémoire

Chapitre 1

Le premier chapitre décrit le contexte scientifique des travaux de ce mémoire. Aussi, nous présentons l'optimisation combinatoire et les métaheuristiques. Le paysage d'un problème d'optimisation est au cœur de nos travaux. Nous commençons par donner les principales définitions correspondantes, puis nous présentons notre état de l'art des indicateurs de la littérature pour analyser le paysage. Enfin, nous présentons les trois problèmes d'optimisation, étudiés dans ce mémoire.

Chapitre 2

Le second chapitre s'intéresse au lien entre le paysage d'un problème et les performances de certaines métaheuristiques. Pour deux problèmes classiques issus de la logistique, un problème asymétrique de tournées de véhicules avec flotte fixe et hétérogène et un problème d'ordonnancement de type flowshop de permutation, nous avons étudié deux structures de paysage différenciées par leur relation de voisinage. Ainsi, la comparaison des structures pour un même problème nous a amené à nous demander si cette structure de paysage impacte la progression d'une métaheuristique. Puis, pour chacun des deux problèmes, nous avons analysé les performances de deux configurations de métaheuristiques classiques de la littérature, l'une avec l'opérateur de voisinage du premier paysage et l'autre avec celui du second. Nous finissons par une analyse croisée des résultats des deux problèmes quant à l'influence du paysage sur les performances des métaheuristiques.

Chapitre 3

Dans le troisième chapitre, nous nous intéressons aux structures neutres que peuvent présenter certains problèmes d'optimisation tels que le problème de flowshop de permutation ou le problème académique des paysages-NKq. Nous commençons par définir la neutralité dans la structure de paysage et donnons les indicateurs qui permettent de l'analyser. Ensuite, nous étudions les paysages des problèmes de flowshop et des paysages-NKq selon les critères de neutralité. Cette étude nous pousse à concevoir une recherche locale simple et efficace que nous appliquons sur ces deux problèmes.

Chapitre 4

Le quatrième chapitre est guidé par la volonté de profiter de la neutralité du paysage sans se focaliser sur les optima locaux. Ainsi, nous commençons par définir une mesure d'évolvabilité pour apprécier la capacité d'une solution à avoir un voisin améliorant. Considérant ainsi plusieurs solutions d'un même plateau, cette mesure d'évolvabilité sert à les différencier. Nous avons alors conçu une métaheuristique en adaptant une stratégie de sélection d'opérateurs issue des méthodes évolutionnaires à la sélection de la solution la plus prometteuse du plateau grâce à une mesure d'évolvabilité. Cette méthode est alors analysée sur les problèmes de flowshop et des paysages-NKq.

Chapitre 5

Le cinquième chapitre se veut un chapitre de discussion sur les travaux évoqués dans le mémoire. Ainsi, nous exposons nos principales contibutions. Puis, une analyse transversale des chapitres 3 et 4 nous amène à réfléchir à la conception d'une nouvelle recherche locale que nous présentons et appliquons au problème de flowshop de permutation. Nous finissons par présenter plusieurs perspectives aux travaux présentés au long de ce mémoire.

Chapitre 1

Contexte

Sommaire

Ce chapitre est destiné à présenter les notions et les outils nécessaires à la bonne compréhension de cette thèse. Nous commençons par donner les définitions liées à l'optimisation combinatoire, puis nous définissons le paysage d'un problème d'optimisation combinatoire et présentons des outils existants de la littérature pour l'analyser. Enfin, nous présentons trois types de problèmes classiques d'optimisation combinatoire que nous utilisons dans cette thèse, pour valider notre travail.

1.1 Optimisation combinatoire

L'optimisation combinatoire tient une place importante dans la recherche opérationnelle, en mathématiques discrètes et en informatique. En effet, les problèmes d'optimisation combinatoire représentent une catégorie de problèmes très difficiles à résoudre [PS82] et de nombreux problèmes pratiques peuvent être formulés sous la forme d'un problème d'optimisation [Rib94]. De plus, l'étude de ces problèmes représente un gain non négligeable sur la qualité de leur résolution. Un problème d'optimisation combinatoire, consiste dans un espace discret (*i.e.* énumérable) de solutions réalisables, à trouver la meilleure solution (ou un ensemble des meilleures solutions). La notion de meilleur est donnée par un critère de qualité via une fonction objectif. Formellement, un problème d'optimisation combinatoire peut être défini par, Ω l'ensemble discret des solutions réalisables du problème, on parle alors d'espace de recherche, et $f : \Omega \longrightarrow \mathbb{R}$ la fonction objectif associée au critère de qualité. Le but est de trouver $s^* \in \Omega$ tel que :

$$s^* = \arg\max_{s \in \Omega} \{f(s)\}$$

On parle d'un problème de maximisation quand la qualité est donnée par une fonction objectif à maximiser et d'un problème de minimisation quand la qualité est donnée par une fonction objectif à minimiser.
La *valeur de fitness* d'une solution de l'espace de recherche est la valeur de la fonction objectif pour cette solution. L'*évaluation* d'une solution correspond au calcul de sa valeur de fitness.
Dans ce contexte, nous définissons les termes d'*optimum global* et d'*optimum local* associés à un problème d'optimisation. Pour un problème de minimisation (resp. maximisation), un optimum global est une solution $s^* \in \Omega$ telle que :

$$\forall s \in \Omega, f(s^*) \leq f(s)$$

(resp. $f(s^*) \geq f(s)$).
Si l'espace de recherche Ω est muni d'une relation de voisinage $\mathcal{V}$. Un optimum local est alors défini comme une solution $s^* \in \Omega$ telle que :

$$\forall s \in \mathcal{V}(s^*), f(s^*) \leq f(s)$$

(resp. $f(s^*) \geq f(s)$).
On parle d'*optima stricts* si les inégalités précédentes sont strictes pour $s \neq s^*$.
Les problèmes d'optimisation combinatoire sont pour la plupart d'une grande difficulté [PS82] mais ont de nombreuses applications pratiques [Rib94] telles que le transport, la gestion,

l'énergie, l'ingénierie... Par exemple, un commerçant doit livrer des articles chez ses clients partout en France et souhaite minimiser le coût lié à la livraison. Il dispose de plusieurs camions et doit affecter à chacun d'eux une liste de clients à livrer. Ceci peut être modélisé par un problème d'optimisation combinatoire où l'ensemble des solutions réalisables correspond à tous les chemins de livraisons possibles et où la distance parcourue par l'ensemble des camions doit être minimale. Ce problème d'optimisation, identifié comme un problème de tournées de véhicules, voit sa difficulté s'accroître à mesure que le nombre de clients augmente.

Une instance d'un problème d'optimisation correspond à des données particulières du problème à résoudre. Dans l'exemple précédent, une instance du problème peut correspondre à une liste de clients et, d'articles à livrer et un ensemble de véhicules disponibles le jour considéré. Le jour suivant, l'instance sera différente et la résolution du problème également.

Une grande partie des problèmes d'optimisation combinatoire font partie des problèmes NP-difficiles pour lesquels il n'existe pas d'algorithme de résolution efficace pour toutes les instances [GJ90]. Il existe deux classes de méthodes pour résoudre ces problèmes d'optimisation, les méthodes :

– exactes : elles garantissent la complétude de la résolution,
– approchées : elles perdent la complétude mais gagnent en efficacité.

Ces méthodes sont souvent testées au préalable sur des instances de la littérature. Ces instances permettent de comparer les méthodes et leur efficacité. On appelle *best-known*, la meilleure solution connue d'une instance. Cette solution peut être la meilleure parmi les solutions de l'espace de recherche, c'est donc une solution optimale, ou la meilleure solution trouvée.

Les méthodes exactes énumèrent implicitement toutes les solutions de l'espace de recherche en utilisant des mécanismes qui détectent des échecs (calcul de bornes) et des heuristiques spécifiques au problème qui orientent les choix. Ces méthodes ont permis de trouver des solutions optimales. Mais ces méthodes s'avèrent, malgré les progrès réalisés, plutôt inefficaces à mesure que la taille du problème devient importante.

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale. En effet, elles permettent d'obtenir des solutions de qualité intéressante en un temps de calcul réduit. Les métaheuristiques font partie de ces méthodes approchées [KGH00]. Les métaheuristiques sont composées de concepts fondamentaux qui permettent de les adapter et de les appliquer à une large classe de problème d'optimisation.

1.2 Métaheuristiques classiques

Il existe deux types de métaheuristiques : les recherches locales, aussi appelées méthodes à solution unique, et les méthodes à population de solutions. Les recherches locales ont tendance à intensifier la recherche en exploitant une partie de l'espace de recherche alors que les méthodes à population de solutions ont plutôt tendance à la diversifier en explorant différentes parties de l'espace de recherche. Dans la suite, nous présentons plus spécifiquement certaines métaheuristiques que nous utilisons dans cette thèse. Précisons qu'on se place dans un contexte de minimisation.

1.2.1 Les recherches locales

L'espace de recherche associé à un problème d'optimisation combinatoire est souvent non énumérable en un temps raisonnable. On essaie donc de relier certaines solutions entre elles. Ainsi, à partir d'une solution, on peut en trouver une autre et ainsi de suite. Il est nécessaire de définir une relation de voisinage (*cf.* section 1.3.2) qui est une application qui associe à toute solution de l'espace de recherche un voisinage *i.e.* un ensemble de solutions (ne la contenant pas elle-même) appelées voisins. Les recherches locales sont des méthodes fondées sur une relation de voisinage et sur une procédure exploitant ce voisinage. Les recherches locales se différencient par la procédure d'exploitation du voisinage, le voisinage pouvant être considéré comme un paramètre de celle-ci.

1.2.1.1 Les méthodes de descente

Les méthodes de descente sont classiques et très rapides [Pap76, PS82]. La procédure exploitant le voisinage consiste à (i) débuter avec une solution initiale s de Ω et (ii) choisir une solution voisine s' de s telle que $f(s') \leq f(s)$, puis à remplacer s par s' et répéter (ii) jusqu'à ce que pour tout voisin s' de s, $f(s) \leq f(s')$. s est alors un optimum local.
Il existe plusieurs procédures de choix d'un meilleur voisin, dont deux qui sont très utilisées. Le *Hill Climbing* (HC) consiste à choisir le voisin de la solution courante ayant la meilleure qualité (exploration exhaustive du voisinage). Le *First Improvement Hill Climbing* (FIHC) consiste à choisir le premier voisin rencontré qui a une meilleure qualité (exploration partielle du voisinage). Le *Netcrawler* [Bar01] se différencie des méthodes de descente traditionnelles en acceptant également les solutions de qualité égale. Ainsi, cette méthode choisit le premier voisin qui a une qualité meilleure ou équivalente.

1.2.1.2 Les recherches locales itérées (Iterated Local Search, ILS)

Les méthodes de descentes sont rapides et simples à mettre en œuvre mais n'aboutissent généralement pas aux meilleurs optima car elles s'arrêtent dès qu'un optimum local est trouvé. Pour ne pas rester bloqué sur cet optimum local, il existe différentes stratégies pour continuer. Ces stratégies permettant de poursuivre la recherche après avoir trouvé un optimum, il faut alors définir un critère d'arrêt. Les critères d'arrêt courants sont le temps d'exécution, le nombre d'itérations, le nombre d'évaluations total...
Une première stratégie pour pallier l'arrêt brutal de la recherche sur un optimum local est d'itérer la méthode de descente. On déroule les étapes suivantes à partir de l'optimum trouvé : (i) appliquer une perturbation sur la solution courante (ii) appliquer une méthode de descente sur cette solution (iii) choisir via un critère d'acceptation si le nouvel optimum devient la solution courante et revenir en (i) jusqu'à ce que le critère d'arrêt soit atteint.
La perturbation peut consister à redémarrer d'une solution prise aléatoirement dans l'espace de recherche ou à choisir une solution dans un voisinage lointain de l'optimum ou encore à choisir un voisin de même qualité que l'optimum...

1.2.1.3 Le recuit simulé (Simulated Annealing, SA)

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui est ajusté pendant la recherche [KGV83].

Elle s'inspire des méthodes de simulation de Metropolis (années 50) en mécanique statistique.
La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une procédure d'exploitation du voisinage qui permet de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre T est déterminé et décroît tout au long de l'algorithme pour tendre vers 0. De la valeur de ce paramètre dépend la probabilité p d'acceptation des solutions détériorantes (plus la température T décroit, plus cette probabilité diminue).
L'intérêt du recuit simulé est qu'il existe une preuve de la convergence asymptotique. Ainsi, lorsque certaines conditions sont vérifiées (schéma de décroissance particulier de T), on a la garantie d'obtenir la solution optimale. Malheureusement, le paramétrage recommandé par la théorie n'est pas réaliste et il faut beaucoup de temps pour arriver à paramétrer ces méthodes. Cette méthode peut, elle aussi, nécessiter un critère d'arrêt, dans le cas, où le paramétrage "optimal" n'a pas été trouvé.

1.2.1.4 La recherche tabou (Tabu Search, TS)

La recherche tabou a été introduite par Glover [GL98] comme une nouvelle stratégie pour échapper aux optima locaux en utilisant une notion de mémoire. L'exploitation du voisinage permet de se déplacer de la solution courante vers son meilleur voisin (celui-ci n'ayant pas forcément une qualité meilleure que la solution courante). Pour éviter de cycler entre un optimum local et son meilleur voisin, la méthode interdit de se déplacer vers une solution récemment visitée. Pour cela, une liste tabou contenant les attributs des dernières solutions visitées est tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée. Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante sans considérer les solutions appartenant à la liste tabou. Néanmoins, la taille de cette liste est un paramètre de la méthode qui s'avère difficile à régler.
De même, cette stratégie nécessite de définir un critère d'arrêt.

1.2.2 Méthodes à population de solutions

Contrairement aux recherches locales, les méthodes à base de population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité. Il existe plusieurs stratégies d'évolution de cette population amenant à des méthodes telles que les colonies de fourmis (Ants system) [DMC96], les algorithmes à essaim de particules (Particule swarm optimiser) [KE95], la recherche par dispersion (Scatter search) [Glo77], les algorithmes génétiques (Genetic algorithm) [Hol92]
Dans cette thèse, nous n'utiliserons que les algorithmes génétiques. Les algorithmes génétiques simulent le processus d'évolution d'une population. À partir d'une population de N solutions du problème représentant des individus, on applique des opérateurs simulant les interventions sur le génome telles que le croisement (crossover) ou la mutation pour obtenir une population de solutions de mieux en mieux adaptées au problème. Cette adaptation est évaluée grâce à la fonction objectif associée au problème d'optimisation.

Un algorithme génétique classique se déroule donc en plusieurs étapes successives :
(i) une population aléatoire est créée,
(ii) chaque individu de la population est évalué par la fonction objectif,
(iii) une stratégie de sélection choisi des individus pour former une population parent,
(iv) l'opérateur de croisement est appliqué avec une probabilité P_c sur les individus de la population parent pour obtenir une population enfant,
(v) l'opérateur de mutation est appliqué avec une probabilité P_m sur les individus de la population enfant,
(vi) la stratégie de remplacement amène à la nouvelle population,
(vii) retour à (ii) si le critère d'arrêt n'est pas atteint.
Il existe différentes variantes dans la conception des algorithmes génétiques. Citons les algorithmes mémétiques où l'opérateur de mutation est remplacé par une recherche locale.

Les métaheuristiques représentent une classe de méthodes algorithmiques permettant de résoudre ou d'approcher la meilleure solution d'un problème d'optimisation. Or pour résoudre un problème d'optimisation trois questions se posent :
– Comment doit-on le modéliser ? (représentation des solutions, fonction objectif, relation de voisinage, contraintes du problème...)
– Quelle métaheuristique est la plus favorable ? (recherche locale et exploration du voisinage, méthode à base de population de solution...)
– Quel est le paramétrage optimal ?
Pour répondre à ce genre de questions, il est possible de définir puis analyser une structure, appelée paysage, associée au problème d'optimisation à résoudre.

1.3 Paysage d'un problème d'optimisation

1.3.1 Qu'est-ce que le paysage ?

La notion de paysage (ou paysage de fitness) a été introduite pour la première fois dans la littérature par Wright [Wri32] dans ses travaux sur l'évolution des êtres vivants. Elle consiste à représenter des individus par rapport à leur *fitness* et donner une structure géométrique au problème, ceci pour essayer de comprendre les dérives de l'évolution. Cette notion fut ensuite transposée à de nombreux domaines comme celui de l'optimisation combinatoire.
Le paysage d'un problème d'optimisation combinatoire est défini par un triplet $(\Omega, \mathcal{V}, f)$ tel que :
– Ω est l'espace de recherche, l'ensemble des solutions réalisables.
– $\mathcal{V} : \Omega \to 2^{\Omega}$ est une relation de voisinage, une application qui associe à toute solution s de l'espace de recherche un ensemble $\mathcal{V}(s)$ de solutions réalisables, appelées *voisins*,
– $f : \Omega \to \mathbb{R}$ est une fonction objectif qui mesure la qualité des solutions réalisables.
Cette définition du paysage revient à donner une structure géométrique basée sur deux composantes à un problème d'optimisation : une relation de voisinage et une fonction objectif. Le paysage d'un problème d'optimisation n'est donc pas unique.
La définition du paysage est liée à la dynamique des métaheuristiques. En effet dans le cadre d'une métaheuristique, la fonction objectif permet de faire des choix entre les

solutions rencontrées et la relation de voisinage est un élément important d'une recherche locale. L'étude du paysage (ou des paysages) associé(s) à un problème d'optimisation peut aider à répondre aux questions posées précédemment telles que :

– Quelle est la meilleure modélisation (représentation, voisinage...) de mon problème ?
– Comment choisir une métaheuristique adaptée à mon problème ?
– Comment paramétrer au mieux la métaheuristique choisie ?

1.3.2 Relation de voisinage

Dans le contexte de l'optimisation combinatoire, la taille de l'espace de recherche augmente avec la taille de l'instance à résoudre. Ces problèmes d'optimisation deviennent difficiles quand les solutions sont nombreuses et deviennent non énumérables en un temps raisonnable. La relation de voisinage permet de connecter entre elles les solutions de l'espace de recherche. Or à cause de la taille de l'espace de recherche, il est impossible de décrire explicitement tous les voisins de toutes les solutions. La relation de voisinage est un mécanisme qui généralise l'affectation des voisins à chaque solution. Elle peut être définie rigoureusement [SS07] à partir de définitions mathématiques pour décrire un espace discret.

Une *opération* est une application symétrique $\delta : \Omega \longrightarrow \Omega$ qui à toute solution s de Ω associe une autre solution s' de Ω telle que $s \neq s'$. Une opération correspond à un mouvement de s à s' dans l'espace de recherche.

Un *opérateur* Δ est une collection d'opérations. Partant d'une solution s de l'espace de recherche, une solution est obtenue pour chaque opération de l'opérateur Δ. Cet ensemble de solutions est appelé le *voisinage* de la solution s pour l'opérateur Δ. Considérant l'opérateur Δ, il est possible de définir plus généralement le voisinage de toute solution de l'espace de recherche. Une *relation de voisinage* est une application $\mathcal{V} : \Omega \longrightarrow 2^{\Omega}$ qui associe à toute solution de l'espace de recherche l'ensemble des solutions obtenues par l'application d'un opérateur.

Soit une relation de voisinage $\mathcal{V}$ définie à partir d'un opérateur Δ, on a l'équivalence :

(i) s' est dans le voisinage de s, noté $\mathcal{V}(s)$,

(ii) il existe une opération δ de l'opérateur Δ tel que $\delta(s) = s'$.

La relation de voisinage dans un espace de recherche permet de définir formellement l'ensemble des voisins pour toutes les solutions de celui-ci. La taille du voisinage d'une solution est le nombre de voisins donc le nombre d'opérations de l'opérateur possibles à partir de cette solution. Dans le cadre de l'optimisation combinatoire, l'opérateur de voisinage correspond à une légère modification dans la représentation des solutions. La taille du voisinage correspond donc au nombre de modifications envisageables selon une règle (un opérateur) pour les solutions de l'espace de recherche. L'espace de recherche est donc vu différemment par une métaheuristique selon l'opérateur de voisinage considéré.

Un *chemin* entre deux solutions s et s' de l'espace de recherche est un sous-ensemble de solutions $\{s_1, ..., s_n\} \subset \Omega$ tel que $s_1 = s$ et $s_n = s'$ et $\forall i \in [\![1; n-1]\!], \exists \delta \in \Delta,\ \delta(s_i) = s_{i+1}$. Un chemin correspond à une séquence finie de mouvements d'une solution à une autre. $\phi(s, s')$ est l'ensemble des chemins possibles pour aller de s à s'. La *distance* entre deux solutions s et s' de l'espace de recherche est définie comme la taille du plus petit chemin

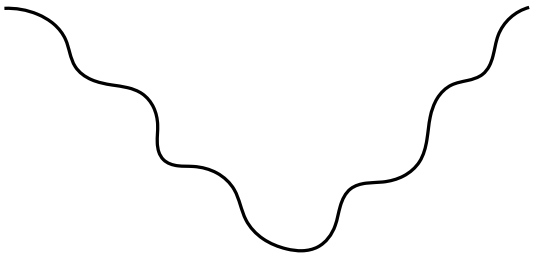


Figure 1.1 – Paysage lisse.

Figure 1.2 – Paysage rugueux.

possible entre s et s' :

$$dist(s, s') = \min_{\phi \in \phi(s,s')} |\phi|$$

Le voisinage d'ordre n, noté $\mathcal{V}_n$ d'une solution s correspond à l'ensemble des solutions s' de E qui sont à exactement une distance de n de s. $\forall s' \in \mathcal{V}_n(s), dist(s, s') = n$, n mouvements sont donc nécessaires pour aller de s à s'. Cette définition de distance dans l'espace de recherche respecte les trois axiomes des distances suivants :

(i) définie positive : $\forall s' \in \mathcal{V}_n(s), dist(s, s') = 0 \Leftrightarrow s = s'$.

(ii) symétrique : l'opération $\delta \in \Delta$ est une application symétrique
i.e. $\forall s_1, s_2 \in \Omega, dist(s_1, s_2) = d(s_2, s_1)$.

(iii) inégalité triangulaire : soient $s_1, s_2, s_3 \in \Omega$, $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$.
Si $p = d(s_1, s_3)$ mouvements sont nécessaires pour transformer s_1 en s_3 et $q = d(s_3, s_2)$ mouvements sont nécessaires pour transformer s_3 en s_2 alors il est possible de transformer s_1 en s_2 en exactement $p+q$ mouvements. Il existe un chemin de longueur $p+q$ pour aller de s_1 à s_2, la distance entre s_1 et s_2 est donc bornée par $p + q$ qui correspondent au nombre maximal d'opérations $\delta \in \Delta$ *i.e.* à la borne maximale.

Les voisins s' d'une solution s sont les solutions exactement à une distance égale à 1 de s *i.e.* $dist(s, s') = 1$.

1.3.3 Structure de paysage

Le paysage est un outil qui permet de se représenter différemment le problème. La relation de voisinage connecte les solutions les unes aux autres et la fonction objectif leur affecte une qualité. Cette définition du paysage donne une structure à l'espace de recherche qu'on peut essayer de visualiser. Si les solutions sont disposées sur un plan selon la relation de voisinage et que chacune a une altitude correspondant à sa qualité, le paysage peut être vu comme une représentation topographique où le relief est donné par la différence des valeurs de fitness entre chaque solution voisine. On parle d'*altitude* pour les zones du paysage correspondant aux solutions ayant une grande fitness et de *profondeur* pour les zones correspondant aux solutions ayant une petite fitness.

Cette représentation permet d'appréhender le degré de difficulté du problème. En effet, il est possible d'imaginer que l'optimum global est plus facile à atteindre sur un paysage relativement lisse que sur un paysage rugueux où les optima locaux sont nombreux. Cette représentation visuelle du problème peut aider à appréhender le comportement d'une méthode de résolution et en particulier des métaheuristiques sur une instance d'un problème

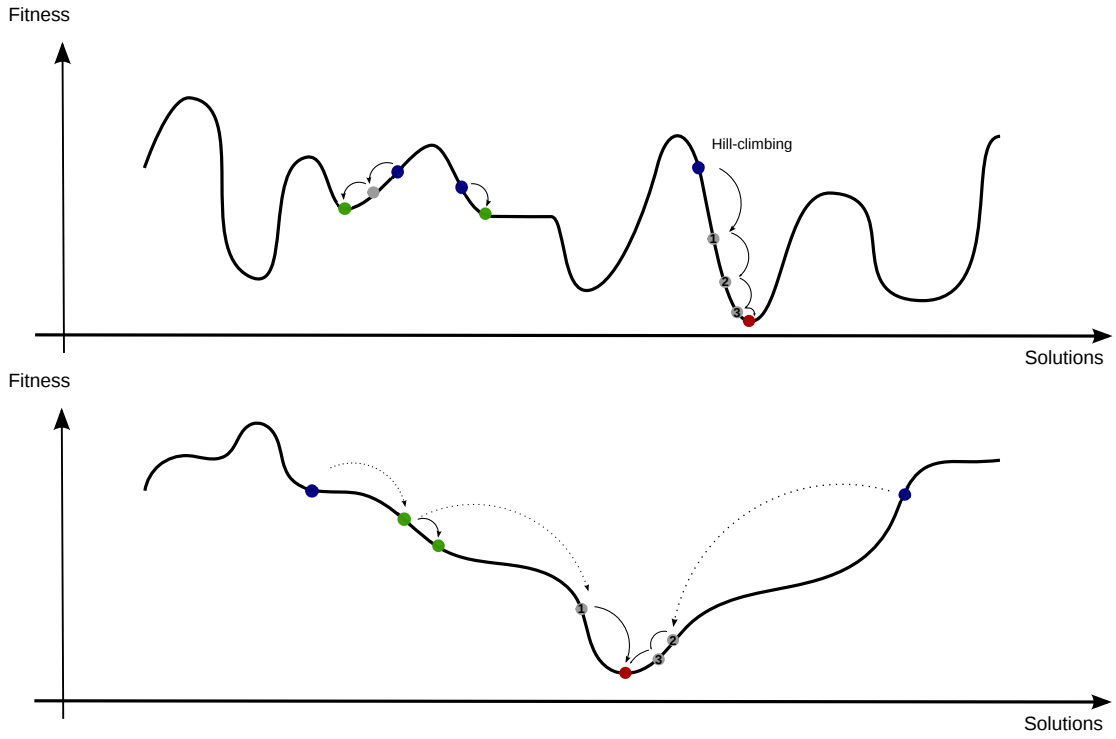


Figure 1.3 – Deux paysages adaptatifs pour un problème d'optimisation. Pour le même espace de recherche, la même fonction d'évaluation mais deux relations de voisinage différentes, les graphiques présentent deux situations pour une recherche locale (ici, le Hill Climbing) partant de trois solutions (en bleu). Dans le premier, alors qu'une solution première solution (à droite) trouve bien l'optimum global (en rouge), les deux autres sont piégées par des optima locaux (en vert). Ces solutions représentées dans le deuxième graphique (avec une relation de voisinage différente) conduisent alors à l'optimum global du problème d'optimisation. (Les flèches pleines correspondent à une application de l'opérateur contrairement à celles qui sont en pointillées)

d'optimisation. Par exemple, si le paysage induit par l'instance d'un problème se révèle être assez lisse et vallonné (Fig. 1.1), une métaheuristique avec une grande capacité d'exploitation sera alors très efficace pour se déplacer de n'importe qu'elle solution vers une solution de très bonne qualité voire vers même l'optimum global. Par contre, si le paysage est très accidenté (Fig. 1.2), une métaheuristique avec une grande capacité d'exploration sera alors plus intéressante à utiliser car elle pourra se sortir plus facilement des nombreux optima locaux qui ponctueront son parcours. Bien évidemment, les problèmes ont généralement une structure de paysage beaucoup plus complexe. Mais localement, en regardant au niveau proche de la solution, on peut s'aider de certaines caractéristiques locales du paysage. On peut alors ajuster les taux d'exploration et d'exploitation d'une métaheuristique suivant la zone du paysage dans laquelle on se trouve. L'étude de la géométrie du paysage selon différentes relations de voisinage ou fonctions d'évaluation peut permettre de choisir une structure facile à parcourir pour une ou plusieurs métaheuristiques puis d'aider dans le paramétrage de celles-ci.
Par exemple, la Figure 1.3 présente pour un même problème d'optimisation deux paysages différenciés par leur relation de voisinage. Le graphique du haut montre un paysage plutôt rugueux faisant apparaître plusieurs optima locaux. Partant d'une solution aléatoire, une méthode de descente classique peut facilement se trouver piéger dans un optimum local. Quant à la relation de voisinage du graphique du bas, elle permet de lisser le paysage : partant d'une solution aléatoire, l'optimum global est facile à atteindre. Remarquons qu'un optimum local pour une relation de voisinage (solution en violet) ne l'est pas forcément pour une autre relation. Notons que, comme dans le milieu naturel, il peut exister des zones de l'espace où les solutions ont la même altitude. Ce qui est représenté par un *plateau*. Cette propriété de *neutralité* sera étudiée en détail dans le chapitre 3.

Pour continuer avec un vocabulaire imagé, définissons les bassins d'attraction. Un bassin d'attraction b_j pour un optimum local s_j^* correspond, pour le paysage $(\Omega, \mathcal{V}, f)$, au sous-ensemble $\{s_1, ..., s_k\}$ de Ω tel que pour tout s_i $(1 \leq i \leq k)$ une méthode de descente classique peut atteindre l'optimum s_j^*. On note

$$b_j = \{s \in \Omega \mid HillClimbing(s) = s_j^*\}$$

La recherche locale utilise l'opérateur défini par la relation de voisinage $\mathcal{V}$. La taille d'un bassin d'attraction correspond au nombre de solutions qui, par des applications successives de l'opérateur de voisinage, peuvent atteindre le même optimum local. Cette taille est donc variable suivant la relation de voisinage utilisée car les liaisons entre les solutions sont différentes. Les recherches locales étant généralement stochastiques, une solution peut appartenir à plusieurs bassins d'attraction. En effet, si une solution a plusieurs voisins améliorants ayant la même valeur de fitness, considérant que la recherche locale peut choisir un voisin plutôt qu'un autre, il est possible d'atteindre plusieurs optima locaux. La figure 1.4 illustre ces situations et permet de se rendre compte que diviser l'espace de recherche s'avère difficile si l'on tient compte de la stochasticité des méthodes de résolution. Les frontières des bassins d'attraction bougent dynamiquement au cours du processus d'optimisation.

Le vocabulaire et les définitions liés au paysage d'un problème d'optimisation étant don-

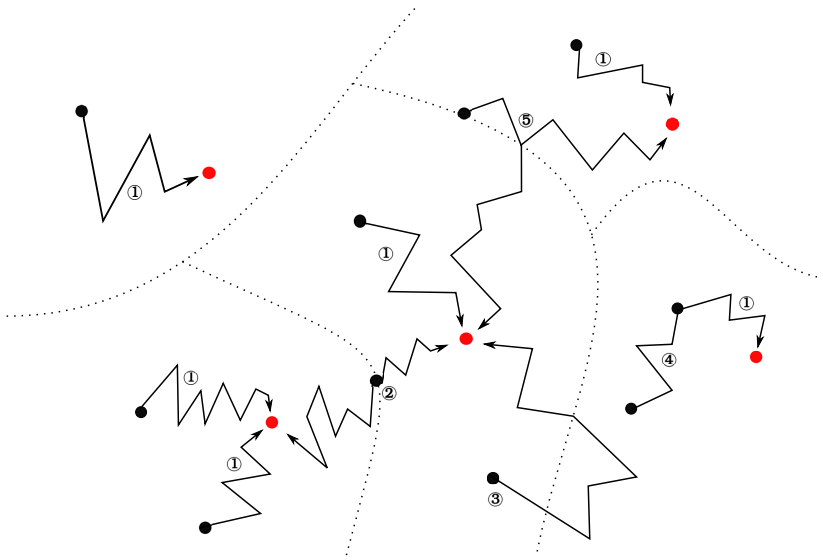


Figure 1.4 – Une partie de l'espace de recherche est représentée. Les lignes en pointillés délimitent 5 bassins d'attraction. Les solutions de départ sont représentées par des points noirs alors que les optima locaux sont en rouge. Chaque optimum local possède un bassin d'attraction différent. Le Hill Climbing étant une recherche locale stochastique, plusieurs trajectoires (les flèches) sont envisageables et représentées par les numéros 1 à 5.(1) représente une trajectoire classique. (2) montre une solution à la limite de deux bassins d'attraction ; la stochasticité du Hill Climbing donnent deux trajectoires différentes qui amènent à deux optima différents. (3) commence sa trajectoire dans un bassin, passe dans un autre mais revient dans son bassin d'origine. (4) rejoint une trajectoire classique. (5) démarre dans un premier bassin d'attraction puis se retrouve sur une solution à la frontière, deux trajectoires sont alors envisagées.

nés, la section suivante s'attache à présenter les outils et les indicateurs de la littérature permettant d'étudier une telle structure.

1.4 Étude du paysage

Le paysage d'un problème d'optimisation combinatoire est un triplet composé de l'espace de recherche des solutions réalisables, d'une relation de voisinage entre ces solutions et d'une fonction objectif. Ceci représente une structure complexe qu'on cherche à étudier pour nous aider dans la résolution du problème d'optimisation. Nous présentons un état de l'art des outils d'analyse de la littérature, puis des indicateurs du paysage basés sur ces outils.

1.4.1 Outils

L'espace de recherche d'un problème d'optimisation combinatoire NP-dur est très vaste et ses solutions peuvent être non énumérables en un temps acceptable. Pour pouvoir étudier le paysage, nous avons identifié plusieurs moyens utilisés dans la littérature, que nous

explicitons ci-après. Le paysage peut être échantillonné soit à l'aide de populations de solutions [Mer04, Bac99, Boe95], soit être caractérisé par des informations collectées tout au long d'une marche [Dra10, Wei90, Mer04, Pel10]. La distance entre les solutions permet de comprendre la configuration de l'espace de recherche et donc du paysage.

1.4.1.1 Échantillonnage à l'aide de populations de solutions

Dans la littérature, des populations de solutions ont été utilisées pour caractériser le paysage d'un problème [Mer00, Bac99, Boe95]. Les auteurs tentent alors de trouver des informations sur le paysage à partir de populations bien choisies. Deux types de populations sont utilisées pour caractériser le paysage : les populations de solutions aléatoires et les populations d'optima locaux. Une population de solutions aléatoires est un ensemble de solutions choisies aléatoirement sans remise dans l'espace de recherche. Cette population est utilisée pour comprendre la structure de l'espace de recherche : Comment les solutions sont-elles connectées ? Sont-elles proches ?... Une population d'optima locaux est un ensemble d'optima locaux trouvés en utilisant par exemple une méthode de descente à partir d'une solution quelconque de l'espace de recherche. La méthode de descente (*cf.* section 1.2.1.1) utilise dans ce cas la même relation de voisinage que celle qui définit le paysage que l'on souhaite étudier. Cette population permet d'avoir des informations sur les optima locaux : sont-ils proches ? très dispersés ? et donc des informations sur les bassins d'attraction.

1.4.1.2 Marches sur le paysage

Une marche est une suite ordonnée de solutions voisines de l'espace de recherche qui correspond à un déplacement à travers l'espace de recherche. Il existe plusieurs types de marches à travers l'espace de recherche : les marches *aléatoire*, *améliorante* ou *neutre*. La marche aléatoire consiste à se déplacer d'une solution vers une solution voisine choisie au hasard. Cette marche peut être très longue, voire infinie, et ne pas tenir compte des valeurs de fitness des solutions pour se déplacer. Par contre, les valeurs de fitness sont utilisées ensuite pour, par exemple, connaître la continuité des altitudes entre les voisins. La marche améliorante correspond aux méthodes de descente qui cherchent dans le voisinage d'une solution un voisin qui a une qualité meilleure que lui-même. La marche neutre, quant à elle, se déplace de voisin en voisin ayant la même qualité. Elle permet de caractériser des zones plates du paysage.
Ces différentes marches permettent d'étudier les relations entre les solutions voisines et étudier la continuité ou non entre leurs caractéristiques.

1.4.1.3 Distance entre les solutions

Dans ce contexte de caractérisation de l'espace de recherche interconnecté par la relation de voisinage, il peut être nécessaire de connaître sa dimension et comment les solutions y sont placées. Ainsi, on utilise la notion de distance présentée dans le paragraphe précédent entre deux solutions de l'espace de recherche. Elle correspond au nombre minimal d'applications de l'opérateur nécessaires pour transformer une solution en une autre. La distance étant

basée sur l'opérateur de voisinage et la définition du paysage aussi, elle est très utile pour caractériser un espace de recherche en fonction de la relation de voisinage considérée.
Considérant le paysage $(\Omega, \mathcal{V}, f)$ comme un graphe dont les nœuds sont les solutions de Ω, les arêtes entre les solutions traduisent les relations de voisinage $\mathcal{V}$ et alors, la distance entre deux solutions revient à compter le nombre minimal d'arêtes entre deux solutions. Néanmoins, il est souvent très difficile de pouvoir calculer cette distance car à moins d'évaluer toutes les possibilités de chemins entre deux solutions, il faut connaître un bon algorithme. Dans le cas des solutions représentées par des permutations, Schiavinotto *et al.* [SS07] donnent les définitions des trois opérateurs de voisinage les plus utilisés et ils décrivent pour chacun le pseudo-code de l'algorithme qui permet de calculer la distance entre deux solutions. On se rend compte que, même si l'algorithme existe, il n'est pas toujours trivial et sa complexité peut être élevée. Le temps de calcul de la distance entre deux solutions pouvant donc être très long, on se demande s'il est vraiment nécessaire de connaître la distance exacte entre les solutions pour pouvoir faire une analyse de paysage.
On commence à trouver des travaux d'analyse de paysage de problèmes d'optimisation surtout à partir des années 1990. Beaucoup de travaux [Boe95, JF95, FRP97, Mer00, Ree97, Dra10] font une étude du paysage du TSP. Pour ce problème, dont les solutions peuvent être représentées par une permutation, il est reconnu que l'opérateur de voisinage fonctionnant le mieux est le 2-opt. Seulement, il n'existe pas d'algorithme pour calculer exactement la distance entre deux solutions en utilisant uniquement cet opérateur. Or, on s'aperçoit dans les études de paysage que la distance joue un rôle important dans la compréhension du problème et de ses structures de paysage. Les auteurs utilisent donc des distances approximatives. Les résultats obtenus semblent satisfaisants et assez corrélés à ce qui était attendu *a priori*. Mais Reeves [Ree97] conclut tout de même en conseillant de regarder l'influence de l'utilisation d'une distance approximative sur les résultats obtenus pour le paysage. Quant à Schiavinotto *et al.* [SS07], ils montrent dans leur étude sur les distances pour les permutations que les approximations classiques des distances exactes sont souvent mauvaises et préconisent d'utiliser les distances exactes même si elles sont coûteuses en temps de calcul. Si la distance entre deux solutions ne peut être calculée exactement alors il faut essayer de tenir compte du biais introduit par la distance de substitution. Ainsi, Draskoczy [Dra10] compare plusieurs voisinages en précisant que deux sur les cinq opérateurs de voisinage comparés ne possèdent pas d'algorithme connu de calcul de distance.
Dans cette thèse, on s'efforcera donc à utiliser des opérateurs dont on connaît un algorithme de calcul de la distance exacte ou pour lesquels on en propose un.

1.4.2 Indicateurs

Dans la littérature, on trouve plusieurs études de paysages adaptatifs de problèmes d'optimisation combinatoire : paysages-NK, One-Max, Max-SAT, TSP, Graph Partitioning, QAP, BQP. Ces études introduisent des outils statistiques plus ou moins généraux pour décrire le paysage d'un problème d'optimisation. On propose ici de découper cette analyse du paysage en trois axes d'étude complémentaires :
– l'espace de recherche interconnecté *i.e.* étudier $(\Omega, \mathcal{V})$,
– l'espace des valeurs de fitness *i.e.* étudier (Ω, f),

– le paysage complet *i.e.* étudier $(\Omega, \mathcal{V}, f)$.

Cette approche nous semble pertinente pour distinguer les caractéristiques du problème induites principalement par la relation de voisinage ou par la fonction d'évaluation. En effet, regarder l'espace de recherche uniquement du point de vue des connexions entre les solutions apporte des informations sur la dimension et la répartition spatiales des solutions du problème. De même, regarder la qualité des solutions indépendamment de leur situation dans l'espace de recherche permet de regarder la distribution selon l'altitude. Évidemment, il est essentiel d'étudier aussi le paysage dans sa globalité pour comprendre les interactions et l'influence mutuelle de la relation de voisinage et de la fonction d'évaluation. Dans la partie précédente, nous avons parlé des populations de solutions qui permettent de caractériser le paysage tout entier sans avoir besoin de le décrire totalement. Les indicateurs que nous présentons dans la suite utilisent une population aléatoire, notée P_{init} et une population d'optima P^*. Clairement, cette dernière est directement liée au choix de la relation de voisinage utilisée par la recherche locale et/ou au choix de la fonction objectif pour évaluer les solutions de l'espace de recherche.

1.4.2.1 L'espace de recherche

Comme la taille de l'espace de recherche croît exponentiellement avec la taille du problème à résoudre, il n'est pas envisageable d'énumérer exhaustivement toutes les solutions pour le décrire. Ainsi pouvoir estimer la répartition des solutions, voire des optima, dans cet espace de recherche permet d'indiquer si les solutions sont très rapprochées ou non (la relation de voisinage choisie permettant de contracter ou inversement de dilater l'espace de recherche), si les optima locaux sont proches ou pas, s'ils se trouvent tous dans une partie bien identifiée (la relation de voisinage auraient tendance à connecter et rapprocher les optima locaux), si les solutions voisines ont des représentations très semblables... Dans la littérature, les auteurs proposent différents indicateurs statistiques que nous présentons ici.

– Distance dans une population : L'espace de recherche est interconnecté différemment selon la relation de voisinage qui définit le paysage. Pour comprendre les interconnexions entre les solutions, on peut calculer les distances entre les solutions d'une population qui caractérise l'espace de recherche. De même, il est possible de calculer la distance par rapport à l'optimum global ou le meilleur optimum local connu [Mer00, Bac99]. La moyenne et les quartiles des distances calculées dans la population permettent d'échantillonner et d'estimer la distribution des solutions de l'espace de recherche.

– Diamètre : Il correspond à la distance maximale trouvée entre deux solutions de la population P_{init}. Le diamètre est utilisé par différents auteurs ([Mer00, Bac99]) car il permet de connaître la dimension spatiale de l'espace de recherche induite par la relation de voisinage. Le diamètre d'une population P est donné par :

$$diam(P) = \max_{s,s' \in P} dist(s, s')$$

De façon théorique, le diamètre de l'espace de recherche est égal à la longueur du chemin le plus long possible existant entre deux solutions. On peut vérifier si la distance

maximale théorique est atteinte souvent ou non dans une population et se représenter l'occupation d'une population dans un espace de recherche.
Bachelet [Bac99] propose trois indicateurs basés sur cette notion de diamètre de population.

– Diamètre moyen : Il donne la distance moyenne entre les solutions deux à deux dans une population. Soit, pour une population P :

$$dmm(P) = \frac{\sum_{s \in P} \sum_{s' \in P} dist_{\mathcal{V}}(s, s')}{|P|^2}$$

– Diamètre moyen relatif : Il permet de caractériser la concentration d'une population P dans l'espace de recherche Ω. Soit,

$$Dmm(P) = 100 \times \frac{dmm(P)}{diam(\Omega)}$$

Un diamètre moyen relatif faible indique que les points de la population P sont regroupés dans une région de l'espace de recherche. Cet indicateur semble intéressant car il prend ces valeurs dans l'intervalle $[0; 100]$, indépendamment de la taille du problème traité ce qui permet de rendre les conclusions génériques.

– Variation de diamètre moyen relatif : Elle rend compte de la différence d'occupation de l'espace de recherche pour deux populations. Par exemple, pour les populations P_{init} et P^*, on calcule cette variation par :

$$\Delta Dmm = Dmm(P_{init}) - Dmm(P^*)$$

La variation de diamètre moyen est donc comprise entre -100 et 100. Elle indique si les solutions optimales trouvées occupe une zone inférieure, égale, voire supérieure à la zone occupée par des solutions générées aléatoirement. Son utilisation peut être intéressante pour comparer deux relations de voisinage et leurs impacts sur la dimension de l'espace de recherche à explorer.

– Entropie : Elle mesure la dissimilarité entre les solutions d'une population suivant une loi de probabilité choisie. Cette loi de probabilité peut tenir compte de critères représentatifs (sur la représentation des solutions), spatiaux (les distances entre les solutions)... L'entropie, pour une variable aléatoire discrète X dans l'espace de probabilité $(\Omega, \mathcal{P}(\Omega), \mathbb{P})$, est donnée par :

$$E(X) = -\sum_{x \in \Omega} \mathbb{P}(X = x) \ln \mathbb{P}(X = x)$$

avec la convention $\ln(0) = 0$. L'entropie d'une population est toujours positive. Une entropie égale à 1 signifie que pour le critère considéré la population étudiée est très diversifiée. Une entropie égale à 0 signifie au contraire que les solutions sont toutes identiques par rapport au critère considéré. Par exemple, l'entropie permet de vérifier si les solutions d'une population sont similaires ou différentes en terme de représentation.

De plus, s'il est possible de regarder toutes les solutions d'un problème en un temps raisonnable alors on peut calculer le nombre d'optima locaux voire même étudier le réseau constitué de ces mêmes optima [VOT08]. Ceci met en évidence le rôle d'une relation de voisinage sur le nombre total d'optima locaux et donc sur sa faculté à piéger facilement une métaheuristique.

1.4.2.2 L'espace objectif

Visuellement, la valeur de fitness correspond à l'altitude de la solution dans le paysage. La distribution des valeurs de fitness d'une population peut donner des indications sur le relief (plat, rugueux, vallonné) que peut induire la fonction d'évaluation. De plus, les valeurs de fitness des optima locaux trouvés avec des relations de voisinage différentes aident à choisir celle qui permet d'obtenir les solutions de meilleures qualités ou une population homogène d'optima de bonne qualité.
Dans la littérature, les auteurs proposent en majorité de regarder les statistiques classiques sur une population : la moyenne, l'écart-type, le minimum, le maximum et les quartiles. En effet, elles informent sur la distribution des valeurs de fitness des solutions d'une population générée aléatoirement, d'une population au cours de la résolution du problème ou d'une population d'optima locaux. Une étude classique informe sur la qualité d'une population de solutions.
Bachelet [Bac99] quant à lui, propose des indicateurs statistiques pour étudier les valeurs de fitness des solutions d'une population.

– Amplitude : Elle donne l'écart relatif entre la valeur de fitness de la meilleure solution et celle de la pire solution d'une population. Pour une population P, on a :

$$Amp(P) = 100 \times \frac{|P|(\max_{s\in P} f(s) - \min_{s\in P} f(s))}{\sum_{s\in P} f(s)}$$

Cet indicateur, à valeur dans $[0; +\infty[$ est assez compliqué à comprendre car il existe plusieurs possibilités pour lesquelles il peut prendre la même valeur alors que les distributions de valeurs de fitnesss sont très différentes. De plus, la différence min : max a des conséquences plus importantes sur la valeur de l'amplitude si les valeurs de fitness du problème sont très grandes.

– Variation d'amplitude : Elle mesure la variation d'amplitude entre une population aléatoire P_{init} et la population P^* associée.

$$\Delta_{Amp} = \frac{Amp(P_{init}) - Amp(P^*)}{Amp(P_{init})}$$

Cette variation est d'autant plus compliquée à expliquer puisqu'elle est calculée à partir d'un indicateur avec lequel il est difficile de conclure. Mais selon que l'amplitude de la population initiale ou l'amplitude de la population des optima locaux soient égales ou différentes, la variation entre les deux peut être positive ($\in [0; 1]$) ou négative ($\in]-\infty; 0]$) sans que l'on ne puisse vraiment dire ce qui les différencie en connaissant uniquement

la valeur de cet indicateur.

– Gap : Il donne la moyenne des écarts relatifs des valeurs de fitness des optima locaux de la population P^* par rapport à la meilleure solution connue s^*.

$$Gap(P^*) = 100 \times \frac{\sum_{s \in P^*}(f(s) - f(s^*))}{|P^*|.f(s^*)}$$

Cet indicateur est d'autant plus grand que les valeurs de fitness de la population considérée sont éloignées de la valeur de fitness de s^*.

Dans une analyse de paysage d'instances du QAP, Bachelet montre que la variation d'amplitude et le gap sont corrélés ce qui, selon lui, permet d'estimer la qualité de l'optimum global, même pour les instances où il n'est pas connu [Bac99]. Ces indicateurs sont en réalité assez difficiles à expliquer et ne suffisent pas pour conclure quant à la qualité ou non d'une population d'optima locaux. Des expérimentations sur la pertinence de ces indicateurs ont montré que les statistiques classiques sont plus satisfaisantes pour déduire des propriétés intéressantes du paysage et donc du problème à optimiser.

1.4.2.3 Le paysage complet

Dans la définition du paysage, la relation de voisinage et la fonction d'évaluation sont étroitement liées. En effet, la relation de voisinage connecte les solutions les unes aux autres alors que la fonction d'évaluation leur donne une altitude. Ainsi, les solutions voisines peuvent avoir des valeurs de fitness proches ou très différentes, ce qui conduit dans le premier cas à un paysage plutôt lisse contrairement au second cas où le paysage serait très rugueux. Des indicateurs statistiques permettent de donner des informations sur le type du paysage défini par un opérateur :

– Coefficient de corrélation fitness-distance (*fitness distance correlation* (FDC) coefficient) : Il détermine s'il existe une relation entre la valeur de fitness et la distance à l'optimum le plus proche. Le FDC [JF95] est défini par :

$$FDC = \frac{Cov(f, d_{opt})}{\sigma(f)\sigma(d_{opt})} = \frac{\langle f d_{opt} \rangle - \langle f \rangle \langle d_{opt} \rangle}{\sqrt{(\langle f^2 \rangle - \langle f \rangle)^2 (\langle d_{opt}^2 \rangle - \langle d_{opt} \rangle)^2}}$$

où $\langle\rangle$ est le produit scalaire et $d_{opt}(s)$ est la distance entre s et l'optimum local le plus proche. Pour une population de solutions $P = \{s_1, s_2, ..., s_n\}$, avec comme notation $f_i = f(s_i)$ la valeur de fitness et, $d_i = d_{opt}(s_i)$ la plus petite distance jusqu'à un optimum global, le FDC est estimé par :

$$FDC \approx \frac{1}{\sigma(f)\sigma(d)} \frac{1}{m} \sum_{i=1}^{m} (f(s_i) - \bar{f})(d_{opt}(s_i) - \bar{d})$$

où $\bar{f}$ est la moyenne des $f(s_i)$ et $\bar{d}$ est la moyenne des $d_{opt}(s_i)$. Quand on ne connaît pas le (voire les) optimum global, alors on prend la meilleure solution connue. Le FDC prend ses valeurs dans $[-1; 1]$. Dans le cas d'une minimisation (maximisation) un FDC = 1

(FDC = -1) montre une très forte relation entre la valeur de fitness d'une solution et sa distance à l'optimum, le plus proche. De même, un FDC = -1 (FDC = 1) montre que plus la solution est éloignée de l'optimum global, plus la valeur de fitness est grande (petite).

– Autocorrélation : Elle mesure la rugosité d'un paysage. On dit alors qu'un paysage est rugueux si les solutions voisines du paysage ont des valeurs de fitness très différentes contrairement à un paysage plutôt lisse où la corrélation est élevée entre les valeurs de fitness des solutions voisines. Weinberger [Wei90] a proposé une fonction d'autocorrélation qui calcule la corrélation entre les valeurs de fitness de solutions distantes de d :

$$\rho(d) = \frac{\langle f(x)f(y)\rangle_{d(x,y)=d} - \langle f\rangle^2}{\langle f^2\rangle - \langle f\rangle^2}$$

$$= 1 - \frac{\langle (f(x) - f(y))^2\rangle_{d(x,y)=d}}{2(\langle f^2\rangle - \langle f\rangle^2)}$$

En générant aléatoirement un très grand nombre N de couples de solutions (x, y) telles que $dist(x, y) = d$, on peut estimer l'autocorrélation par :

$$\rho(d) = \frac{1}{\sigma^2(f)N} \sum_{\substack{(x,y)\in\Omega\times\Omega \\ dist(x,y)=d}} (f(x) - \bar{f})(f(y) - \bar{f})$$

Cette formule présente un inconvénient majeur car elle nécessite de savoir : soit calculer exactement la distance entre deux solutions, soit générer de manière fiable une solution à exactement une distance égale à d de toute solution de l'espace de recherche. Il n'est pas toujours évident de savoir faire au moins l'une de ces deux possibilités. Weinberger a donc proposé une autre fonction similaire : la fonction de corrélation d'une marche aléatoire que l'on assimile généralement à la fonction d'autocorrélation définie précédemment car elle est plus simple à calculer. Pour une série temporelle $\{f(x_t)\}$, on définit cette fonction comme :

$$r(s) = \frac{\langle f(x_t)f(x_{t+s})\rangle - \langle f\rangle^2}{\langle f^2\rangle - \langle f\rangle^2}$$

la corrélation des valeurs de fitness entre toutes solutions à t et $t + s$. Si le paysage est statistiquement isotrope *i.e.* la série temporelle $\{f(x_t)\}$ est un processus aléatoire stationnaire, il suffit d'une unique marche aléatoire pour calculer $r(s)$:

$$r(s) = \frac{1}{\sigma^2(f)(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \bar{f})(f(x_{t+s}) - \bar{f})$$

avec m la longueur de la marche aléatoire. Ces deux fonctions d'autocorrélation du paysage sont intéressantes car elles décrivent la rugosité du paysage du point de vue du voisinage plus ou moins éloigné. Mais, afin d'utiliser ces indicateurs pour paramétrer les métaheuristiques avant et pendant la résolution, il est préférable d'avoir une valeur unique à traiter.

– Longueur de corrélation : Elle traduit la rugosité vis-à-vis des voisins dans le paysage. Elle est calculée à partir de l'autocorrélation. Pour cela, il faut que la marche aléatoire soit une série temporelle isotrope, gaussienne ou markovienne [Wei90]. Stadler [Sta96] propose un premier indicateur qui est la valeur en 1 de la fonction d'autocorrélation :

$$l = -\frac{1}{\ln(|\rho(1)|)} = -\frac{1}{\ln(|r(1)|)}$$

pour $r(1), \rho(1) \neq 0$. Plus la longueur de corrélation l est petite, plus le paysage est considéré comme rugueux. Une mesure similaire pour la longueur de corrélation a été proposée par [AZ98] qui définit le coefficient d'autocorrélation l' :

$$l' = -\frac{1}{1-\rho(1)} \approx l$$

Ces deux indicateurs sont plus pratiques à utiliser mais pas forcément évident à comparer quand la taille du problème augmente. Il est pratique de normaliser ces deux indicateurs par le diamètre de l'espace de recherche. Soit,

$$\xi = \frac{l}{diam(\Omega)}$$

Ainsi, si ξ est proche de 1, on considère que le paysage est fortement corrélé au contraire si ξ est proche de 0, il n'existe aucune corrélation entre les valeurs de fitness des solutions voisines du paysage.

– Longueur de marche moyenne relative : Elle est la normalisation par rapport au diamètre de l'espace de recherche de la moyenne des longueurs de marche. Soit :

$$Lmm(P) = 100 \times \frac{lmm(P)}{diam(\Omega)}$$

où $lmm(P)$ est la moyenne des longueurs de marche pour passer de la population P_{init} à P^*. Comme pour l'indicateur précédent, il est plus facile de comparer des valeurs normalisées et d'avoir des conclusions automatiques quant à la facilité de se diriger ou non vers un optimum local. Si Lmm est proche de zéro, alors le paysage semble assez plat et ondulé. Par contre, si Lmm est proche de 1, alors il est possible qu'il y ait de grandes vallées à explorer. Notons que Lmm peut aussi être supérieure à 1.

– Taux d'évasion (*escape rate* en anglais) : Il représente la probabilité de sortir du bassin d'attraction d'un optimum local par rapport à une distance donnée. Pour une distance d,

$$\tau_d = \frac{1}{|P^*|} \sum_{s_i^* \in P^*} \mathbb{P}(HillClimbing(s) \neq s_i^* \mid d(s, s_i^*) = d)$$

Cette mesure a été initialement proposée par Merz [Mer04] pour évaluer la taille des bassins d'attraction. Plus le taux est élevé, plus il semble facile de s'échapper d'un optimum local. Néanmoins, il est aussi d'autant plus facile de se retrouver piégé dans plusieurs optima locaux car le paysage paraît alors plus rugueux. Pelikan [Pel10] utilise

le taux d'évasion dans un étude de paysage-NK et conclut sur la nécessité de continuer à étudier cet indicateur pour comprendre son rôle sur la difficulté d'un problème.

– Point de qualité (PQ) [Dra10] : Il mesure la capacité d'une solution à conduire aux meilleures solutions de l'espace de recherche. Le point de qualité permet d'analyser et de visualiser l'espace de recherche.

$$PQ(s) = \frac{|\{s' \mid (d_{opt}(s') \leq d_{opt}(s) \wedge f(s') \leq f(s)) \vee (d_{opt}(s') \geq d_{opt}(s) \wedge f(s') \geq f(s))\}|}{|\mathcal{V}(s)|}$$

dans le cas d'un problème de minimisation. Dans ses travaux, Draskoczy [Dra10] montre que plus le PQ est élevé (proche de 1) et plus le paysage est adapté pour résoudre le problème. Les optima locaux de bonnes qualités sont proches de l'optimum global alors que les plus mauvais sont les plus éloignés de la population. De plus, il assure qu'un PQ proche de 1 apporte la même information que le FDC, de part leur définition. Il compare ses résultats expérimentaux, en représentant en fonction du PQ sur un premier graphique, les valeurs de fitness des optima locaux de P^* et, leur distance à l'optimum global, sur un deuxième.

1.4.3 Synthèse

Cette section propose un état de l'art des indicateurs existants pour décrire et étudier la structure de paysage d'un problème d'optimisation, tel qu'il a été défini en début de section. L'étude a été divisée en trois axes qui visent à distinguer l'influence de la relation de voisinage et/ou de la fonction d'évaluation sur le paysage induit du problème d'optimisation traité. Des indicateurs de la littérature ont ainsi été présentés et expliqués.

Le tableau 1.1 présente une vue d'ensemble des indicateurs de la littérature présentés dans cette section pour analyser le paysage d'un problème d'optimisation combinatoire. Ce tableau s'organise en six colonnes. La première permet de séparer les indicateurs selon l'un des axes d'études énoncés : l'espace de recherche interconnecté, l'espace des valeurs de fitness ou le paysage complet. Les deuxième, troisième et quatrième colonnes donnent respectivement le nom, la référence bibliographique et la formule de chacun des indicateurs présentés dans cette section. Pour chacun d'eux, le domaine de valeurs est précisé dans la cinquième colonne. Il a été montré que la distance entre les solutions n'étaient pas une partie triviale du travail d'analyse du paysage. En effet, il est parfois difficile voire impossible de calculer la distance entre deux solutions de l'espace de recherche suivant certains opérateurs de voisinage. La sixième colonne indique donc par une croix "x" si le calcul de l'indicateur traité nécessite ou non de connaître un algorithme pour calculer la distance exacte entre les solutions. (x) indique que l'indicateur peut, selon sa définition, avoir besoin ou non de l'algorithme de calcul de distance.

Vues d'analyse	Indicateurs	Référence	Formule	Domaine de valeurs	Distance
Espace de recherche	Diamètre	-	$diam(P) = \max_{s,s' \in P} dist_{\mathcal{V}}(s, s')$	$[\![0; card\ P]\!]$	x
	Diamètre moyen	[Bac99]	$dmm(P) = \frac{\sum_{s \in P} \sum_{s' \in P} dist_{\mathcal{V}}(s, s')}{\lvert P \rvert^2}$	$[0; diam(P)]$	x
	Diamètre moyen relatif	[Bac99]	$Dmm(P) = 100 \times \frac{dmm(P)}{diam(\Omega)}$	$[0; 100]$	x
	Variation du diamètre moyen relatif	[Bac99]	$\Delta Dmm = Dmm(P_{init}) - Dmm(P^*)$	$[-100; 100]$	x
	Entropie	[Gre87]	$E(X) = -\sum_{x \in \Omega} \mathbb{P}(X = x) \ln \mathbb{P}(X = x)$	$[0; \infty[$	(x)
Espace objectif	Amplitude	[Bac99]	$Amp(P) = 100 \times \frac{\lvert P \rvert (max_{s \in P} f(s) - min_{s \in P} f(s))}{\sum_{s \in P} f(s)}$	$[0; \infty[$	-
	Variation relative d'amplitude	[Bac99]	$\Delta_{Amp} = \frac{Amp(P_{init}) - Amp(P^*)}{Amp(P_{init})}$	$]\infty; 1]$	-
	Gap	[Bac99]	$Gap(P^*) = 100 \times \frac{\sum_{s \in P*} (f(s) - f(s^*))}{\lvert P^* \rvert . f(s^*)}$	$[0; \infty[$	-

Vues d'analyse	Indicateurs	Référence	Formule	Domaine de valeurs	Distance
Paysage complet	Coefficient de corrélation fitness-distance	[Wei90]	$FDC \approx \frac{1}{\sigma(f)\sigma(d)} \frac{1}{m} \sum_{i=1}^{m} (f(s_i) - \bar{f})(d_{opt}(s_i) - \bar{d})$	$[-1; 1]$	x
	Autocorrelation	[Wei90]			
	Fonction d'autocorrélation		$\rho(d) = \frac{1}{\sigma^2(f)N} \sum_{\substack{(x,y)\in\Omega\times\Omega \\ dist(x,y)=d}} (f(x) - \bar{f})(f(y) - \bar{f})$	$[-1; 1]$	x
	Fonction de corrélation d'une marche aléatoire		$r(s) = \frac{1}{\sigma^2(f)(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \bar{f})(f(x_{t+s}) - \bar{f})$	$[-1; 1]$	-
	Longueur de corrélation	[Wei90]	$l = -\frac{1}{\ln(\lvert\rho(1)\rvert)}$ ou $-\frac{1}{\ln(\lvert r(1)\rvert)}$	$[0; \infty[$	(x)
	Longueur de marche moyenne relative	[Bac99]	$Lmm(P) = 100 \times \frac{lmm(P)}{diam(\Omega)}$	$[0; \infty[$	(x)
	Longueur d'autocorrélation	[AZ98]	$l' = -\frac{1}{1-\rho(1)} = -\frac{1}{1-r(1)}$	$[0; \infty[$	(x)
	Taux d'évasion	[Mer04]	$\tau_d = \frac{1}{\lvert P^*\rvert} \sum_{s_i^* \in P^*} \mathbb{P}(HillClimbing(s) \neq s_i^* \mid d(s, s_i^*) = d)$	$[0; 1]$	x
	Point de qualité	[Dra10]	$PQ(s) = \frac{\lvert E(s')\rvert}{\lvert \mathcal{V}(s)\rvert}$ $E(s') = \{s' \mid (d_{opt}(s') \leq d_{opt}(s) \wedge f(s') \leq f(s)) \vee (d_{opt}(s') \geq d_{opt}(s) \wedge f(s') \geq f(s))\}$	$[0; 1]$	x

1.5 Problèmes étudiés

Au cours de cette thèse, différents problèmes d'optimisation combinatoire seront étudiés et utilisés pour analyser leur(s) paysage(s) et/ou tester les performances des métaheuristiques.

1.5.1 Les paysages-NK

Kauffman [Kau93] a introduit les paysages-NK comme un moyen d'expérimenter les propriétés de paysages d'optimisation suivant des paramètres variables (ex : taille N, rugosité). Ces paysages purement académiques règlent à travers le paramètre K l'épistasie entre les gènes d'une solution *i.e.* leurs interactions.

1.5.1.1 Définition du problème

L'ensemble des solutions de ces paysages sont dans l'espace $\{0;1\}^N$ à N dimensions.
K $(\in [|0; N-1|])$ est le degré d'interactions entre les variables *i.e.* l'épistasie du problème. Le modèle étant épistatique, chaque coordonnée d'une solution $s = (s_1, ..., s_n)$ est liée avec K autres coordonnées. Le coût associé à une solution tient compte de ces interdépendances. Pour une coordonnée $s_i \in \{0;1\}$, sa contribution $f_i : \{0;1\}^{K+1} \to \mathcal{U}([0;1])$ dépend de K autres coordonnées notées $s_{i_1}, ..., s_{i_K}$. Elle est calculée telle que : f_i associe une valeur choisie uniformément dans $([0;1])$ à toute combinaison de $\{0;1\}^{K+1}$. On crée alors une table des contributions qui associe à tout élément de $\{0;1\}^{K+1}$ une valeur dans $[0;1]$.
Ainsi, pour une solution s, la fonction objectif f est définie telle que :

$$f(s) = \frac{1}{N} \sum_{i=1}^{N} f_i(s_i, s_{i_1}, ..., s_{i_K})$$

On cherche à maximiser cette fonction objectif.
Notons qu'il y a deux manières différentes de choisir les K coordonnées qui interagissent avec s_i : (i)soit elles sont choisies aléatoirement parmi les $N-1$ coordonnées restantes : l'épistasie est dite *aléatoire*, (ii) soit elles sont réparties équitablement autour de s_i : l'épistasie est dite *centrée*. La figure 1.5 illustre l'épistasie centrée.

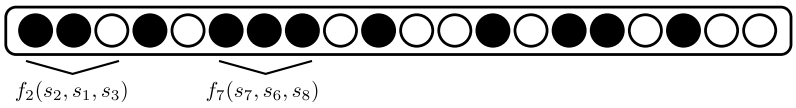


Figure 1.5 – Élément d'un paysage-NK avec $N = 20$ et $K = 2$, épistasie centrée. Le noir et le blanc représentent respectivement les bits de valeur 1 et 0.

Pour le problème ainsi défini, la rugosité du paysage dépend de la valeur de K. En effet, une relation de voisinage entraîne une légère variation sur la solution. Plus les gènes interagissent entre eux (K grand), plus la valeur de la contribution de chacun sera impactée lors de la variation. Entre les voisins, les valeurs de fitness ont alors plus tendance à être différentes. Plus K est petit, plus il est probable d'avoir des séquences identiques entre solutions voisines et donc d'avoir des contributions identiques pour chacun des gènes.

Pour jouer sur la rugosité, des variantes aux paysages-NK ont été proposées et ainsi ajouter de la neutralité. Barnett [Bar01] a proposé le paysage-NKp où les fonctions de contributions f_i sont particularisées, telles qu'il y ait une probabilité p qu'elles deviennent nulles, $\mathbb{P}(f_i(s_i, s_{i_1}, ..., s_{i_K}) = 0) = p$. La table des contributions fait donc apparaître des zéros avec une probabilité p. Plus p est proche de 1, plus le paysage est neutre.
Newman *et al.* [NE98] ont, quant à eux, proposé le paysage-NKq et particularisé les contributions f_i en les générant différemment. Les contributions de la table sont donc effacées puis remplacées selon la loi uniforme $\mathcal{U}([0; q-1])$. Les contributions sont alors données par le rapport entre cette valeur et $q-1$, ce qui permet de rester dans l'intervalle $[0; 1]$. Plus q est petit, plus le paysage est neutre.
La figure 1.6 illustre sur un exemple les modifications apportées aux contributions pour NKp et NKq dans le cas où $K = 2$.

	NK	NKp	NKq
●●●	0.23	0.0	0
○●●	0.77	0.77	1
●○●	0.34	0.0	1
○○●	0.22	0.0	0
●●○	0.63	0.0	0
○●○	0.57	0.57	0
●○○	0.18	0.0	1
○○○	0.49	0.0	1

Figure 1.6 – Exemple d'une table des contributions de f_i d'un paysage-NK, $K = 2$ (épistasie centrée). Dans cet exemple, $p = 0.75$ pour le paysage-NKp et $q = 2$ pour le paysage-NKq. La première colonne décrit toute les configurations possible de $\{0; 1\}^3$. Les autres colonnes donnent les valeurs des contributions de f_i pour chacune des configurations pour les paysages NK, NKp et NKq.

1.5.1.2 Jeux de données

Dans le cadre d'expérimentations sur les paysages-NK, il suffit de donner une valeur aux différents paramètres :
– la taille (N),
– le degré de rugosité (K),
– le modèle d'épistasie (aléatoire ou centrée),
– la probabilité (p) d'avoir des contributions nulles, pour le modèle NKp,
– la borne supérieure des valeurs de contributions (q), pour le modèle NKq.
L'instance générée du problème ne dépend alors plus que du générateur de nombres aléatoires choisi et de sa graine.

1.5.1.3 Modélisation classique du problème

1.5.1.3.1 Représentation Les solutions des paysages-NKappartiennent à l'espace $\{0;1\}^N$. On les représente à l'aide d'un vecteur de N bits.

1.5.1.3.2 Évaluation L'évaluation d'une solution consiste à calculer la valeur de la fonction objectif *i.e.* la somme des contributions de chaque bit de cette solution.

1.5.1.3.3 Voisinage La relation de voisinage, choisie pour les paysages-NK, est celle associée à la distance de Hamming. Pour toute solution de $\{0;1\}^N$, les solutions qui lui sont voisines, sont celles qui n'ont qu'un seul bit de différence par rapport à elle (opérateur 1-flip). La taille du voisinage d'une solution est donc égale à N.

1.5.1.3.4 Initialisation On crée un vecteur aléatoire de N bits.

1.5.1.4 Méthodes de résolution

Dans cette thèse, nous utiliserons uniquement les paysages-NKq (parmi les trois présentés) comme cadre applicatif. Geard *et al.* [GWH+02] montrent dans une étude comparative des trois types de paysage (NK, NKp et NKq) que la manière dont on crée la neutralité dans le problème est significative : les paysages-NKq sont plus proches des paysages-NK. En effet contrairement aux paysages-NKp, les paysages-NKq correspondent aux paysages-NK où des collines auraient été transformées en plateau. La distribution de la neutralité est moins impactée par la rugosité du problème ce qui fait que les paramètres K et q peuvent influencer la topographie du paysage de façon plus indépendante.
La plupart des travaux de la littérature s'attache à étudier les caractéristiques des paysages-NKq mais peu utilisent les propriétés de ces paysages pour tester l'efficacité des méthodes de résolution. Nous avons retenu le travail de Verel *et al.* [VCC04] qui utilise les propriétés de neutralité du paysage pour concevoir une méthode efficace appelée *scuba search* qui se déplace sur le voisin de la solution courante qui a le plus de chance de conduire aux meilleurs optima. Évidemment sur un paysage neutre, ce voisin peut avoir la même fitness que la solution courante. Les performances de cette méthode sont comparées à celles d'un Hill Climbing à un ou deux pas et à un Netcrawler. Les auteurs montrent que leur méthode est plus efficace sur les instances testées.

1.5.2 Problème de flowshop

Le second problème que nous utilisons est un problème d'ordonnancement de type flowshop (*Flowshop Scheduling Problem*, FSP) qui fait partie des problèmes d'ordonnancement les plus étudiés de la littérature. Les problèmes d'ordonnancement se composent de tâches à effectuer sur plusieurs machines disponibles. Dans le cas particulier du flowshop, les machines sont utilisées dans le même ordre pour toutes les tâches. Ces tâches sont caractérisées par leur durée d'exécution sur chaque machine. Principalement, on cherche à minimiser la date d'achèvement de l'ordonnancement (*makespan*, en anglais) *i.e.* la date d'achèvement de la dernière tâche ordonnancée sur la dernière machine. Le problème de

flowshop est un problème d'optimisation qui a pour but de minimiser le temps de réalisation de toutes les tâches. Nous nous intéressons dans cette thèse au problème de flowshop de permutation où la séquence des jobs est identique et unidirectionnelle sur chacune des machines, comme illustré sur la figure 1.7.

1.5.2.1 Définition du problème

Le problème de flowshop consiste à ordonnancer un ensemble de N jobs $\{J_1, J_2, ..., J_N\}$ sur un ensemble de M machines $\{M_1, M_2, ..., M_M\}$. Les machines sont des ressources dites critiques car au plus une tâche peut s'exécuter à un même instant sur une machine. Un job J_i est composé de M tâches consécutives à réaliser dans l'ordre sur les M machines. Chaque tâche a une durée d'exécution spécifique sur chaque machine. On note p_{ij} la durée de la $j$ème tâche du job J_i *i.e.* le temps nécessaire à la machine M_j pour réaliser le job J_i. Ici, nous cherchons à minimiser la date d'achèvement de la dernière tâche ordonnancée, note C_{max}. Si on note C_i la date d'achèvement du job J_i sur la machine M_M, la fonction objectif de ce problème s'écrit :

$$C_{max} = \max_{i \in \{1,...,N\}} C_i$$

1.5.2.2 Jeux de données

1.5.2.2.1 Instances aléatoires Taillard [Tai93] a proposé des jeux d'instances de problèmes de type flowshop de permutation pour minimiser la date d'achèvement de la dernière tâche ordonnancée. Ces instances sont très utilisées dans la littérature [Stü98, RM05].
Différentes tailles sont disponibles. Le nombre N de jobs à ordonnancer sur une machine appartient à l'ensemble $\{20; 50; 100; 200; 500\}$. Le nombre M de machines sur lesquelles sont ordonnancées ces tâches appartient à l'ensemble $\{5; 10; 20\}$. La durée p_{ij} d'exécution du job J_i sur la machine M_j est générée aléatoirement selon la distribution uniforme $\mathcal{U}([1; 99])$. Pour chaque taille de problème $(N \times M)$, dix instances sont disponibles sur sa page web [1], on les note $N/M/k$ avec $k \in [1; 10]$. Cette page web recense les solutions optimales des instances résolues dans le cas contraire, les bornes inférieure et supérieure sont données. On remarque que très peu d'instances avec 20 machines sont résolues de manière optimale. Les instances avec 5 et 10 machines ont été résolues de manière optimale mais cette résolution a pu nécessiter un temps très long de calcul, notamment par la mise en œuvre de méthodes exactes. Le nombre de machines paraît donc déterminant, en plus de la taille de la représentation (le nombre de tâches), dans la difficulté à résoudre l'instance du problème posé.

1.5.2.2.2 Instances structurées Les instances aléatoires de Taillard sont les plus utilisées dans la littérature. Or, les instances aléatoires peuvent ne pas être assez représentatives des instances réelles. Watson *et al.* [WBWH02] ont donc proposé des instances dites structurées pour le problème de flowshop de permutation de minimisation pour la date d'achèvement de la dernière tâche ordonnancée. Ces instances présentent trois

1. http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

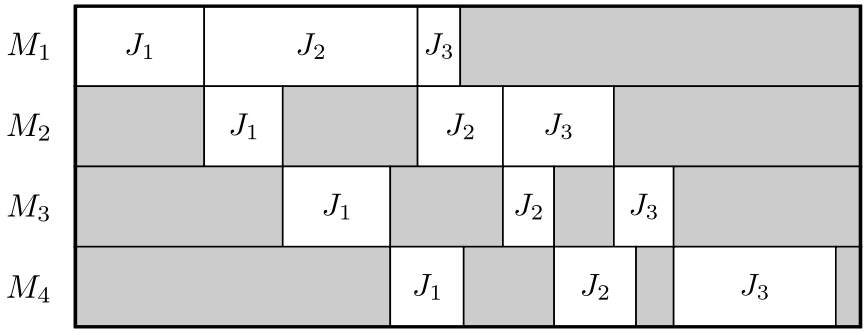


Figure 1.7 – Exemple d'une solution d'un problème de flowshop de permutation où trois jobs (J_1, J_2, J_3) sont ordonnancés sur quatre machines (M_1, M_2, M_3, M_4)

types de structurations différentes quant à la génération selon la distribution uniforme des durées opératoires de chaque tâche sur chaque machine. Ces durées sont corrélées (i) par rapport aux jobs (instances jobs-corrélées), (ii) par rapport aux machines (instances machines-corrélées) ou (iii) par rapport aux jobs et aux machines (instances jobs/machines-corrélées). De plus, un paramètre $\alpha \in [0; 1]$ règle le degré de corrélation. Quand $\alpha = 0$, les intervalles des distributions uniformes sont identiques, les instances sont alors proches des instances aléatoires de Taillard. À mesure que α croît, les intervalles se différencient et les instances deviennent de plus en plus structurées. Différentes tailles sont disponibles. Le nombre N de jobs à exécuter sur une machine appartient à l'ensemble $\{20; 50; 100; 200\}$. Le nombre M de machines sur lesquelles sont ordonnancées les tâches est égal à 20. Seules des instances pour 20 machines ont été générées car elles représentent les plus difficiles de la littérature selon Taillard. Pour chaque taille $(N \times M)$ et chaque $\alpha \in \{0.0; 0.1; 0.2; ...; 1.0\}$, 100 instances ont été générées et sont disponibles sur le web [2].

1.5.2.3 Modélisation

1.5.2.3.1 Représentation Pour représenter une solution du problème de flowshop de permutation, il suffit de connaître l'ordre d'exécution des jobs. Les solutions réalisables sont alors représentées à l'aide de permutation. Pour une instance de N jobs, N est la taille de la permutation et $N!$, la taille de l'espace de recherche *i.e.* le nombre de solutions réalisables.

1.5.2.3.2 Évaluation L'évaluation d'une solution consiste à calculer sa fonction objectif *i.e.* son makespan C_{max}. Celle-ci est complètement déterminée par l'ordre des jobs puisque chaque job doit être ordonnancé sur l'ensemble des machines.

1.5.2.3.3 Voisinages Pour les problèmes de permutation, tel le problème de type flowshop que nous considérons, il existe principalement trois opérateurs de voisinage [SS07] : insertion, échange adjacent ou non. L'échange adjacent (*swap* en anglais) étant peu utilisé au sein des métaheuristiques, nous ne présentons ici que les deux autres.

2. `http://www.cs.colostate.edu/sched/generator`

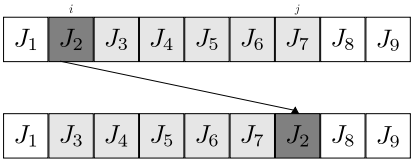


Figure 1.8 – Opérateur *insertion* pour le problème de flowshop.

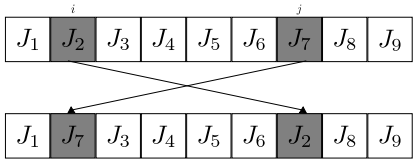


Figure 1.9 – Opérateur *échange* pour le problème de flowshop.

Insertion (*insert* en anglais) : cet opérateur consiste à changer de place un job en l'insérant à une autre place dans la permutation. Ainsi, pour un job positionné en i, il peut être réinséré à une position $j \neq i$. Les jobs positionnés entre ces deux positions se retrouvent donc décalés. La figure 1.8 illustre l'opérateur d'insertion. À partir d'une solution, on peut créer $(N-1)^2$ solutions différentes.

Échange (*exchange* ou *interchange* en anglais) : cet opérateur consiste à échanger la place de deux jobs de la permutation. Ainsi, pour les positions i et j ($i \neq j$), le job en i se déplace en j et le job en j se déplace en i. La figure 1.9 illustre l'opérateur d'échange. À partir d'une solution, on peut créer $N(N-1)/2$ solutions voisines.

1.5.2.3.4 Initialisation Pour une instance composée de N jobs à effectuer, on crée aléatoirement un vecteur de permutation de taille N.

1.5.2.4 Méthodes de résolution

Comme déjà dit précédemment, le problème de flowshop est très étudié dans la littérature. Il fait partie généralement des problèmes d'optimisation combinatoire NP-difficile. Certaines méthodes exactes ont été utilisées pour résoudre des variantes du flowshop [KS75, RMB98, RMB99]. Ces méthodes s'avèrent peu rapides du fait de la taille du problème donc des méthodes approchées qui donnent des résultats proches de l'optimum en un temps raisonnable ont été développées. Ces méthodes peuvent être classées selon que la recherche de l'optimum se fait par des heuristiques de construction de solutions [CDS70, Dan77, Kou98, NEH83] ou par des recherches locales qui améliorent la solution courante par des petites modifications dans la solution. On peut citer la recherche tabou [GW04, NS96, Ree93, Tai90, WH89], le recuit simulé [IMT95, OS90, OP89] et des recherches locales itérées [Stü98, RS07]. Des méthodes évolutionnaires telles que les algorithmes génétiques [SB92, Ree95, RY98, RMA06] et les colonies de fourmis [Stü97] ont aussi été mises en œuvre et analysées sur le problème de type flowshop à permutation.

1.5.3 Problème asymétrique de tournées de véhicules

Les problèmes de tournées de véhicules sont des problèmes issus de la recherche opérationnelle. Il s'agit de déterminer les tournées d'une flotte de véhicules qui doivent livrer, visiter ou intervenir chez une liste de clients. Chaque client ne doit être présent que dans une et une seule tournée. Le but est de minimiser le coût lié à ces tournées. Ce type de problème est une extension classique du problème du voyageur de commerce.

Ici, nous nous intéressons aux problèmes de tournées de véhicules avec contrainte de capacité (Capacited Vehicle Routing Problem (CVRP)) qui est un problème classique de la littérature. Les demandes d'un ensemble de clients doivent être satisfaites grâce à une flotte de véhicules qui ont chacun une capacité limitée, tout en cherchant à minimiser la distance parcourue par l'ensemble de ces véhicules.
Le problème asymétrique de tournées de véhicules avec une flotte fixe hétérogène et contrainte de capacité (Heterogenous Fixed Fleet Asymmetric Vehicle Routing Problem (HFF-AVRP)) proposé par Taillard [Tai99], est une extension plus réaliste du CVRP classique où le nombre de véhicules disponibles est limité, chacun des véhicules est différencié et les distances entre les clients sont asymétriques.

1.5.3.1 Définition du problème

Le HFF-AVRP est défini comme un graphe orienté complet $G = (V, A)$ où :
– $V = (v_0, v_1, v_2, \ldots, v_{nc})$ est l'ensemble des sommets et,
– $A = \{(v_i, v_j) : v_i, v_j \in V\}$ est l'ensemble des arcs pour un problème avec un nombre nc de clients.

Le sommet v_0 correspond au dépôt et les sommets v_1 à v_{nc} correspondent aux clients. Une flotte hétérogène de nv véhicules utilise le dépôt comme point de départ et d'arrivée des tournées à effectuer. Soit w_k le coût par unité de distance du véhicule $k \in \{1, ..., nv\}$. Chaque véhicule disponible peut être au plus affecté à une tournée. Une demande positive (ou nulle) q_i est associée à chaque sommet v_i ($i = 1..nc$). Une distance positive (ou nulle) d_{ij} ($d_{ii} = 0$ pour tout $i = 1...nc$) est associée à chaque arc $(v_i, v_j) \in A$ ($v_i \neq v_j$) du graphe et il existe au moins $i, j \in \{1, ..., nc\}$, $i \neq j$ tels que $d_{ij} \neq d_{ji}$ (la matrice des distance est asymétrique). Soit $\delta_{ij}^k : \Omega \longrightarrow \{0; 1\}$, une application qui à toute solution de l'espace de recherche associe 1 si l'arc (v_i, v_j) est parcouru par le véhicule k, 0 sinon. Ici, le but est de minimiser la somme des coûts de chaque tournée. La fonction objectif s'écrit alors :

$$f(s) = \sum_{k=1}^{nv} w_k \sum_{i=0}^{nc} \sum_{j=0}^{nc} d_{ij} \delta_{ij}^k(s)$$

pour toute solution s vérifiant les trois contraintes : (i) chaque véhicule commence et termine sa tournée au dépôt, (ii) chaque client doit appartenir à une et une seule tournée et sa demande doit être satisfaite entièrement, (iii) la demande totale des clients d'une tournée ne doit pas excéder la capacité du véhicule qui lui est affecté.

1.5.3.2 Jeux de données

Fischetti *et al.* [FTV94] ont proposé 8 instances pour le problème asymétrique de tournées de véhicules. Étant donné que les meilleures solutions à ce problème sont construites avec deux ou trois véhicules, nous avons utilisé ces jeux en imposant une flotte fixe de 10 véhicules de capacité identique. Dans le but de simuler le caractère hétérogène de cette flotte, chaque véhicule est affecté d'un coût unitaire différent *i.e.* d'une pondération de la distance qu'il doit parcourir. Ce coût est distribué uniformément dans l'intervalle $[1; 1.3[$ comme suit :

$$\{1, 1.03, 1.06, 1.09, 1.12, 1.15, 1.18, 1.21, 1.24, 1.27\}$$

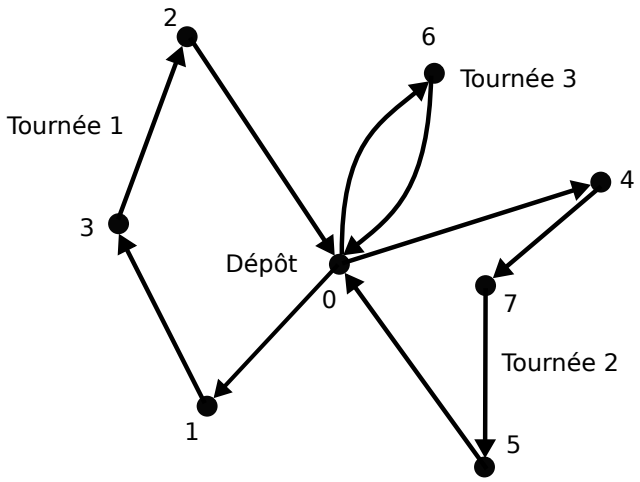


Figure 1.10 – Exemple d'une solution du problème de HFF-AVRP où 7 clients sont visités par une flotte de 3 véhicules (5 véhicules étant disponibles).

Le coût du trajet d'un véhicule est donc obtenu en multipliant la distance qu'il doit parcourir par son coût unitaire.

1.5.3.3 Modélisation

1.5.3.3.1 Représentation Pour le problème de tournées de véhicules, il a été proposé et utilisé plusieurs représentations différentes des solutions. La représentation joue un rôle très important dans l'étude d'un problème, car elle impacte directement l'efficacité des méthodes de résolution.
La figure 1.10 illustre une solution envisageable pour une instance du problème de HFF-AVRP avec 7 clients et 5 véhicules disponibles. Ici, seulement les trois premiers véhicules sont utilisés. Le premier véhicule visite le client 1, puis le 3 pour finir par le 2, le deuxième visite le client 4 puis 7 puis 5 et le troisième visite seulement le client 6. Le quatrième et le cinquième véhicule ne desservent aucun client. Cet exemple est utilisé pour décrire les représentations proposées dans la littérature.
En 1983, Beasley [Bea83] introduit la représentation des solutions sous forme de *big tour*. Cela consiste à lister les clients dans l'ordre de leur visite puis d'appliquer un algorithme de coupure pour construire des tournées qui respectent toutes les contraintes posées, c'est le principe de *route first / cluster second*. Les solutions sont ensuite représentées sous forme de listes de tournées où les tournées correspondent elles-mêmes à une liste de clients ordonnés. La représentation de la solution illustrée par la figure 1.10 est avant la coupure (1 3 2 4 7 5 6) puis après (1 3 2, 4 7 5, $6, \varnothing, \varnothing$).
Cette représentation des solutions, dans sa représentation finale, ressemble à celle choisie par Gendreau *et al.* [GHL94] et par Kubiak [Kub07], qui considère une solution sous forme d'un ensemble de tournées, une tournée étant elle-même un ensemble de clients donnés dans l'ordre de visite. Cette représentation est souvent utilisée, car elle permet d'accéder rapidement aux clients présents dans une tournée précisément. La représentation de la solution illustrée par la figure 1.10 est alors [[1 3 2], [4 7 5], [6], [], []].

La troisième représentation de solution utilisée, est inspirée elle aussi d'une représentation d'une solution pour le problème du voyageur de commerce. En effet, on utilise un vecteur de taille, le nombre de clients. La coordonnée 1 représente le client 1, la coordonnée 2, le client 2 et ainsi de suite. On représente notre solution de manière à ce que le numéro de la coordonnée considérée corresponde au client prédécesseur du client indiqué dans cette même coordonnée. Quand un client termine une tournée, alors on met un zéro comme valeur pour la coordonnée correspondante. La représentation de la solution illustrée par la figure 1.10 serait $[\ \overset{1}{3}\ \overset{2}{0}\ \overset{3}{2}\ \overset{4}{7}\ \overset{5}{0}\ \overset{6}{0}\ \overset{7}{5}\ ]$. Une variation à cette représentation consiste à mettre un chiffre négatif correspondant au numéro de tournée, plutôt qu'un zéro car, avec certaines contraintes, toutes les tournées ne peuvent pas être considérées dans le désordre (ex : un camion qui fait deux tours dans la journée avec des clients qui ne peuvent être livrés que le matin).
Pour l'analyse de paysage, nous avons vu qu'il est nécessaire de définir une distance entre les solutions envisageables du problème traité. Nous avons choisi une représentation des solutions du problème de HFF-AVRP qui nous semble plus évidente que celles proposées dans la littérature pour travailler par la suite sur la distance. De plus, la flotte de véhicule étant hétérogène, la représentation doit permettre de repérer le véhicule affecté à une tournée précise. Cette représentation est inspirée de celle du *big tour*. Chaque client est représenté par un numéro entre 1 et le nombre total de clients, et le dépôt est représenté par un zéro. Elle ressemble à une permutation où le zéro est présent plusieurs fois. Une solution envisageable est représentée par une liste de clients intercalés par les dépôts ; la liste commence et finit par un zéro. Une tournée correspond à la succession dans l'ordre des clients entre chaque dépôt (zéro). La représentation de la solution illustrée par la figure 1.10 est alors (0 1 3 2 0 4 7 5 0 6 0 0 0).

1.5.3.3.2 Évaluation L'évaluation d'une solution consiste à calculer sa fonction objectif *i.e.* la somme des distances parcourues par chacun des véhicules multipliées par le coût associé à chacun d'eux.

1.5.3.3.3 Voisinage Les opérateurs *insertion* et *échange* sont des opérateurs classiques de la littérature. Nous avons donc choisi de les adapter à la représentation que nous proposons. Soit $N = nc+nv-1$ qui correspond au nombre d'éléments qui peuvent changer de place dans la solution.

Insertion : cet opérateur consiste à changer de place un client ou un zéro (représentant la fin d'une tournée) en l'insérant à une autre place dans la solution. Ainsi pour un élément positionné en i, il peut être réinséré à une position $j \neq i$. Les éléments positionnés entre ces deux positions se retrouvent donc décalés. À partir d'une solution, on peut créer $(N-1)^2$ solutions différentes mais toutes ces solutions ne sont pas toutes réalisables car les contraintes peuvent ne pas être respectées (en particulier, les contraintes de capacité).

Échange : cet opérateur consiste à échanger la place de deux éléments de la solution. Ainsi, pour les positions i et j ($i \neq j$), l'élément i se déplace en j et l'élément en j se déplace en i. On interdit que les éléments en position i et j soient tous les deux des zéros sinon l'échange donne la même solution. À partir d'une solution, on peut créer $N(N-1)/2$ solutions différentes mais toutes ces solutions ne sont pas toutes réalisables

car les contraintes peuvent ne pas être respectées (en particulier, les contraintes de capacité).

Pour les problèmes de tournées de véhicules, l'opérateur *2-opt* est connu pour être efficace, mais à notre connaissance, il est actuellement trop difficile de trouver un algorithme de calcul de la distance entre deux solutions pour cet opérateur. C'est la raison pour laquelle nous avons choisi de ne pas l'étudier cet opérateur de voisinage dans cette thèse.

1.5.3.3.4 Initialisation Pour créer une solution initiale, on commence par générer aléatoirement une permutation avec les clients. Puis on ajoute un zéro au début et à la fin, puis pour respecter les contraintes de capacité, on sature les tournées en ajoutant un zéro entre deux clients quand la demande du second fait excéder la capacité totale de la demande de l'ensemble des clients de la tournée. Ensuite, s'il reste des véhicules qui ne sont pas affectés à des clients, on les rajoute au hasard dans la représentation.

1.5.3.4 Méthodes de résolution

Les problèmes asymétriques de tournées de véhicules ayant une flotte fixe et hétérogène ne font pas partie des cas classiques d'études. Néanmoins des méthodes sophistiquées ont été proposées dans la littérature pour résoudre les problèmes avec une flotte hétérogène. Prins [Pri09] décrit deux versions d'un algorithme mémétique *i.e.* une métaheuristique hybridant un algorithme génétique et une recherche locale pour résoudre plusieurs variantes du problème de tournées de véhicules avec une flotte hétérogène. Dans l'étape de mutation de l'algorithme génétique, deux recherches locales sont proposées. La première consiste à appliquer avec une certaine probabilité une méthode de descente alors que la deuxième calcule une distance basée sur la représentation des solutions [SS06], pour tester la diversité. Il montre que cet algorithme est efficace et arrive même à trouver des nouveaux *best-known*. Gendreau *et al.* [GLMT99] ont conçu une recherche tabou qui exploite les caractéristiques du problème de tournées avec une flotte hétérogène et embarque un mécanisme d'échange de flotte. Cette heuristique montre de bonnes performances et trouve elle aussi de nouveaux *best-known*. Tarantilis *et al.* [TKV04] proposent quant à eux, BATA, une recherche locale utilisant des opérateurs de voisinages tels que le *2-Opt* et l'*échange* de nœuds ou d'arcs. BATA autorise à bouger vers une solution de moins bonne qualité grâce à un critère d'acceptation qui évolue non pas de manière linéaire au cours du temps mais de manière adaptative avec un système de *backtracking*. Les expérimentations montrent que BATA est robuste et efficace. Notons que ces méthodes ont été testées dans le cas où les distances ne sont pas asymétriques et la flotte n'est pas fixe mais peuvent représenter de bonnes pistes d'analyse pour attaquer ce problème en particulier.

Ces problèmes d'optimisation combinatoires sont de nature et difficulté différentes. Les paysages-NK font partie d'une classe de problèmes purement académiques alors que les problèmes de type flowshop de permutation et de tournées de véhicules sont issus de la logistique et sont donc proches des problèmes réels. Dans cette thèse, nous utilisons ces problèmes pour les différentes expérimentations que nous mettons en œuvre.

1.6 Conclusion

Ce chapitre expose le contexte scientifique des travaux de ce mémoire. Beaucoup de problèmes réels peuvent s'exprimer sous la forme d'un problème d'optimisation combinatoire. Qu'il s'agisse de problèmes de logistique, de production, d'extraction de connaissances... les problèmes d'optimisation sont généralement très difficiles à résoudre. Il existe deux catégories de méthodes pour les résoudre : les méthodes exactes, qui assurent de trouver la meilleure solution réalisable mais souvent en un temps exponentiellement croissant avec la taille du problème et les méthodes approchées, comme les métaheuristiques, qui se veulent plus rapides et qui tentent de s'approcher de la meilleure solution réalisable sans garantie. Dans ce mémoire, ce sont ces dernières méthodes qui nous intéressent.

Les métaheuristiques peuvent être classées selon deux groupes : les recherches locales, qui font évoluer une solution unique et les méthodes à population de solutions. Les métaheuristiques se veulent génériques afin d'être utilisées pour résoudre tout type de problèmes d'optimisation. Ainsi, elles ont, en général, un certain nombre de paramètres qui permettent de les adapter efficacement au problème traité. Cependant, ces paramètres sont fastidieux à régler et il n'existe pas vraiment de règles pour y arriver facilement. Il est alors possible d'avoir recours à l'analyse de la structure du problème d'optimisation.

Dans ce cadre, le paysage d'un problème d'optimisation correspond à la description d'une structure associée au problème pour mieux le caractériser et le comprendre. Le paysage est défini au moyen d'une relation de voisinage et d'une fonction d'évaluation des solutions de l'espace de recherche. Ainsi, le problème se structure sous ces aspects et il est alors possible de le caractériser par des indicateurs spécifiques. Nous avons présenté un état de l'art des indicateurs de la littérature que nous avons classés en trois catégories suivant qu'ils caractérisent l'espace de recherche, ou l'espace des valeurs de fitness ou encore le lien entre les deux. Dans ce mémoire, l'étude de la structure du paysage permettra, entre autre, de comprendre la dynamique des métaheuristiques.

Dans ce chapitre, plusieurs problèmes d'optimisation combinatoire ont été introduits car ils seront utilisés au cours des différentes analyses et expérimentations. Les paysages-NK sont des problèmes académiques classiques permettant d'étudier la pertinence des indicateurs et la dynamique des métaheuristiques. Le problème de flowshop est un problème classique d'ordonnancement qu'il est intéressant d'étudier car proche des problèmes réels. Le problème de tournées de véhicules est un problème de logistique très étudié en entreprise mais la difficulté à bien le modéliser le rend très complexe à résoudre. Dans la suite, nous étudierons le paysage de plusieurs instances de ces problèmes afin de créer un lien avec la conception de méthodes de résolution génériques et efficaces.

Chapitre 2

Influence du paysage sur les performances des métaheuristiques

Publications en lien avec le chapitre :

- M.-É. Marmion, L. Jourdan, C. Dhaenens, Fitness Landscape Analysis and Metaheuristics Efficiency, *Journal of Mathematical Modelling and Algorithms*, 2012
- M.-É. Marmion, J. Humeau, L. Jourdan, C. Dhaenens, Comparison of Neighborhood for the HFF-AVRP, In *Proceedings of the 8th ACS/IEEE International Conference on Computer Systems and Applications, NATCOMP workshop*, AICCSA 2010, IEEE Press, 2010

Sommaire

Nous avons vu dans le chapitre précédent que l'analyse de paysage est un outil de modélisation et de compréhension de la structure inhérente à un problème d'optimisation combinatoire. Or le choix d'une méthode robuste de résolution d'un problème d'optimisation est très difficile. En particulier, la relation de voisinage est un paramètre central de la plupart des métaheuristiques. En effet, les recherches locales utilisent une relation de voisinage identifiable à celle utilisée dans la définition du paysage. Quant aux méthodes à population de solutions, elles l'utilisent souvent dans l'opérateur de mutation. Il est donc légitime de se demander si le paysage d'un problème joue un rôle dans les performances d'une métaheuristique appliquée à ce problème et comment une analyse de paysage peut aider à comparer les performances de deux métaheuristiques dont le seul paramètre qui diffère serait leur relation de voisinage.
A partir de l'état de l'art sur l'analyse de paysage présenté dans le chapitre 1, nous présentons plus précisément les indicateurs qui nous paraissent plus pertinents et une démarche pour étudier le paysage d'un problème d'optimisation. Ainsi, nous serons en mesure de mener une analyse de paysage pour deux problèmes d'optimisation issus de la logistique et nous discuterons de l'influence du paysage sur les performances des métaheuristiques classiques de la littérature.

2.1 Proposition d'une approche pour l'analyse de paysage

Comme présenté dans le chapitre 1, il existe plusieurs indicateurs pour analyser le paysage d'un problème d'optimisation. Ces indicateurs sont plus ou moins faciles à implémenter et/ou plus ou moins significatifs selon le type de problème à optimiser. Ici, nous présentons les indicateurs qui nous ont paru les plus pertinents pour caractériser le paysage et expliquer l'influence de celui-ci dans le processus d'optimisation. Ces indicateurs sont généralisables donc utilisables pour étudier la majorité des problèmes rencontrés. Dans la suite, une description plus précise de chaque indicateur est donnée ainsi que leur intérêt quant à notre étude sur l'influence du paysage dans le processus d'optimisation des métaheuristiques.

2.1.1 Indicateurs pour l'espace de recherche interconnecté

La taille de l'espace de recherche, *i.e.* le nombre de solutions réalisables, est connue. Or cette taille est souvent telle que toutes les solutions ne peuvent être énumérées en un temps fini. Il est donc nécessaire de pouvoir caractériser l'espace de recherche afin d'utiliser ces informations pour trouver les solutions de bonne qualité. Nous utilisons alors des indicateurs qui permettent d'estimer la largeur de l'espace de recherche ou qui testent si les populations de solutions considérées sont bien diversifiées.

Distances dans une population : La distance entre les solutions d'une population permet d'apprécier si les solutions sont proches ou non relativement à la relation de voisinage choisie. En effet, deux relations de voisinage induisent des interconnexions différentes entre les solutions de l'espace de recherche. Considérant deux relations de voisinage, nous pouvons mesurer l'impact de chacune sur le paysage en étudiant les distributions des distances deux à deux d'une même population de solution. Ainsi, les valeurs maximum et

minimum informent sur l'étendue de cette population dans l'espace de recherche. Nous retrouvons ici, la notion de diamètre de l'espace de recherche présentée dans le chapitre 1. De même les quartiles informent sur la dispersion des solutions au sein de cette population : sont-elles toutes très proches ? ou également réparties sur l'espace de recherche ? En outre, pour une relation de voisinage, la comparaison des distances entre les solutions d'une population choisies aléatoirement sans remise dans l'espace de recherche et une population d'optima locaux peut informer sur l'intérêt d'exploiter une zone de l'espace de recherche ou au contraire d'en explorer des différentes.

Entropie : L'entropie mesure la dissimilarité entre les solutions d'une population suivant une loi de probabilité choisie. Nous avons choisi de regarder les dissimilarités en terme de représentation des solutions. Ainsi, nous pensons que cette mesure complète la mesure de distance qui caractérise uniquement les interconnexions entre les solutions de l'espace de recherche. Ainsi, pour une population de solutions donnée, l'impact de la relation de voisinage sur l'interconnexion de solutions avec des représentations très semblables peut être étudié. Une population peut donc être très étendue dans l'espace de recherche mais les solutions peuvent se ressembler et vice-versa.

Ces deux mesures permettent d'apprécier, pour n'importe quelle population, si les solutions sont très éloignées ou non les unes des autres et si elles sont très différentes ou non.

2.1.2 Indicateurs pour l'espace des valeurs de fitness

Pour caractériser cet espace, nous avons choisi de ne regarder que la distribution des valeurs de fitness d'une population. Ainsi, les valeurs maximum et minimum informent sur l'étendue des qualités des solutions et les quartiles sur leur dispersion en terme de valeurs de fitness. La qualité globale d'une population peut ainsi être appréciée. Par exemple, si les valeurs de fitness sont souvent identiques le paysage induit par cette fonction d'évaluation ne pourra être que très plat et donc très difficile à optimiser. Considérant deux relations de voisinage $\mathcal{V}_1$ et $\mathcal{V}_2$, la comparaison entre une population d'optima locaux pour $\mathcal{V}_1$ et d'une population d'optima locaux pour $\mathcal{V}_2$ peut donner des informations quant à leur qualité. Ceci peut aider à comprendre pourquoi une métaheuristique peut être facilement attrapée par des optima locaux de mauvaise qualité.

2.1.3 Le paysage complet

La paysage complet, considérant en même temps la relation de voisinage et la fonction objectif, peut être caractérisé à l'aide de marches. En effet, l'espace de recherche n'étant pas énumérable rapidement, nous cherchons à caractériser certaines zones et la continuité des caractéristiques des solutions voisines. Une marche recueillant des informations sur les solutions qu'elle visite au cours de son déplacement s'avère très utile.

Longueur de marche : Elle se calcule le long d'une méthode de descente de type *Hill Climbing* (strict) où le voisinage est complètement évalué afin de se déplacer à chaque pas vers la meilleure solution. Considérant deux relations de voisinage, la comparaison des longueurs de marche pour les deux paysages induits permet de savoir s'il est possible

avec peu de mouvements de trouver des optima locaux. La comparaison des valeurs de fitness trouvées informe alors sur leur qualité. Cette mesure permet d'appréhender les irrégularités (pics et vallées) d'un paysage.

Autocorrélation : Les valeurs d'autocorrélation correspondent aux valeurs de la fonction d'autocorrélation calculées le long d'une marche aléatoire. Cette fonction informe sur les relations entre des solutions voisines $\rho(1)$ ou k-voisines $\rho(k)$ *i.e.* sur la régularité du paysage. Ici, considérons l'autocorrélation des valeurs de fitness entre voisins de l'espace de recherche. Nous nous intéressons surtout aux premières valeurs d'autocorrélation (k proche de 1) car elles donnent les informations sur le paysage local. Ainsi une valeur d'autocorrélation proche de 1 indique que le paysage est assez lisse localement. En revanche, plus cette valeur est proche de 0, plus le paysage est caractérisé comme très rugueux localement. Dans notre étude, nous effectuons une marche aléatoire de longueur 10 000 ; ce qui nous paraît suffisant pour échantillonner le paysage.

2.1.4 Mise en œuvre de notre approche

Rappelons que le but de ce chapitre est de montrer l'influence du paysage sur les performances des métaheuristiques. Ces dernières ayant pour la plupart une relation de voisinage comme paramètre, nous proposons ici d'étudier le paysage d'un problème suivant deux relations de voisinage différentes dans le but d'en montrer les similarités et les différences. Ensuite, nous montrons en quoi cette étude de paysage est pertinente pour la compréhension des performances des métaheuristiques.

Un paysage est une structure définie par une relation de voisinage $\mathcal{V}$ elle-même définie à partir d'un opérateur de voisinage. Notons $\mathcal{P}_{op1}$ et $\mathcal{P}_{op2}$ deux paysages d'un problème d'optimisation respectivement pour les opérateurs de voisinage $op1$ et $op2$.
Pour comparer deux paysages du point de vue de l'*espace de recherche interconnecté* et de l'*espace des valeurs de fitness*, nous utilisons trois populations : la première, notée *RPop*, est une population formée de solutions choisies aléatoirement dans l'espace de recherche, la deuxième $OPop_{op1}$ et la troisième $OPop_{op2}$ sont des populations d'optima locaux obtenus à partir de chaque solution de *RPop* en réalisant un Hill Climbing ayant une relation de voisinage basée respectivement sur l'un et l'autre des opérateurs op_1 et op_2. Chaque population contient 1000 solutions. Pour chacune des populations, nous calculons les valeurs des indicateurs décrits ci-dessus. Ces valeurs permettent ainsi d'étudier les paysages $\mathcal{P}_{op1}$ et $\mathcal{P}_{op2}$ séparément puis de les comparer.
Cette approche expérimentale étant stochastique, 30 populations aléatoires de 1000 solutions ont été générées pour obtenir 30 populations $OPop_{op1}$ et 30 populations $OPop_{op2}$. Pour chaque taille du jeu de données, nous avons donc 30 résultats par indicateur pour chacun des paysages $\mathcal{P}_{op1}$ et $\mathcal{P}_{op2}$.

Nous venons de présenter les indicateurs de paysage. Dans ce chapitre, nous souhaitons montrer l'influence du paysage sur la dynamique des métaheuristiques et de leurs performances. Prenant deux problèmes classiques d'optimisation combinatoire issus de la logistique, un problème de tournées de véhicules et le problème de flowshop de permutation,

nous analysons pour chacun d'eux deux paysages construits avec deux relations de voisinage différentes. Puis, nous étudions les performances de certaines métaheuristiques pour comprendre comment la structure de paysage induit par l'un ou l'autre des opérateurs favorise ou non le déplacement vers les solutions de bonne qualité.

2.2 Étude du problème HFF–AVRP

2.2.1 Préliminaires

L'analyse de la structure de paysage et plus particulièrement, l'analyse de l'espace de recherche nécessite de calculer la distance entre deux solutions. Rappelons que la distance est le nombre minimal d'applications de l'opérateur de voisinage pour passer d'une solution à une autre. Cette distance n'est pas triviale à calculer et peut représenter un coût non négligeable de l'analyse de paysage. Dans cette section, nous présentons l'*insertion-distance* et l'*échange-distance* qui sont respectivement les distances définies à partir des opérateurs de voisinage *insertion* et *échange*.

2.2.1.1 Insertion-distance

L'*insertion-distance* est basée sur l'opérateur de voisinage d'*insertion*. Elle permet de calculer la distance entre deux solutions du paysage $\mathcal{P}_{IN}$. L'*insertion-distance* est donnée par l'algorithme de programmation dynamique de la *Plus Longue Sous-séquence Commune* (PLSC) entre deux solutions [BSUS72, Cri88]. Cet algorithme est très utilisé en bioinformatique pour calculer les correspondances entre deux séquences d'ADN, par exemple. La distance entre deux solutions correspond à la différence de la taille de la solution $(nc+nv+1)$ par la PLSC. La distance maximale *i.e.* la plus grande distance pouvant être identifiée entre deux solutions est donc le nombre de clients.

2.2.1.2 Échange-distance

L'*échange-distance* est basée sur l'opérateur de voisinage d'*échange*. Elle permet de calculer la distance entre deux solutions du paysage $\mathcal{P}_{EX}$. Notre représentation du problème étant une représentation originale, dédiée à ce problème (section 1.5.3), nous ne connaissons pas d'algorithme qui permette de la calculer facilement. Dans la suite nous proposons un algorithme pour calculer la distance exacte entre les solutions de l'HFF-AVRP.
Pour calculer cette distance, il faut être certain de trouver le nombre minimal d'*échange* pour passer d'une solution s_1 à une autre s_2. Ceci est rendu difficile par la présence de plusieurs "0" dans la représentation des solutions. Bachelet [Bac99] a proposé un algorithme qui permet de calculer avec une complexité en $\mathcal{O}(N)$, la distance exacte entre deux solutions représentées par des permutations. Notre représentation des solutions étant inspirée de celle des permutations, une première idée fut d'adapter cet algorithme à notre cas. Pour pouvoir l'utiliser il est alors nécessaire de différencier chacun des véhicules car son principe est d'utiliser le caractère bijectif d'une permutation. Or, cette différenciation ajoute un biais lors du calcul du nombre d'applications nécessaires de l'opérateur d'*échange* : la distance entre deux solutions se trouve surestimée.

Algorithme 1 Algorithme de l'*échange-distance*

Entrée : s_1, s_2, N
Sortie : *distance* : *Échange-distance* entre s_1 et s_2
$distance \leftarrow 0$
$list \leftarrow ()$
tmp {variable temporaire pour stocker l'index pour appliquer l'*échange*}
pour $i = 1$ à N **faire** {N est la taille des solutions s_1 et s_2}
  si $s_{1i} \neq s_{2i}$ **alors**
    si $list \neq ()$ et il existe j tel que $s_{1list_j} = s_{2i}$ **alors** {$s_{2i} \in list$}
      tmp $\leftarrow list_j$
      si $s_{1i} = 0$ **alors** {0 est remis en place, l'index doit être enlevé de $list$}
        $list \leftarrow list \setminus list_j$
      fin si
    sinon
      si $s_{2i} = 0$ **alors**
        si il existe j tel que $s_{1j} = 0$ et $s_{1i} = s_{2j}$ **alors** {s_{1i} ne peut pas être directement bien replacé}
          tmp $\leftarrow j$
        sinon
          $list \leftarrow list :: i$
        fin si
      sinon {il existe j tel que $s_{1j} = s_{2i}$ *i.e.* $s_{2i} \neq 0$ et $j \notin list$}
        tmp $\leftarrow j$
      fin si
    fin si
    si tmp existe **alors**
      Échange(s_1,i,tmp)
      $distance \leftarrow distance + 1$
    fin si
  fin si
fin pour

Partant de ce constat, il fallait que les *zéros* qui représentent les véhicules restent indifférenciés. Une deuxième idée fut donc de replacer les clients sans s'occuper des *zéros* de la représentation, *i.e.* des véhicules. Remarquons qu'alors tous les zéros sont automatiquement replacés. L'algorithme réarrange de gauche à droite les éléments correspondants aux clients d'une première solution pour qu'elle devienne égale à la deuxième. Dans le cas où un zéro doit être replacé et qu'aucune application de l'opérateur ne permet de replacer correctement le client courant à sa bonne position, la position du zéro est ajoutée à une liste mémoire. Cette liste permet d'avoir une mémoire des positions qui doivent contenir des véhicules, donc être identiques dans les deux solutions considérées, mais qui contiennent encore des clients. Cette liste doit donc être lue à chaque étape de l'algorithme pour vérifier si le client recherché y est. L'algorithme 1 présente ces différentes étapes. Il utilise la procédure Echange(s, i, j) qui intervertit les éléments des positions i et j de s.

Cet algorithme est en $\mathcal{O}(N^2)$. Cet algorithme est déroulé sur un exemple dans l'annexe . Des expérimentations sur un nombre différent de clients ($nbC = 10, 100$ et 1000), de véhicules ($nbV = 1$ et 5% nbC, 20% nbC et 50% nbC) et pour différentes valeurs de distances ($nbC/8$, $nbC/4$, $nbC/2$, $nbC - 1$ et nbC) ont permis de valider l'algorithme en montrant que la distance calculée entre deux solutions était toujours égale à la distance générée.

2.2.2 Analyse de paysage

2.2.2.1 Design expérimental

Jeu de données : Nous avons utilisé le jeu de données de problème asymétrique de tournées de véhicule de Fischetti *et al.* [FTV94] décrit dans le chapitre 1 où nous avons ajouté un coût unitaire différent à chaque véhicule de la flotte pour la rendre hétérogène.

Opérateurs de voisinage : La distance entre deux solutions de l'espace de recherche est difficile à calculer. Il est pourtant nécessaire de savoir la calculer pour mener une étude du paysage tel que nous l'envisageons. Dans les préliminaires de cette section, nous avons présenté une mesure de distance pour l'opérateur *insertion* et une mesure de distance pour l'opérateur *échange*. Nous obtenons deux paysages : $\mathcal{P}_{IN}$ pour le paysage défini avec la relation de voisinage basée sur l'opérateur *insertion* (IN) et $\mathcal{P}_{EX}$ pour celui de l'opérateur *échange* (EX). Ces deux opérateurs étant couramment utilisés, nous pensons que l'étude de ces paysages peut apporter une compréhension plus fine de leur impact sur les performances d'une métaheuristique. Dans la suite, *RPop* désigne la population aléatoire, $OPop_{IN}$ et $OPop_{EX}$ désigne respectivement les populations d'optima locaux obtenues pour le paysage $\mathcal{P}_{IN}$ et pour le paysage $\mathcal{P}_{EX}$.

2.2.2.2 Résultats expérimentaux

Pour chacun des paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$, les résultats correspondent aux 30 valeurs calculées puis moyennées.

Distances dans une population : Tout d'abord, rappelons que la distance maximale (DMax) entre deux solutions est le nombre de clients de l'instance, que ce soit pour le paysage $\mathcal{P}_{IN}$ ou le paysage $\mathcal{P}_{EX}$. Cette distance maximale correspond au diamètre théorique de l'espace de recherche. Les distances *insertion-distance* et *échange-distance* ont été calculées entre les solutions de *RPop*, $OPop_{IN}$ et $OPop_{EX}$. Les tableaux 2.1 et 2.2 présentent le minimum, le maximum et les quartiles (la médiane, Q1 et Q3) pour les huit instances du jeu de données des distances moyennes calculées pour les 30 populations de chaque type. La valeur de DMax est précisé pour chaque instance pour pouvoir comparer les différentes valeurs de distances au diamètre théorique.
Au regard des valeurs médianes des distances entre les solutions de la population aléatoire *RPop*, les solutions tendent à être plus proches entre-elles en considérant la relation de voisinage $\mathcal{V}_{IN}$. Les valeurs maximales des distances entre deux solutions atteignent toujours celle du diamètre théorique quand elles sont calculées sur le paysage $\mathcal{P}_{EX}$ contrairement

Tableau 2.1 – Distances entre les solutions du paysage $\mathcal{P}_{IN}$.

		RPop			$OPop_{IN}$		
Instances	DMax	Min	Med_{Q_1,Q_3}	Max	Min	Med_{Q_1,Q_3}	Max
A034-02f	33	21	$28_{28,29}$	33	6	$23_{21,24}$	29
A036-03f	35	22	$30_{28,29}$	35	5	$24_{22,26}$	31
A039-03f	38	25	$33_{32,34}$	37	10	$27_{25,29}$	34
A045-03f	44	30	$38_{37,39}$	43	10	$32_{30,33}$	39
A048-03f	47	32	$41_{40,42}$	46	13	$33_{32,35}$	41
A056-03f	55	40	$48_{47,49}$	53	21	$40_{38,42}$	49
A065-03f	64	47	$56_{55,57}$	61	27	$48_{46,50}$	57
A071-03f	70	53	$61_{60,62}$	67	33	$53_{51,55}$	62

Tableau 2.2 – Distances entre les solutions du paysage $\mathcal{P}_{EX}$.

		RPop			$OPop_{EX}$		
Instances	DMax	Min	Med_{Q_1,Q_3}	Max	Min	Med_{Q_1,Q_3}	Max
A034-02f	33	24	$32_{31,32}$	33	16	$31_{29,32}$	33
A036-03f	35	25	$34_{33,34}$	35	19	$33_{32,34}$	35
A039-03f	38	28	$37_{36,37}$	38	22	$36_{35,37}$	38
A045-03f	44	34	$42_{41,43}$	44	29	$42_{41,43}$	44
A048-03f	47	37	$45_{44,46}$	47	30	$45_{43,46}$	47
A056-03f	55	44	$53_{52,54}$	55	40	$53_{51,54}$	55
A065-03f	64	53	$62_{62,63}$	64	48	$61_{60,63}$	64
A071-03f	70	59	$68_{67,69}$	70	56	$67_{66,68}$	70

au paysage $\mathcal{P}_{IN}$ (exception pour les deux plus petites instances). Les remarques sont identiques pour les populations d'optima locaux $OPop_{IN}$ et $OPop_{EX}$. Ceci permet de conclure que le paysage $\mathcal{P}_{IN}$ a tendance à être plus contracté que le paysage $\mathcal{P}_{EX}$.

Notons que les valeurs minimales des deux populations $OPop_{IN}$ et $OPop_{EX}$ sont toutes strictement supérieures à zéro et indiquent qu'aucun des optima locaux ne sont identiques au sein d'une population.

La médiane, Q1 et Q3 ont des valeurs très proches. Ces valeurs de quartiles montrent que les distances entre les solutions de la population aléatoire sont assez homogènes pour les deux paysages considérés. Par contre, les conclusions sont différentes pour les deux paysages pour les population d'optima locaux. En effet, les optima locaux du paysage $\mathcal{P}_{IN}$ sont plus proches les unes des autres que les solutions de la population aléatoire. Là aussi, nous pouvons parler de contraction : le diamètre du paysage en altitude (les solutions choisies aléatoirement dans l'espace de recherche étant plus probablement de mauvaise qualité) est plus large que celui au niveau des optima locaux. Quant au paysage $\mathcal{P}_{EX}$, les

optima locaux de $OPop_{EX}$ sont en médiane quasiment aussi éloignés les uns des autres que les solutions aléatoires de *RPop*. Le paysage $\mathcal{P}_{EX}$ est donc aussi large en altitude qu'au niveau des optima locaux.

Entropie : Ici nous considérons que l'entropie d'une population mesure la probabilité de n'avoir aucun arc commun entre les solutions. Elle est définie par Grefenstette [Gre87] comme :

$$E = \frac{1}{nc+nv} \sum_{i=1}^{nc+nv} E_i, \text{ avec } E_i = -\frac{1}{\ln(nc+nv)} \sum_{j=1}^{nc+nv} \left(\frac{n_{ij}}{|P|} \ln \left(\frac{n_{ij}}{|P|} \right) \right)$$

où $|P|$ est la taille de la population et n_{ij} est le nombre de fois où l'arc connectant i à j est présent dans la population. Pour les huit instances considérées, l'entropie est d'environ 95% pour la population aléatoire *RPop*, d'environ 44% pour la population d'optima locaux $OPop_{IN}$ et d'environ 54% pour la population d'optima locaux $OPop_{EX}$. Les solutions générées aléatoirement sont réellement différentes, peu d'arcs sont présents au moins dans deux solutions. La population aléatoire est bien diversifiée du point de vue de la représentation des solutions. En revanche, les optima locaux ont plus d'arcs communs, certains d'entre eux doivent favoriser la bonne qualité des solutions. Les optima locaux du paysage $\mathcal{P}_{IN}$ sont plus semblables en terme d'arcs que ceux du paysage $\mathcal{P}_{EX}$.

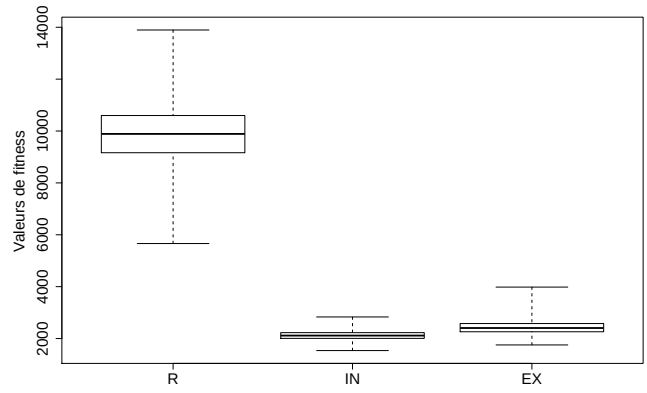


Figure 2.1 – Comparaison des valeurs de fitness des populations aléatoires (R), des populations d'optima locaux obtenus pour le paysage de l'insertion (IN) et pour le paysage de l'échange (EX) pour l'instance A034-02f.

Valeurs de fitness : Premièrement, il est nécessaire de vérifier que les valeurs de fitness de la population des solutions aléatoires sont normalement distribuées. Ceci montre que la population initiale est bien diversifiée. Un test statistique de normalité (ici Kolmogorov-Smirnov) a validé que la distribution des valeurs de fitness est normale. La comparaison des boîtes à moustaches (figure 2.1) de la population *RPop* et des populations d'optima locaux $OPop_{IN}$ et $OPop_{EX}$ montre que les valeurs de fitness des deux populations d'optima

Tableau 2.3 – Valeurs de fitness relatives aux populations des optima locaux $OPop_{IN}$ et $OPop_{EX}$.

	$OPop_{IN}$			$OPop_{EX}$		
Instances	Min	Med	Max	Min	Med	Max
A034-02f	**1604**	2117	2832	1882	2412	3519
A036-03f	**1934**	2421	3082	2190	2928	4312
A039-03f	**2101**	2630	3330	2316	3197	4296
A045-03f	**2244**	2843	3558	2678	3436	5137
A048-03f	**2382**	2998	3776	2848	3550	4954
A056-03f	**2268**	3071	4007	2788	3670	5544
A065-03f	**2835**	3642	4535	3421	4397	5597
A071-03f	**3104**	3785	4996	3593	4513	6077

locaux sont vraiment meilleures que celles de la population de solutions aléatoires. Les paysages $\mathcal{P}_{IN}$ ou $\mathcal{P}_{EX}$ ne sont pas plats. Le tableau 2.3 donne le minimum, la médiane et le maximum de valeurs de fitness pour les deux populations $OPop_{IN}$ et $OPop_{EX}$. Les optima locaux obtenus avec le paysage $\mathcal{P}_{IN}$ sont de meilleure qualité que ceux obtenus avec le paysage $\mathcal{P}_{EX}$. De plus, dans notre cas, les meilleures solutions pour chacunes des instances sont obtenues sur le paysage $\mathcal{P}_{IN}$. Nous pouvons conclure que, partant d'une même solution aléatoire, il est plus facile de trouver une meilleure solution en se déplaçant sur le paysage $\mathcal{P}_{IN}$ que sur le paysage $\mathcal{P}_{EX}$ où la recherche est plus facilement attirée dans les bassins d'attraction des optima locaux de moins bonne qualité.

Tableau 2.4 – Longueurs de marche relatives aux populations des optima locaux $OPop_{IN}$ et $OPop_{EX}$.

	$OPop_{IN}$			$OPop_{EX}$		
Instances	Min	Med	Max	Min	Med	Max
A034-02f	19	29	48	16	27	41
A036-03f	22	33	51	15	28	44
A039-03f	24	35	52	15	31	49
A045-03f	27	40	56	19	35	54
A048-03f	31	44	60	17	38	62
A056-03f	37	50	66	28	44	70
A065-03f	42	57	79	34	51	72
A071-03f	46	64	83	39	57	82

Longueur de marche : La longueur de marche relative correspond au nombre de pas nécessaires le long d'une méthode de descente, ici le Hill Climbing, *i.e.* la taille du chemin partant d'une solution (ici aléatoire) pour arriver à un optimum local. La longueur de

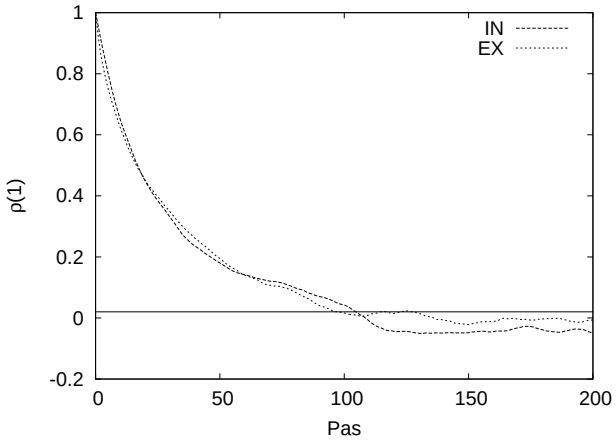


Figure 2.2 – Fonction d'autocorrélation des valeurs de fitness d'une marche aléatoire sur les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ de l'instance A034-02f.

marche permet d'apprécier si la recherche est facilement attirée par un optimum local. Le tableau 2.4 donne le minimum, la médiane et le maximum des longueurs de marche pour les deux populations $OPop_{IN}$ et $OPop_{EX}$. En moyenne, les longueurs de marche sont plus longues pour le paysage $\mathcal{P}_{IN}$. Ce dernier permet donc des descentes plus profondes. Or, la taille du voisinage du paysage $\mathcal{P}_{EX}$ est deux fois plus petite que celle du paysage $\mathcal{P}_{IN}$. La taille du voisinage et les longueurs de marches étant plus petites pour le paysage $\mathcal{P}_{EX}$ implique qu'une recherche locale se déplace plus rapidement sur ce paysage.

Tableau 2.5 – Valeurs d'autocorrélation des paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$.

	$\mathcal{P}_{IN}$			$\mathcal{P}_{EX}$		
Instances	$\rho(1)$	$\rho(2)$	$\rho(3)$	$\rho(1)$	$\rho(2)$	$\rho(3)$
A034-02f	0.96	0.93	0.90	0.93	0.88	0.84
A036-03f	0.96	0.92	0.88	0.93	0.88	0.84
A039-03f	0.95	0.90	0.85	0.93	0.87	0.83
A045-03f	0.96	0.93	0.90	0.94	0.90	0.87
A048-03f	0.97	0.93	0.90	0.94	0.90	0.88
A056-03f	0.97	0.94	0.91	0.96	0.93	0.91
A065-03f	0.98	0.96	0.93	0.96	0.93	0.91
A071-03f	0.98	0.96	0.93	0.96	0.95	0.93

Autocorrélation : Pour étudier la rugosité du paysage, calculons les valeurs de la fonction d'autocorrélation associées aux marches aléatoires sur les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$. La figure 2.2 donne les 200 premières valeurs de l'autocorrélation pour ces deux paysages de l'instance *A034-02f* (34 clients, 10 véhicules). La ligne horizontale est à 2% pour indiquer la limite du bruit dans le calcul du signal. Puisque l'information du paysage local est donné par les premières valeurs d'autocorrélation, le tableau 2.5 donne la valeur des trois

premières pour les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$. Les valeurs d'autocorrélation ne sont pas très différentes entre $\mathcal{P}_{EX}$ et $\mathcal{P}_{IN}$ et nous observons sur la figure que les deux courbes sont très proches et se croisent même pour $\rho(20)$. Les deux paysages semblent donc assez proches en terme de rugosité.

2.2.3 Étude des performances de métaheuristiques classiques

Les résultats de l'analyse des paysages proposés pour ce problème de tournées de véhicules montrent que la relation de voisinage induit des paysages de nature différente. Dans cette section, nous étudions les performances de métaheuristiques classiques pour ce type de problème de tournées de véhicules et nous analysons l'influence du paysage sur ces performances.

2.2.3.1 Design expérimental

Jeu de données : Le même jeu de données de l'analyse de paysage est utilisé.

Voisinages : La section précédente analysait le paysage $\mathcal{P}_{IN}$ défini avec la relation de voisinage basée sur l'opérateur *insertion* (IN) et le paysage $\mathcal{P}_{EX}$ défini avec la relation de voisinage basée sur l'opérateur *échange* (EX). Nous allons donc garder ces deux relations de voisinages pour les intégrer aux métaheuristiques choisies et ainsi expliquer l'influence du paysage sur ces méthodes.

Métaheuristiques : Deux types de recherches locales, l'une itérée, l'autre tabou et un algorithme génétique ont été identifiés comme les méthodes les plus utilisées parmi les métaheuristiques pour ce type de problème de tournées de véhicules.

– Les recherches locales : le but de cette section est d'étudier les performances des métaheuristiques pour montrer ensuite l'influence du paysage sur la dynamique de celles-ci. Comme notre approche d'analyse de paysage utilise le Hill Climbing pour trouver des optima locaux, nous avons choisi de l'embarquer au sein des recherches locales itérée ou tabou pour avoir une dynamique algorithmique identique entre l'étude du paysage et des performances. Par conséquent, nous avons une meilleure lisibilité quant à l'impact du paysage sur leurs résutats. De même, chaque recherche locale est analysée avec les relations de voisinage $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$ respectivement définies à partir des opérateurs d'*insertion* et d'*échange* présentés dans la section précédente.

  La *recherche locale itérée* (RL1) embarque une méthode de perturbation de la solution courante et un critère d'acceptation. Dans notre étude, la perturbation consiste à remplacer la solution courante par une solution choisie aléatoirement dans l'espace de recherche. Le critère d'acceptation garde en mémoire la solution de meilleure qualité rencontrée au cours de la recherche. Cette recherche locale est considérée comme rapide car elle recommence une nouvelle recherche dès qu'un optimum local est atteint.

La *recherche tabou* est constituée d'une *liste tabou* qui est une mémoire à plus ou moins court terme de la recherche. La taille de cette liste est un paramètre qui influence beaucoup les performances. Comme la taille du voisinage d'une solution dépend pour les deux relations de voisinage $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$ du nombre de clients, nous avons testé avec deux tailles différentes de listes dépendant du nombre de clients : RL2 a une *liste tabou* de taille égale à $nc/2$ et RL3 égale à nc ;

– L'algorithme génétique : il existe de nombreux paramètres à définir dans un algorithme génétique. Pour notre étude, nous avons choisi un algorithme génétique (AG) avec une population initiale de 100 individus générés aléatoirement. La sélection est basée sur un tournoi binaire. Le remplacement conserve le meilleur parent dans la population enfant. La mutation correspond au choix aléatoire d'un voisin pour P_m des individus de la population. Le croisement, illustré par la figure 2.3, consiste à choisir une tournée numéro i aléatoirement parmi celles d'une première solution parent sélectionnée (P1) pour être réinsérée à la place de la tournée i d'une seconde solution parent (P2). Comme l'individu P2 contient déjà les clients de la tournée sélectionnée, ils sont alors tous effacés de cette dernière. Ainsi, il n'y a pas de redondance de clients dans la solution P2. Si la tournée i de P2 était initialement vide, alors l'étape de croisement est terminée et l'enfant est créé. Sinon, les tournées $i...nv$ de P2 sont déplacées vers la droite pour rendre possible l'insertion de la nouvelle tournée en i. Dans le cas où P2 possède alors trop de tournées *i.e.* $nv + 1$, chaque client de la dernière tournée est inséré à la fin des autres tournées. Commençons bien sûr par insérer les clients dans la tournée ayant le coût le plus faible tout en respectant les contraintes du problème (pour nous, la capacité des véhicules). L'enfant est créé une fois que tous les clients sont réinsérés. Le croisement est appliqué entre les parents P1 et P2 de la population et vice versa, avec une probabilité P_c. Pour garder de la diversité dans la population enfant, P_m et P_c ont été choisi arbitrairement égaux à 0.7.

Tableau 2.6 – Description des instances et nombre maximal d'évaluations pour chacune d'entre elles. Pour plus de lisibilité, une lettre est associée à chaque instance.

Instances	nc	nv	nbEval
A034-02f (A)	33	10	500 000
A036-03f (B)	35	10	550 000
A039-03f (C)	38	10	600 000
A045-03f (D)	44	10	1 000 000
A048-03f (E)	47	10	2 000 000
A056-03f (F)	55	10	180 000
A065-03f (G)	64	10	2 700 000
A071-03f (H)	70	10	3 500 000

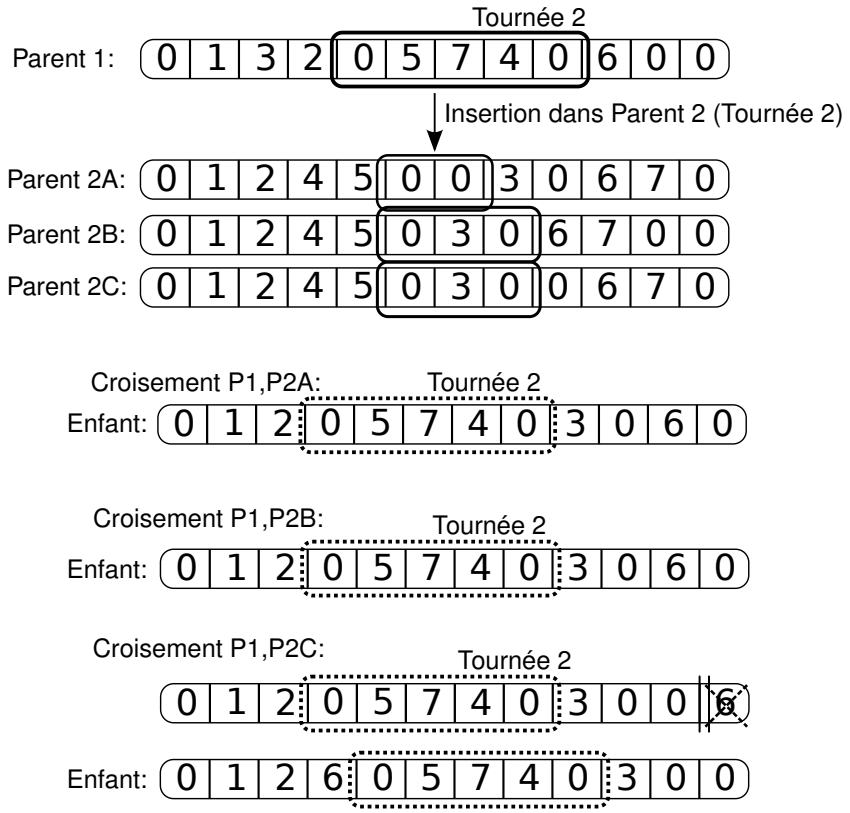


Figure 2.3 – Le croisement est effectué entre un Parent1 et un Parent2. La tournée 2 est choisie dans le Parent1 pour être insérée à la même position dans le Parent 2 après avoir enlevé les clients redondants. Trois cas doivent être pris en considération et sont illustrés par Parents 2A, 2B et 2C. Dans le Parent2A, la tournée 2 est vide donc l'insertion de la nouvelle tournée est évidente. Dans le Parent2B, la dernière tournée étant vide, il suffit de décaler vers la droite les tournées numérotées à partir de 2 pour insérer la nouvelle. Dans le Parent2C, le décalage provoque une tournée additionnelle, donc les clients sont réinsérés en respectant les contraintes du problème dans les tournées précédentes en commençant par celles ayant le plus faible coût.

Mise en œuvre : La plateforme logicielle ParadisEO a été utilisée pour implémenter toutes les méthodes testées [CMT04]. ParadisEO est dédiée à la conception flexible des métaheuristiques ; l'implémentation des composants communs aux différentes méthodes choisies et comparées est donc partagée (initialisation, voisinage, *etc.*).

Les métaheuristiques étant des méthodes stochastiques, chacune a été exécutée 30 fois pour chacune des instances du jeu de données. Pour chaque instance, le critère d'arrêt est le même. Une étude préliminaire sur la convergence des métaheuristiques analysées a permis de définir sa valeur pour chacune des instances (tableau 2.6).

2.2.3.2 Résultats expérimentaux

Tableau 2.7 – Opérateur *insertion* : meilleure, moyenne et écart-type des performances des 30 solutions trouvées par chaque méthode pour les 8 instances. Pour chaque instance, la meilleure performance trouvée est écrite en caractères gras.

	Instances							
	A	B	C	D	E	F	G	H
RL1								
Meilleure	1740.4	1964.2	**2000.2**	2294	2399.4	2429.8	2879.6	3131.6
Moyenne	1873.3	2136.87	2323.59	2524.15	2663.63	2695.54	3210.37	3354
Écart-type	65.06	79.31	114.30	97.48	93.41	98.66	122.03	143.85
RL2								
Meilleure	1755.6	1921.8	2183	2209.8	2331.2	2305.2	2620.2	2758.6
Moyenne	1991.65	2258.37	2397.17	2511.61	2685.52	2591.27	3128.15	3188.2
Écart-type	135.95	118.09	156.71	167.03	222.91	163.71	262.67	230.59
RL3								
Meilleure	**1697.2**	**1921.8**	2103	**2018.6**	2308.8	2269.4	2634.2	2758.6
Moyenne	1950.22	2208.69	2344.49	2430.23	2631.55	2545.47	3023.35	3117.32
Écart-type	120.64	127.03	165.2	184.96	162.93	160.16	242.36	216.18
AG								
Meilleure	1735.6	1951.2	2147.8	2126.4	**2308**	**2250.6**	**2541.2**	**2715**
Moyenne	1926.95	2183.26	2376.55	2458.95	2648.35	2501.55	2957.34	3059.01
Écart-type	106.75	128.46	137.49	185.93	141.54	170.76	168.72	177.5

Les quatre méthodes RL1, RL2, RL3 et AG ont été exécutées en utilisant premièrement l'opérateur *insertion*, puis deuxièmement l'opérateur *échange*. Pour chaque méthode et sur chaque instance, les tableaux 2.7 et 2.8 présentent la meilleure, la moyenne et l'écart-type des performances des 30 exécutions respectivement pour les opérateurs d'*insertion* et d'*échange*.

Pour apprécier la qualité des résultats, des tests statistiques ont été effectués entre les quatre méthodes pour chacun des deux voisinages. Les performances d'une méthode est caractérisée par une population de 30 valeurs de fitness des solutions qu'elle a trouvées. La normalité de la distribution de cette population est validée par le test de Kolmogorov-Smirnov. Dans ce cas, le test de Levene assure l'égalité des variances entre deux populations

Tableau 2.8 – Opérateur *échange* : meilleure, moyenne et écart-type des performances des 30 solutions trouvées par chaque méthode pour les 8 instances. Pour chaque instance, la meilleure performance trouvée est écrite en caractères gras.

	Instances							
	A	B	C	D	E	F	G	H
RL1								
Meilleure	1932.8	2306.6	2481	2585.4	2886.4	2985	3356.4	3592.4
Moyenne	2045.56	2460.65	2670.26	2860.97	3042.27	3175.1	3785.03	3901.11
Écart-type	5069.96	8217.12	11110.6	10429.5	7534.2	12957.6	27774.5	29561.4
RL2								
Meilleure	1676.4	1959.6	2179.2	2221.8	2417.2	**2231.4**	2707	2944.6
Moyenne	1933.03	2301.76	2537.04	2522.09	2684.45	2712.96	3156.57	3279.93
Écart-type	19619	30519.9	74883	25861.9	31090.8	81690.9	66678.6	27427.6
RL3								
Meilleure	**1629**	**1919.6**	**2141**	**2132.4**	**2318.6**	2268	**2498.4**	**2791.2**
Moyenne	1834.35	2159.43	2374.09	2419.19	2571.52	2606.52	2995.47	3083.62
Écart-type	16337.7	16114.5	41443.5	19866.8	20311.6	52933.7	53627.3	41305.8
AG								
Meilleure	1943.8	2245.4	2365.6	2570.2	2878.2	2685	3332.6	3412.4
Moyenne	2229.36	2611.51	2770.6	3033.78	3205.05	3191.04	3701.08	3875.36
Écart-type	24144.9	27747	36322.3	75451.8	39993.6	56581.1	39375.1	72171.5

à comparer. Puis, le test de Student teste l'hypothèse nulle (H0) que les moyennes des deux distributions normales sont égales. Dans le cas, où la distribution des valeurs de fitness pour une méthode n'est pas normale, le test des rangs non-paramétrique apparié de Wilcoxon est utilisé pour tester l'hypothèse nulle (H0) qui évalue si les moyennes des populations sont différentes ou non. Les méthodes sont comparées deux à deux. Chaque test statistique renvoit une p-valeur comme résultat du test de l'hypothèse nulle (H0). Dans cette étude, l'intervalle de confiance est à 95%. Si cette p-valeur est plus grande que 0.05, les deux méthodes sont considérées comme équivalentes pour l'expérience menée. Sinon, les performances sont significativement différentes et la meilleure méthode de l'expérience menée est celle qui a la meilleure moyenne. Le tableau 2.9 présente les résultats de ces tests statistiques de comparaison des valeurs de fitness de deux populations.

D'après le tableau 2.8, considérant la relation de voisinage $\mathcal{V}_{EX}$, RL3 donne la solution de meilleure qualité parmi l'ensemble des méthodes testées pour 7 instances (RL2 trouve la meilleure de l'instance F) et la moyenne des valeurs de fitness renvoyées est toujours la meilleure. La méthode RL3 semble dominer les trois autres méthodes. Les tests statistiques confirment la dominance de RL3 pour la relation de voisinage $\mathcal{V}_{EX}$ en terme de qualité des solutions trouvées pour nos exécutions (tableau 2.9). De plus, les tests statistiques montrent que l'algorithme génétique (AG) est dans tous les cas dominés par les trois méthodes de recherches locales ou, est équivalent à la méthode RL1 pour 3 instances sur 8 (F à H).

Considérant la relation de voisinage $\mathcal{V}_{IN}$, les résultats sont plus difficile à décrire. Pour faciliter l'analyse, les instances sont classés en deux types : les plus petites (A à E) et les

Tableau 2.9 – Comparaison statistique des performances des méthodes RL1, RL2, RL3 et AG sur les huit instances. $>$ signifie que la méthode de gauche domine celle du dessus, $<$ le contraire et $=$ que les méthodes sont statistiquement équivalentes.

		INSERTION			ÉCHANGE		
Instances		RL2	RL3	AG	RL2	RL3	AG
A034-02f	RL1	$>$	$>$	$=$	$<$	$<$	$>$
(A)	RL2		$<$	$<$		$<$	$>$
	RL3			$=$			$>$
A036-03f	RL1	$>$	$>$	$=$	$<$	$<$	$>$
(B)	RL2		$<$	$<$		$<$	$>$
	RL3			$=$			$>$
A039-03f	RL1	$>$	$=$	$=$	$<$	$<$	$>$
(C)	RL2		$<$	$=$		$<$	$>$
	RL3			$=$			$>$
A045-03f	RL1	$=$	$<$	$=$	$<$	$<$	$>$
(D)	RL2		$<$	$=$		$<$	$>$
	RL3			$=$			$>$
A048-03f	RL1	$=$	$=$	$=$	$<$	$<$	$>$
(E)	RL2		$<$	$=$		$<$	$>$
	RL3			$=$			$>$
A056-03f	RL1	$<$	$<$	$<$	$<$	$<$	$=$
(F)	RL2		$<$	$<$		$<$	$>$
	RL3			$=$			$>$
A065-03f	RL1	$=$	$<$	$<$	$<$	$<$	$=$
(G)	RL2		$<$	$<$		$<$	$>$
	RL3			$=$			$>$
A071-03f	RL1	$<$	$<$	$<$	$<$	$<$	$=$
(H)	RL2		$<$	$=$		$<$	$>$
	RL3			$=$			$>$

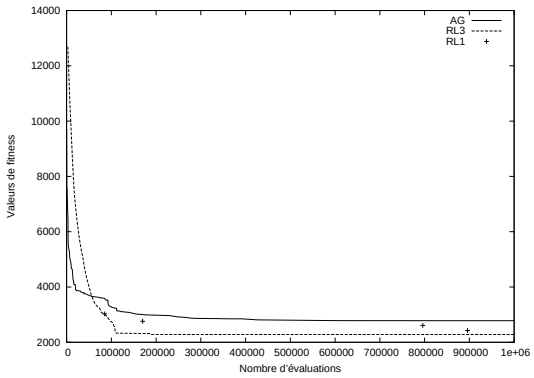


Figure 2.4 – Valeurs de fitness en fonction du nombre d'évaluations. Illustre la convergence de RL1, RL3 et GA sur l'instance A045-03f ($nc = 44$, $nv = 10$) avec la relation de voisinage $\mathcal{V}_{IN}$. RL2 n'est pas représentée car sa convergence est identique à celle de RL3.

plus grandes (F à H). D'après le tableau 2.9, dans le cas des plus petites instances, RL3 n'est jamais dominé par aucun autre algorithme. Il est tout de même statistiquement équivalent à RL1 ou AG pour certaines petites instances. AG est statistiquement équivalent aux trois recherches locales, excepté pour les instances (A) et (B) où ses performances sont meilleures que RL2. RL1 et RL3 montrent tout de même de meilleures performances que RL2 pour les instances (C) et (D). De manière générale, ces comparaisons statistiques affirment la prédominance des recherches locales rapides (RL1) ou avec une grande mémoire (RL3) pour les plus petites instances.
Pour les plus grandes instances, AG et RL3 ont des performances non différentiables statistiquement et sont alors considérées comme équivalentes pour le paysage $\mathcal{P}_{IN}$. Ces deux méthodes obtiennent les meilleures performances sur ces instances.
AG semble devenir intéressant quand l'espace de recherche grossit : l'exploration de plusieurs zones amènent à de bonnes solutions.

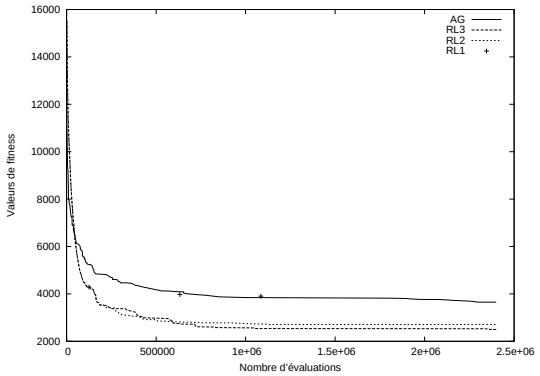


Figure 2.5 – Valeurs de fitness en fonction du nombre d'évaluations. Illustre la convergence de RL1, RL2, RL3 et GA sur l'instance A065-03f ($nc = 64$, $nv = 10$) pour une exécution pour la relation de voisinage $\mathcal{V}_{EX}$.

Ces remarques nous amènent donc à conclure qu'indéniablement en considérant la rela-

Tableau 2.10 – Éfficacité des méthodes selon l'opérateur de voisinage utilisé.

Instances	Comparaison
A-H	$RL1_{IN} > RL1_{EX}$
A,B,D,E,G,H	$RL2_{IN} = RL2_{EX}$
C,F	$RL2_{IN} > RL2_{EX}$
A	$RL3_{IN} < RL3_{EX}$
B,C,D,E,F,G,H	$RL3_{IN} = RL3_{EX}$
A-H	$AG_{IN} > AG_{EX}$

tion de voisinage $\mathcal{V}_{EX}$ basée sur l'opérateur *échange*, le meilleur choix pour résoudre notre problème de tournées de véhicules est RL3, une recherche tabou ayant une *liste tabou* de grande taille (ici, le nombre de clients). Par contre, en considèrant la relation de voisinage $\mathcal{V}_{IN}$ basée sur l'opérateur *insertion*, les recherches locales semblent être plus efficaces que l'algorithme génétique, sur les petites instances étudiées avec un nombre maximal d'évaluations raisonnable. Quant aux autres instances, aucun algorithme n'a des performances vraiment meilleures que les autres.

La convergence d'une méthode peut être étudiée en regardant l'évolution de la valeur de fitness en fonction du nombre d'évaluations. Cette évaluation informe sur la rapidité d'une méthode à trouver une bonne solution ou sur son incapacité à améliorer la qualité à partir d'un certain nombre d'évaluations. Les figures 2.4 et 2.5 illustrent la convergence des méthodes analysées respectivement pour les relations de voisinage $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$. Ces convergences soutiennent les conclusions précédentes. En effet, la relation de voisinage $\mathcal{V}_{EX}$ avantage réellement RL3 alors que les méthodes ont un comportement statistiquement identique avec la relation de voisinage $\mathcal{V}_{IN}$.

Le tableau 2.10 présente les résultats des tests statistiques des comparaisons deux à deux des méthodes avec pour l'une l'opérateur *insertion* et pour l'autre l'opérateur *échange*. Ces résultats montrent que RL1 et AG ont des performances bien meilleures en utilisant l'opérateur *insertion* plutôt que l'opérateur *échange* ; par contre, les recherches tabou RL2 et RL3 ont des performances similaires en utilisant l'un ou l'autre des opérateurs.

2.2.4 Discussion

Ce chapitre est consacré à l'influence du paysage sur les performances des métaheuristiques. Dans cette première section, nous avons commencé par analyser deux paysages d'un problème asymétrique de tournées de véhicules avec flotte fixe hétérogène (HFF-AVRP), différenciés par leur relation de voisinage. Les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ sont respectivement définis par les relations de voisinage $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$, elle-mêmes définies par l'application des

opérateurs de voisinage d'*insertion* (IN) et d'*échange* (EX). Dans un deuxième temps, intégrant ces opérateurs dans trois recherches locales et un algorithme génétique, nous avons comparé leurs performances sur le même jeu de données que celui utilisé pour l'analyse des paysages. Cette section a pour objectif de faire le lien entre ces deux études : comment en s'aidant de l'analyse des paysages aurions-nous pu prévoir les résultats des performances des différentes métaheuristiques ?

Pour synthétiser les résultats, nous avons testé statistiquement si la différence des performances de chaque métaheuristique sur chacun des paysages induits est significative pour chacune des instances du jeu de données. Ces résultats sont présentés dans le tableau 2.10. Premièrement, les performances obtenues par les méthodes implémentées montrent que celles des recherches tabou (RL2 et RL3) sont similaires pour les deux opérateurs de voisinages considérés. En effet, ces méthodes ont pour but de visiter le voisinage en utilisant une large exploration autorisée par la *liste tabou*. Elles semblent ne pas être très sensibles aux paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ induits.

Rappelons que les longueurs de marche du Hill Climbing sur le paysage $\mathcal{P}_{IN}$ sont plus longues. De même, les valeurs de fitness des optima locaux sont meilleures. Ces deux remarques impliquent que le paysage $\mathcal{P}_{IN}$ a des vallées plus profondes que le paysage $\mathcal{P}_{EX}$. Le parcours d'un Hill Climbing semble donc être favorisé sur le paysage $\mathcal{P}_{IN}$ par rapport à $\mathcal{P}_{EX}$. La recherche locale itérée RL1 correspond à un Hill Climbing qui est redémarré d'une solution choisie aléatoirement dans l'espace de recherche. RL1 réussit à atteindre des optima locaux de meilleures qualités en se déplaçant sur $\mathcal{P}_{IN}$ que sur $\mathcal{P}_{EX}$. Ces performances confirment les conclusions précédentes de l'étude des vallées des paysages quant à la faculté de trouver des solutions de meilleure qualité en considérant le problème sous le paysage $\mathcal{P}_{IN}$.

Les valeurs d'autocorrélation sont assez identiques entre les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ et montrent que les deux paysages possèdent une forte rugosité. Localement, le paysage $\mathcal{P}_{IN}$ est un peu plus lisse que celui de $\mathcal{P}_{EX}$. Les distances entre les solutions des populations aléatoire ou d'optima locaux indiquent que $\mathcal{P}_{IN}$ est plus contracté et que les optima locaux ne sont pas aussi éloignés les uns des autres que des solutions choisies aléatoirement dans l'espace de recherche. La mutation de l'algorithme génétique AG correspond à l'application de l'opérateur de voisinage *insertion* ou *échange* ce qui induit localement respectivement un paysage $\mathcal{P}_{IN}$ ou $\mathcal{P}_{EX}$. Les conclusions précédentes sur ces paysages tendent à favoriser l'application de l'opérateur *insertion* car localement la qualité de la solution mutée a plus de chance d'être meilleure. Les optima locaux étant plus proches les uns des autres, la population enfant a plus de chance d'évoluer vers des solutions de meilleure qualité.

L'analyse de paysage telle qu'elle a été menée sur ce problème de tournées de véhicules permet d'expliquer les performances de certaines métaheuristiques. Les indicateurs utilisés ont montré certains effets du paysage sur le parcours d'une recherche locale ou d'un algorithme génétique. Néanmoins, le problème asymétrique de tournées de véhicules avec flotte fixe hétérogène est très difficile et les opérateurs choisis pour notre étude sont des opérateurs classiques de la littérature. Pour résoudre ce type de problème, les méthodes

efficaces utilisent généralement plusieurs opérateurs de voisinage en même temps ce qui rend difficile la définition d'une structure de paysage. De plus, les indicateurs de l'espace de recherche étant basés pour la plupart sur la distance induite par la relation de voisinage, cette étude est difficilement transposable à d'autres voisinages plus complexes. Aussi nous pensons que cette analyse est un travail préliminaire d'étude d'un problème de tournées de véhicules sous l'angle de la structure de paysage.

Afin de montrer l'impact de la structure de paysage sur les performances des métaheuristiques sur un problème plus classique de la littérature, nous réalisons le même type d'étude pour le problème de flowshop.

2.3 Étude du problème de flowshop

Dans le chapitre 1, nous avons présenté le problème de flowshop de permutation que nous étudions ici. Deux opérateurs de voisinages, l'opérateur *insertion* et l'opérateur *échange*, ont été décrits pour ce problème. Nous commençons par analyser deux paysages associés à ce problème, puis nous comparons les performances de métaheuristiques utilisées classiquement pour ce type de problème, sur les paysages précédents. Enfin nous analysons l'influence du paysage sur ces performances.

2.3.1 Analyse de paysage

2.3.1.1 Design expérimental

Jeu de données : Nous avons choisi les instances de Taillard [Tai93] décrites dans le chapitre 1. Ici, seules les instances ayant 20, 50 ou 100 tâches à ordonnancer sur 5, 10 ou 20 machines sont utilisées. Seule la première instance de chaque taille est considérée. Le nom de l'instance est donné par $N/M/i$ où N est le nombre de tâches à ordonnancer, M est le nombre de machines et i est le numéro de l'instance (ici $i = 1$)

Opérateurs de voisinage : Rappelons qu'une solution est représentée à l'aide d'une permutation. Dans la littérature [dBS01, MIT96], les deux opérateurs de voisinage les plus utilisés sont l'opérateur *insertion* (IN) et l'opérateur *échange* (EX). Pour les problèmes de permutation, Schiavinotto *et al.* [SS07] décrivent des algorithmes pour calculer les distances associées à ces deux opérateurs respectivement *insertion*-distance et *échange*-distance dans le cas de solutions représentées par des permutations. Considérant l'un ou l'autre de ces opérateurs, la distance maximale entre deux solutions est égale à la taille de la permutation moins un. Nous obtenons donc deux paysages : $\mathcal{P}_{IN}$ pour le paysage définit avec la relation de voisinage basée sur l'opérateur *insertion* et $\mathcal{P}_{EX}$ pour celui de l'opérateur *échange*. Ces deux opérateurs étant couramment utilisés, nous pensons que l'étude de ces paysages peuvent apporter une compréhension plus fine de son impact sur les performances d'une métaheuristique. Dans la suite, $RPop$ désigne la population aléatoire, $OPop_{IN}$ la population d'optima locaux obtenue pour le paysage $\mathcal{P}_{IN}$ et $OPop_{EX}$ obtenue pour le paysage $\mathcal{P}_{EX}$.

Tableau 2.11 – Distances entre les solutions du paysage $\mathcal{P}_{IN}$.

		RPop			$OPop_{IN}$		
Instances	DMax	Min	Med_{Q_1,Q_3}	Max	Min	Med_{Q_1,Q_3}	Max
20/ 5/01	19	9	$13_{13,14}$	16	7	$12_{11,12}$	15
20/10/01	19	9	$13_{13,14}$	16	3	$10_{9,11}$	14
20/20/01	19	9	$13_{13,14}$	16	4	$12_{10,13}$	15
50/ 5/01	49	33	$39_{38,40}$	42	31	$37_{36,38}$	41
50/10/01	49	34	$39_{38,40}$	42	29	$35_{34,37}$	40
50/20/01	49	34	$39_{38,40}$	42	28	$34_{33,36}$	39
100/ 5/01	99	77	$83_{82,84}$	88	74	$81_{80,82}$	86
100/10/01	99	77	$83_{82,84}$	88	71	$78_{77,80}$	84
100/20/01	99	77	$83_{82,84}$	88	72	$79_{78,81}$	85

2.3.1.2 Résultats expérimentaux

Cette section présente les résultats de l'analyse des paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$. Les valeurs présentées des indicateurs sont les moyennes de celles calculées pour chacune des 30 exécutions.

Distances dans une population : Dans le cas du flowshop de permutation, rappelons que la distance maximale (DMax) entre deux solutions est le nombre de tâches moins un à ordonnancer que ce soit pour le paysage $\mathcal{P}_{IN}$ ou le paysage $\mathcal{P}_{EX}$. Cette distance maximale correspond au diamètre théorique de l'espace de recherche. Les distances *insertion-distance* et *échange-distance* ont été calculées entre les solutions de *RPop*, $OPop_{IN}$ et $OPop_{EX}$. Les tableaux 2.11 et 2.12 présentent le minimum, le maximum et les quartiles (la médiane, Q1 et Q3), pour les neuf instances du jeu de données, des distances moyennes calculées pour les 30 populations de chaque type. La valeur de DMax est précisé pour chaque instance pour pouvoir comparer les différentes valeurs de distances au diamètre théorique.
Le diamètre de l'espace de recherche correspond à la plus grande distance existante entre les solutions de l'espace de recherche. La population *RPop* est composée de solutions choisies aléatoirement dans l'espace de recherche. C'est donc un bon témoin de la place qu'occupe les solution dans l'espace de recherche en général. Dans cette étude, nous observons que la valeur théorique du diamètre est atteinte uniquement pour le paysage $\mathcal{P}_{EX}$ et ce que ce soit pour la population aléatoire ou la population des optima locaux. Par contre, le diamètre du paysage $\mathcal{P}_{IN}$ est inférieur à DMax. Les solutions sont donc moins dispersées sur $\mathcal{P}_{IN}$ que sur $\mathcal{P}_{EX}$. Le paysage $\mathcal{P}_{IN}$ a une tendance à être plus contracté que le paysage $\mathcal{P}_{EX}$.
De plus, la médiane, Q1 et Q3 ont des valeurs très proches ce qui montre que les distances entre les solutions de la population aléatoire sont assez homogènes pour les deux paysages analysés. Par contre, les conclusions sont différentes pour les deux paysages selon les résultats pour les populations d'optima locaux. Les solutions de $OPop_{EX}$ semblent dispersées de la même manière sur l'espace de recherche que les solutions de *RPop*. Cette remarque

Tableau 2.12 – Distances entre les solutions du paysage $\mathcal{P}_{EX}$.

		RPop			$OPop_{EX}$		
Instances	DMax	Min	Med_{Q_1,Q_3}	Max	Min	Med_{Q_1,Q_3}	Max
20/ 5/01	19	11	$17_{16,17}$	19	10	$16_{15,17}$	19
20/10/01	19	11	$17_{16,17}$	19	7	$15_{14,16}$	19
20/20/01	19	11	$16_{16,17}$	19	8	$16_{14,17}$	19
50/ 5/01	49	39	$46_{44,47}$	49	38	$45_{44,46}$	49
50/10/01	49	39	$46_{44,47}$	49	37	$45_{43,46}$	49
50/20/01	49	39	$46_{44,47}$	49	36	$44_{42,45}$	49
100/ 5/01	99	88	$95_{94,96}$	99	87	$95_{93,96}$	99
100/10/01	99	88	$95_{94,96}$	99	86	$94_{92,95}$	99
100/20/01	99	88	$95_{94,96}$	99	86	$94_{92,95}$	99

n'est pas tout à fait applicable au cas des solutions sur le paysage $\mathcal{P}_{IN}$, mais il est facile de constater que les médianes et les intervalles donnés par les quartiles sont quand même assez proches. Le paysage $\mathcal{P}_{IN}$ est un peu plus contracté au niveau des optima locaux qu'il ne l'est en altitude, au niveau des solutions choisies aléatoirement *RPop* (qui sont généralement de mauvaise qualité). Le paysage $\mathcal{P}_{EX}$ est aussi large en altitude qu'au niveau des optima locaux.

Entropie : Dans le cas du flowshop de permutation, nous considérons que l'entropie d'une population mesure la probabilité de n'avoir aucune succession de tâches communes entre les solutions. Elle est définie comme :

$$E = \frac{1}{N}\sum_{i=1}^{N} -\frac{1}{\ln N}\sum_{j=1}^{N}\left(\frac{n_{ij}}{|P|}\ln\left(\frac{n_{ij}}{|P|}\right)\right)$$

où $|P|$ est la taille de la population et n_{ij} est le nombre de fois où la tâche i est suivie de la tâche j dans la population. Pour les instances considérées, l'entropie est la même quelque soit la taille du problème, ici le nombre de tâches à ordonnancer. L'entropie est d'environ 61% pour la population aléatoire *RPop*, d'environ 56% pour la population d'optima locaux $OPop_{IN}$ et d'environ 57% pour la population d'optima locaux $OPop_{EX}$. Les solutions générées aléatoirement ont quand même des similarités assez visibles quant à l'ordonnancement des différentes tâches sur les machines. Que ce soit les optima du paysage $\mathcal{P}_{IN}$ ou ceux du paysage $\mathcal{P}_{EX}$, ils ne se ressemblent pas plus et sont aussi diversifiés que les populations aléatoires du point de vue de l'ordonnancement des tâches.

Valeurs de fitness : Premièrement, il est nécessaire de vérifier dans une étude de paysage que les valeurs de fitness de la population des solutions aléatoires sont normalement distribuées ce qui atteste que la population initiale est bien diversifiée. La figure 2.6 donnent les boîtes à moutaches des trois populations considérées (*RPop*, $OPop_{IN}$ et $OPop_{EX}$) pour l'instance 20/10/01. L'allure des boîtes à moustaches étant la même pour

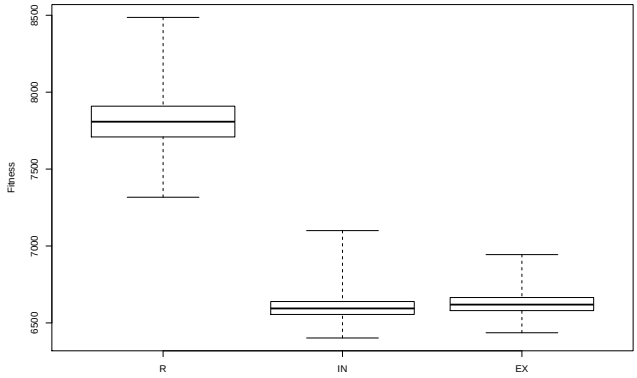


Figure 2.6 – Comparaison des valeurs de fitness pour l'instance 20/10/01.

Tableau 2.13 – Valeurs de fitness relatives aux populations des optima locaux $OPop_{IN}$ et $OPop_{EX}$.

	$OPop_{IN}$			$OPop_{EX}$		
Instances	Min	Med	Max	Min	Med	Max
20/ 5/01	**1279**	1297	1381	1283	1324	1390
20/10/01	**1585**	1631	1758	1595	1662	1806
20/20/01	**2305**	2391	2515	2319	2408	2528
50/ 5/01	**2724**	2752	2822	2724	2752	2856
50/10/01	**3059**	3148	3312	3076	3180	3339
50/20/01	**3952**	4044	4209	3979	4082	4288
100/ 5/01	**5493**	5510	5642	5494	5527	5666
100/10/01	**5802**	5883	6102	5822	5964	6196
100/20/01	**6436**	6594	6886	6464	6620	6891

les autres instances étudiées, nous ne les représentons pas ici. Les valeurs de fitness des deux populations d'optima locaux sont généralement meilleures que celles de la population de solutions aléatoires. Certains optima locaux des paysages $\mathcal{P}_{IN}$ ou $\mathcal{P}_{EX}$ ayant des valeurs de fitness moins bonnes que certaines solutions de $RPop$. Les paysages $\mathcal{P}_{IN}$ ou $\mathcal{P}_{EX}$ ne sont pas plats.

Le tableau 2.13 donne le minimum, la médiane et le maximum de valeurs de fitness pour les deux populations $OPop_{IN}$ et $OPop_{EX}$. Les optima locaux obtenus avec le paysage $\mathcal{P}_{IN}$ sont de meilleure qualité que ceux obtenus avec le paysage $\mathcal{P}_{EX}$. De plus, dans notre cas, les meilleures solutions pour chacunes des instances sont obtenues sur le paysage $\mathcal{P}_{IN}$. Par conséquent, partant d'une même solution aléatoire, il est plus probable de trouver une meilleure solution en se déplaçant sur le paysage $\mathcal{P}_{IN}$ que sur le paysage $\mathcal{P}_{EX}$ où les solutions sont plus facilement attrapées par des optima locaux de moins bonne qualité.

Tableau 2.14 – Longueurs de marche relatives aux populations des optima locaux $OPop_{IN}$ et $OPop_{EX}$.

	$OPop_{IN}$			$OPop_{EX}$		
Instances	Min	Med	Max	Min	Med	Max
20/ 5/01	2	7	15	2	6	14
20/10/01	5	13	25	3	11	25
20/20/01	4	11	23	3	10	21
50/ 5/01	4	11	20	3	9	18
50/10/01	9	23	43	7	20	40
50/20/01	12	27	47	10	23	43
100/ 5/01	6	16	30	4	12	25
100/10/01	14	32	55	8	23	48
100/20/01	18	42	73	14	36	67

Longueur de marche : La longueur de marche relative à une population d'optima locaux permet d'apprécier si une méthode de descente sera facilement attrapée par un optimum local ou pas. En effet, considérant dans cette étude de paysage la méthode de descente la plus classique, elle en est un bon indicateur étant donné que la longueur de marche est le nombre de pas nécessaires en partant d'une solution (ici aléatoire) pour être attrapé par un optimum local. Le tableau 2.14 donne le minimum, la médiane et le maximum des longueurs de marches pour les deux populations $OPop_{IN}$ et $OPop_{EX}$. En moyenne, la descente est plus profonde sur le paysage $\mathcal{P}_{IN}$. Par contre, la taille du voisinage et les longueurs de marche sont plus petites pour le paysage $\mathcal{P}_{EX}$ ce qui implique qu'une recherche locale sera bien plus rapide. Ces deux remarques incitent donc à étudier le compromis entre durée et qualité. Constatons aussi que pour les deux paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$, les longueurs de marche minimum sont en moyenne assez petites ce qui signifie que des optima locaux peuvent être très rapidement atteints. Ces longueurs expliquent probablement les mauvaises valeurs de fitness de certains optima locaux.

Autocorrélation : Pour étudier la rugosité du paysage, les valeurs de la fonction d'autocorrélation de deux marches aléatoires sur les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ sont calculées. La figure 2.7 donne les 250 premières valeurs d'autocorrélation pour ces deux paysages. La ligne horizontale est à 2% pour indiquer la limite du bruit dans le calcul du signal d'autocorrélation. Puisque l'information du paysage local est donnée par les premières valeurs d'autocorrélation, le tableau 2.5 donne la valeur des trois premières des paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$. Les valeurs d'autocorrélation sont assez différentes entre $\mathcal{P}_{EX}$ et $\mathcal{P}_{IN}$. Les valeurs étant plus petites pour $\mathcal{P}_{EX}$, nous en déduisons qu'il doit être plus rugueux localement. Les valeurs décroissent aussi plus vite, ce paysage n'est donc pas régulier. $\mathcal{P}_{IN}$ a des valeurs d'autocorrélation assez élevées ce qui est intéressant pour la dynamique des recherches locales.

Tableau 2.15 – Valeurs d'autocorrélation des paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$.

	$\mathcal{P}_{IN}$			$\mathcal{P}_{EX}$		
Instances	$\rho(1)$	$\rho(2)$	$\rho(3)$	$\rho(1)$	$\rho(2)$	$\rho(3)$
20/ 5/01	0.88	0.79	0.71	0.83	0.69	0.59
20/10/01	0.87	0.77	0.69	0.83	0.69	0.58
20/20/01	0.88	0.79	0.71	0.83	0.70	0.60
50/ 5/01	0.96	0.93	0.89	0.92	0.86	0.80
50/10/01	0.95	0.90	0.86	0.91	0.84	0.78
50/20/01	0.94	0.89	0.84	0.91	0.84	0.77
100/ 5/01	0.98	0.95	0.93	0.96	0.93	0.90
100/10/01	0.98	0.95	0.93	0.96	0.92	0.88
100/20/01	0.97	0.95	0.93	0.95	0.91	0.88

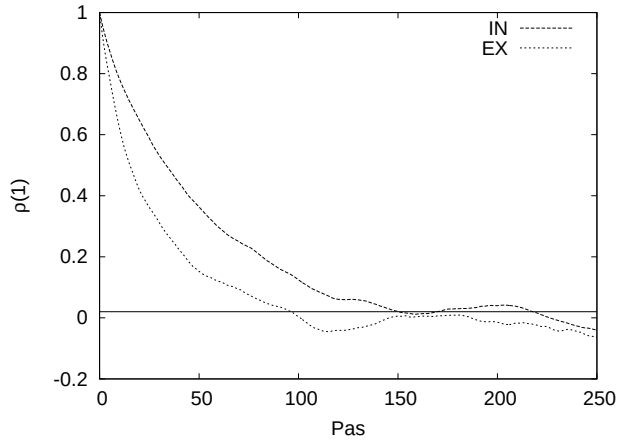


Figure 2.7 – Fonction d'autocorrélation des valeurs de fitness le long d'une marche aléatoire sur les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ de l'instance 100/20/01.

2.3.2 Étude des performances de métaheuristiques classiques

Le flowshop est un problème très étudié dans la littérature. La question de l'opérateur de voisinage pour les recherches locales, et des opérateurs génétiques (mutation, sélection, crossover) pour les algorithmes génétiques est posée et prend une part non négligeable du travail en amont de la résolution. Parmi la littérature, nous avons retenu deux articles. En effet, ceux-ci traitent des comparaisons des performances de métaheuristiques présentant dans leur conception les opérateurs de voisinage *insertion* et *échange*.

Den Besten *et al.* [dBS01] proposent une étude des performances du VND (*Variable Neighborhood Descent*) selon la relation de voisinage choisie. Le VND est une recherche locale qui alterne l'utilisation de différentes relations de voisinage dans une méthode de descente. Ici, les auteurs cherchent à montrer le bénéfice d'alterner entre deux relations de voisinage plutôt que d'en utiliser qu'une seule en testant sur des problèmes d'ordonnancement de tâches

sur une ou plusieurs machines. Ils utilisent donc le problème de flowshop avec minimisation de la date d'achèvement de la dernière tâche ordonnancée et présentent leur résultats pour les relations de voisinages $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$ respectivement basées sur les opérateurs de voisinage *insertion* et *échange*. Ainsi, ils comparent avec un temps CPU maximum, les performances entre un VND utilisant itérativement l'*insertion* puis l'*échange* et un VND utilisant itérativement l'*échange* et l'*insertion*. Pour montrer l'intérêt d'utiliser itérativement deux voisinages, ils comparent ces VND à deux recherches locales itérées composées d'une méthode de descente stricte (HC) ayant comme relation de voisinage $\mathcal{V}_{IN}$ pour la première et $\mathcal{V}_{EX}$ pour la deuxième. Les résultats qui nous intéressent sont les performances de ces deux dernières recherches locales itérées sur les instances de Taillard (20/20 ; 50/10-20 ; 100/20). Les performances sont données comme déviation par rapport à la meilleure valeur de fitness connue. Pour toutes les instances testées, la déviation est plus petite avec $\mathcal{V}_{IN}$ qu'avec $\mathcal{V}_{EX}$. *A priori* l'utilisation de l'opérateur *insertion* rend la recherche locale plus efficace.

Murata *et al.* [MIT96] étudient les performances des algorithmes génétiques (AG) pour le problème de flowshop avec minimisation de la date d'achèvement de la dernière tâche ordonnancée sur des instances de Taillard. Ils comparent alors plusieurs AG étant différenciés par leurs opérateurs génétiques. Ici, c'est la mutation qui nous intéresse. En effet, elle utilise un opérateur de voisinage pour apporter de la diversité dans une population. Les auteurs présentent quatre opérateurs de mutation dont deux qui attirent plus particulièrement notre attention : l'opérateur *insertion* et l'opérateur *échange*. Les performances sont comparées en fonction du nombre d'évaluation. La convergence d'une méthode peut être étudiée en regardant l'évolution de la valeur de fitness en fonction du nombre d'évaluation. Cette évolution informe sur la rapidité d'une méthode à trouver une bonne solution ou sur son incapacité à améliorer la qualité à partir d'un certain nombre d'évaluations. Les opérateurs d'*insertion* et d'*échange* obtiennent les meilleures performances sur les tests réalisés (instances 20/10). Mais qu'importe le nombre d'évaluations considéré, la prédominance de l'opérateur *insertion* quant à ses performances est indéniable. Tenant compte de ces observations, les auteurs poursuivent leur étude par la comparaison sur les instances de Taillard ($N \in \{20 : 50\}$), ayant le nombre d'évaluations comme critère d'arrêt, d'un AG ayant l'*insertion* comme mutation et trois recherches locales avec un voisinage basé sur l'*insertion* : recherche locale itérée (ILS), recherche tabou (TS) et recuit simulé (SA). Quelque soit l'instance, les performances donnent : TS > SA > ILS > GA.

Dans cet article, la prédominance de l'opérateur *insertion* en terme de performances est encore constatée pour les instances de Taillard. Les recherches locales sont aussi plus efficaces que les algorithmes génétiques pour ce type de problème et sont donc plus largement étudiées dans la littérature [RM05, RS07, Stü98].

Ces deux études montrent que les performances des métaheuristiques sont meilleures en utilisant l'opérateur *insertion* que l'opérateur *échange* sur les instances de Taillard pour le problème de Flowshop.

2.3.3 Discussion

Ce chapitre est consacré à l'influence du paysage sur les performances des métaheuristiques. Dans cette section, nous avons commencé par analyser deux paysages du problème de flowshop avec minimisation de la date d'achèvement de la dernière tâche ordonnancée, différenciés par leur relation de voisinage. Les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ sont respectivement définis par les relations de voisinage $\mathcal{V}_{IN}$ et $\mathcal{V}_{EX}$ elle-mêmes définies par l'application des opérateurs de voisinage d'*insertion* (IN) et d'*échange* (EX). Dans un deuxième temps, nous avons cherché à comparer les performances de certaines métaheuristiques présentant dans leur conception ces deux opérateurs de voisinage. L'étude de deux articles de la littérature donne des résultats sur l'efficacité d'une méthode selon la relation de voisinage utilisée par une recherche locale ou selon l'opérateur de voisinage utilisé dans l'opérateur de mutation d'un algorithme génétique. Cette section de la thèse permet de faire le lien entre ces deux études : comment en s'aidant de l'analyse des paysages aurions-nous pu prévoir les résultats des performances des différentes métaheuristiques ?

Les longueurs de marche sont en moyenne plus longues sur le paysage $\mathcal{P}_{IN}$ que sur le paysage $\mathcal{P}_{EX}$, le Hill Climbing descend donc plus profondément en se déplaçant sur $\mathcal{P}_{IN}$. Les valeurs d'autocorrélation sont plus élevées sur le paysage $\mathcal{P}_{IN}$ que sur le paysage $\mathcal{P}_{EX}$. Le paysage $\mathcal{P}_{IN}$ est donc moins accidenté *i.e.* plus continu et plus lisse localement. Ceci facilite le déplacement d'une recherche locale vers une bonne solution. De plus, la qualité des optima locaux évalués pour le paysage $\mathcal{P}_{IN}$ est meilleure que celle pour le paysage $\mathcal{P}_{EX}$. Ces deux remarques incitent à dire qu'une recherche locale exploitant le voisinage de l'*insertion* a plus de chance de trouver des bonnes solutions qu'une recherche locale exploitant le voisinage de l'*échange*. Les performances constatées par den Besten *et al.* [dBS01] soutiennent cette idée car leurs résultats montrent que la déviation par rapport à la meilleure solution connue est plus importante pour une recherche locale se déplaçant sur $\mathcal{P}_{EX}$ que sur $\mathcal{P}_{IN}$.
Murata *et al.* [MIT96] montrent qu'un algorithme génétique est plus efficace en utilisant l'opérateur *insertion* comme opérateur de mutation plutôt que l'opérateur *échange*. Clairement, l'algorithme génétique est influencé par le paysage local défini par la relation de voisinage basée sur l'opérateur de mutation.

Dans ces deux études, les recherches locales et les algorithmes génétiques obtiennent de meilleures performances quand l'opérateur *insertion* est utilisé. Dans la suite, nous considérons uniquement le paysage $\mathcal{P}_{IN}$. Les distances entre les optima locaux obtenus par un Hill Climbing montrent qu'ils sont assez proches les uns des autres. De plus, la fonction d'autocorrélation présente des valeurs élevées ($\geq 88\%$) qui décroissent très progressivement. Le paysage $\mathcal{P}_{IN}$ n'est donc pas très rugueux et se prête donc bien au déplacement de voisin en voisin. Les travaux de Murata *et al.* [MIT96] soutiennent cette hypothèse car ils montrent que l'algorithme génétique a toujours de moins bonnes performances que la recherche locale itérée, le recuit simulé et la recherche tabou. Ces valeurs d'autocorrélation permettent d'expliquer aussi la prédominance de la recherche tabou qui grâce à sa mémoire peut exploiter la continuité du paysage $\mathcal{P}_{IN}$.

Les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ présentent quand même certaines similarités. En effet, les lon-

gueurs de marches paraissent dans les deux cas assez courtes. Cette observation couplée à la qualité très moyenne des optima locaux obtenus dans les populations $OPop_{IN}$ et $OPop_{EX}$ en comparant aux meilleures valeurs connues de la littérature nous amène à nous demander si des caractéristiques liées au problème n'auraient pas été détectées avec cette analyse de paysage. En effet, en regardant les valeurs de fitness, nous observons que beaucoup d'entre elles sont égales. Y a-t-il beaucoup de solutions de même qualité ? Sont-elles toutes des optima locaux ? Comment sont-elles réparties dans l'espace de recherche ? Sont-elles voisines ? Ces questions sont assez importantes et nous tenterons d'y répondre dans la suite de cette thèse.

2.4 Conclusion

Ce chapitre prouve l'influence du paysage sur la dynamique de recherche d'une métaheuristique. Se basant sur les indicateurs de paysage présentés dans le premier chapitre, nous avons proposé une démarche qui nous paraît adaptée pour appréhender un problème d'optimisation par une analyse de paysage. La distance et l'entropie permettent de mieux connaître la dispersion des solutions d'une population sur l'espace de recherche et leurs ressemblances en terme de représentation. Les statistiques classiques sur les valeurs de fitness aident à évaluer la qualité des solutions regardées. Les longueurs de marche indiquent entre autres si une recherche locale peut se trouver bloquée facilement par un optimum local. Les valeurs d'autocorrélation montrent la rugosité du paysage. Les premières valeurs permettent d'avoir une vision plus locale du paysage et ainsi d'appréhender le comportement probable d'une métaheuristique vis-à-vis de ses performances.
Tous ces indicateurs ont été mis en œuvre sur deux problèmes issus de la logistique : un problème asymétrique de tournées de véhicules avec flotte fixe hétérogène (HFF-AVRP) et un problème d'ordonnancement de tâches sur des machines (Flowshop). Les mêmes opérateurs de voisinage, à savoir *insertion* (IN) et *échange* (EX), adaptés à chacune des représentations ont été choisis comme opérateurs de voisinage pour définir respectivement les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$.

Des différences comme des similitudes peuvent être tirées entre les deux problèmes. Pour le HFF-AVRP, les paysages $\mathcal{P}_{IN}$ et $\mathcal{P}_{EX}$ sont assez rugueux alors que pour le flowshop $\mathcal{P}_{EX}$ est beaucoup plus rugueux que $\mathcal{P}_{IN}$. En revanche, les distances et le diamètre des paysages amènent aux mêmes différences. L'opérateur *insertion* a tendance à contracter l'espace de recherche. En effet pour les deux problèmes, la distance maximale entre les solutions n'est jamais atteinte et les solutions sont beaucoup plus proches que ce soit les optima locaux ou des solutions choisies aléatoirement dans l'espace de recherche. Les longueurs de marches montrent aussi qu'une recherche locale descend plus profondément en se déplaçant sur le paysage $\mathcal{P}_{IN}$. Ainsi, pour les deux problèmes, il est clair que les recherches locales itérées obtiennent de meilleures performances avec le voisinage $\mathcal{V}_{IN}$ qu'avec le voisinage $\mathcal{V}_{EX}$. Quant aux performances des algorithmes génétiques, elles sont équivalentes à celle d'une recherche tabou pour le HFF-AVRP alors qu'elles sont moins bonnes dans le cas du flowshop. Ceci peut s'expliquer par la rugosité de $\mathcal{P}_{IN}$ pour le HFF-AVRP alors que le flowshop est plutôt "lisse" localement pour $\mathcal{P}_{IN}$.

Dans ce chapitre, nous avons montré pour chaque problème traités comment le paysage influence la dynamique d'une métaheuristique. L'analyse croisée sur ces deux problèmes de logistique met en évidence que certaines des caractéristiques du paysage sont données par l'opérateur de voisinage (ex : distance entre les solutions, longueurs de marche) alors que d'autres sont plus intrinsèques au problème (ex : longueurs de corrélation, entropie). Une étude du paysage est donc nécessaire pour informer sur ces caractéristiques afin de prédire et expliquer les performances des métaheuristiques. L'analyse de cette structure de paysage ne serait-elle pas aussi un bon moyen de concevoir des métaheuristiques tirant ainsi parti de ses caractéristiques ?

Chapitre 3

Neutralité : Structure de paysage et recherche locale

Publications en lien avec le chapitre :

- M.-É. Marmion, L. Jourdan, C. Dhaenens, A. Liefooghe, S. Verel, On the Neutrality of Flowshop Scheduling Fitness Landscapes, In *Proceedings of the 5th Learning and Intelligent OptimizatioN Conference*, LION 2011, LNCS, Springer, pages 1–15, 2011
- M.-É. Marmion, L. Jourdan, C. Dhaenens, A. Liefooghe, S. Verel, NILS : a Neutrality-based Iterated Local Search and its application to Flowshop Scheduling, In *Proceedings of the 12th European Conference of Evolutionary Computation in Combinatorial Optimization*, EvoCOP 2011, LNCS, Springer, pages 191–202, 2011

Sommaire

Le paysage d'un problème d'optimisation fournit une représentation géométrique au problème grâce à la relation de voisinage entre les solutions de l'espace de recherche et la fonction objectif associée au problème. Certaines solutions de l'espace de recherche peuvent avoir la même valeur de fitness (solution neutre au regard de l'optimisation). De nombreux problèmes d'optimisation présentent de telles solutions identiques en valeur. La relation de voisinage rapproche-t-elle ou non ces solutions de même valeur de fitness ? Peut-on parler de neutralité du paysage d'un problème d'optimisation ?
Dans cette section, nous définissons les paysages neutres puis nous décrivons des outils et des indicateurs permettant de caractériser ce type de paysage. Ensuite, nous utilisons, à titre d'exemple, ces indicateurs pour analyser le paysage du problème de flowshop de permutation et des paysages-NKq. La structure des paysages nous a amené à concevoir une recherche locale itérée efficace pour ce type de problème. Le but de ce chapitre est donc de montrer l'intérêt d'utiliser l'analyse de paysage pour concevoir une métaheuristique simple et efficace.

3.1 Neutralité et paysage

3.1.1 Définitions

Soient $\mathcal{V}$ une relation de voisinage pour les solutions de l'espace de recherche Ω et f la fonction objectif associée au problème d'optimisation. Un *voisin neutre* d'une solution s de l'espace de recherche est une solution voisine s' telle que sa valeur de fitness est égale à celle de s. Soit une solution $s \in \Omega$, l'ensemble de ses voisins neutres $\mathcal{V}_n(s)$ est défini par :

$$\mathcal{V}_n(s) = \{s' \in \mathcal{V}(s) \mid f(s') = f(s)\}$$

Le *degré de neutralité* d'une solution est le nombre de ses voisins neutres. Ainsi, pour une solution $s \in \Omega$, notons $|\mathcal{V}_n(s)|$, le degré de neutralité de la solution s. Un paysage est défini comme neutre quand il existe plusieurs solutions de l'espace de recherche ayant un degré de neutralité non nul.
Un *réseau de neutralité* est un sous-graphe connecté dont les sommets sont des solutions ayant la même valeur de fitness. Deux sommets d'un réseau de neutralité sont connectés s'ils sont voisins. Un réseau de neutralité est aussi appelé *plateau*. Une *porte* dans un réseau de neutralité est une solution dont au moins un de ses voisins est strictement de meilleure qualité. Rappelons qu'un *optimum local* est une solution dont aucun voisin n'est de meilleure qualité. Quand toutes les solutions d'un réseau de neutralité sont des optima locaux, on parle d'un *réseau de neutralité d'optima locaux*. Le réseau ne contient alors aucune porte. Notons dès à présent que nous différencions un plateau d'optima locaux et le plateau d'un optimum local qui, pour ce dernier, ne présuppose rien quant à l'existence ou non d'au moins une porte. Nous appelons plateaux des optima locaux au sens large, tout plateau découvert à partir d'un optimum local et pouvant donc contenir des portes. Dans la suite de ce mémoire, ce sont ces plateaux des optima locaux qui nous intéressent.

3.1.2 Outils et indicateurs

Certains problèmes d'optimisation présentent une structure neutre quand le degré de neutralité des solutions est non nul. Il est donc nécessaire d'avoir des outils pour identifier ce

type de problème et pour les caractériser.

3.1.2.1 Degré de neutralité

La moyenne des degrés de neutralité des solutions de l'espace de recherche permet de déterminer si le paysage est neutre ou non. Le *degré de neutralité moyen* du paysage se calcule à partir d'une population de solutions qui échantillonne l'espace de recherche. La structure du paysage est impactée par la présence et la distribution des optima locaux. Le degré de neutralité moyen des optima locaux se calcule lui, à partir d'une population d'optima locaux. Ces deux types de degrés de neutralité mettent en évidence la présence de plateaux dans le paysage, en général, et aussi de plateaux au niveau des optima locaux. Les plateaux sont la représentation de la neutralité d'un problème. Pour les problèmes neutres, l'analyse de paysage doit donc s'attacher à l'étude des caractéristiques de ses plateaux.

3.1.2.2 Les marches neutres

Les marches neutres sont utilisées pour décrire les plateaux d'un paysage neutre. Formellement, une marche neutre $(s_0, s_1, \ldots, s_m)$ partant de s et terminant en s' est une suite de solutions appartenant à l'espace de recherche où $s_0 = s$ et $s_m = s^{'}$ et pour tout $i \in [0, m-1]$, s_{i+1} est un voisin de s_i tel que $f(s_{i+1}) = f(s_i)$.
Une marche neutre échantillonne un ensemble de solutions voisines pour collecter des informations sur ces solutions d'un même plateau et ainsi permettre d'appréhender la structure du paysage. Au long de la marche neutre, le degré de neutralité de chacune des solutions est enregistré. Ce degré est alors utilisé pour estimer la régularité des solutions neutres sur le plateau. L'*autocorrélation du degré de neutralité* au long de la marche neutre [BPRV03] mesure la corrélation de la structure du plateau et permet de vérifier en comparant au modèle nul qu'il n'est pas aléatoire. Le modèle nul correspond à la même marche dont les solutions sont ordonnées aléatoirement. Il représente alors le même réseau mais de manière aléatoire et permet de donner une signification à la corrélation entre voisins du réseau de neutralité. La fonction d'autocorrélation ρ du degré de neutralité se calcule de la même manière que la fonction d'autocorrélation des valeurs de fitness le long d'une marche aléatoire (*cf.* section 1.4.2). Pour le modèle nul, les valeurs d'autocorrélation doivent être nulles. Au contraire pour le réseau étudié, si la première valeur d'autocorrélation $\rho(1)$ est proche de 1, alors la variation du degré de neutralité est faible entre deux solutions voisines. Dans ce cas, l'étude des plateaux devient significative pour décrire la neutralité du paysage.
Dès lors que l'analyse des marches neutres montre que les plateaux ont une structure propre au problème, il est nécessaire d'en tirer de l'information. En effet, le parcours d'une marche neutre de voisin en voisin de même qualité sur un plateau apporte des connaissances locales sur le paysage. Ces marches neutres bien que se déplaçant aléatoirement sont généralement assez longues et peuvent potentiellement boucler et visiter plusieurs fois les mêmes solutions. De telles marches neutres seraient donc moins fiables pour expliquer les propriétés des plateaux et donc des paysages neutres. Une première information à vérifier sur les marches neutres utilisées est que les solutions rencontrées sont toutes différentes pour que les données collectées soient d'autant plus fiables. Pour ce faire, le *pourcentage de solutions*

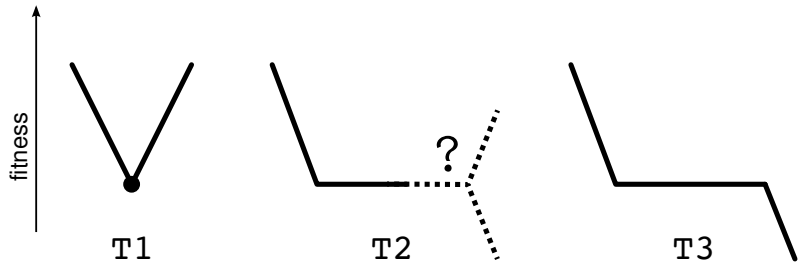


Figure 3.1 – Typologies des plateaux (problème de minimisation).

visitées au moins deux fois par la marche neutre est calculé. Une deuxième information à connaître est l'existence des portes sur les plateaux. Pour ceci, on regarde pour chaque solution de la marche neutre si elle a au moins un voisin qui a une meilleure qualité que la valeur de fitness du plateau. S'il existe au moins une porte sur un plateau, on peut supposer qu'il sera possible de sortir du plateau en continuant le processus d'optimisation.

3.1.2.3 Typologie des plateaux

Afin de nous repérer facilement, nous proposons une typologie de plateaux en trois types (Figure 3.1) :
- type T1 : le plateau est composé d'un et un seul optimum local qui n'a aucun voisin neutre. Le plateau est dit dégénéré ;
- type T2 : le plateau est considéré comme un plateau d'optima locaux ; la marche neutre n'a rencontré aucune porte, solution possédant un voisin avec une meilleure qualité que la valeur de fitness du plateau. Évidemment dans la majorité des cas, le plateau n'a pas été énuméré exhaustivement, néanmoins nos connaissances du plateau ne permettent pas de savoir s'il est possible de s'en échapper ;
- type T3 : le plateau possède au moins une porte, ce n'est donc pas un plateau d'optima locaux.

Pour étudier la structure neutre d'une instance, quelques plateaux sont échantillonnés par une marche neutre pour caractériser le paysage tout entier. Le *pourcentage de chaque type de plateaux* permet de savoir si le paysage présente des plateaux ou non, et s'ils sont des plateaux d'optima locaux ou au contraire, si au moins une porte peut être trouvée après un nombre raisonnable de déplacements sur le plateau. Pour des plateaux de type T3, le *nombre moyen de solutions visitées pour trouver une porte* indique si les portes sont facilement atteignables par une marche neutre se déplaçant aléatoirement entre les solutions d'un même plateau.

3.1.2.4 Compromis coût/qualité

Le *Netcrawler* (*cf.* 1.2.1.1) [Bar01] est une méthode de descente qui accepte les solutions neutres. L'utilisation d'une telle recherche locale paraît intéressante quand le paysage associé au problème d'optimisation présente de la neutralité. Il existe donc un compromis à étudier entre d'une part utiliser une méthode de descente de type Hill Climbing où la neutralité n'est pas prise en compte et d'autre part utiliser une méthode de type Netcrawler

qui accepte les mouvements vers les voisins neutres. Le *compromis coût/qualité* correspond à la comparaison du nombre de solutions visitées sur un plateau pour trouver une porte et le nombre de solutions visitées par une méthode de descente (type Hill Climbing) pour trouver un optimum local avec la qualité des solutions trouvées dans les deux cas.

3.2 Neutralité du problème de flowshop

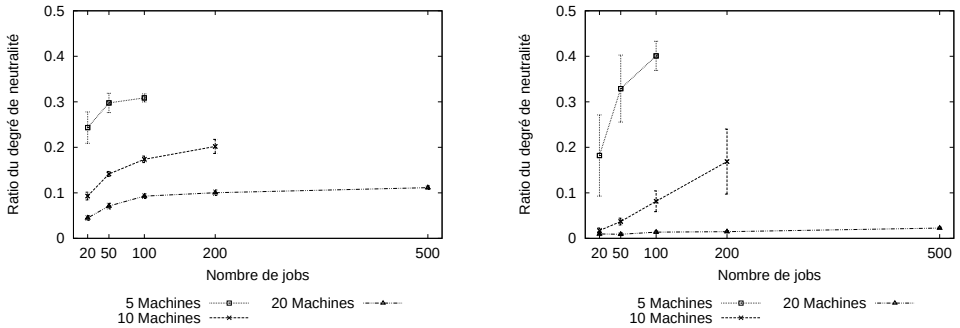
Nous souhaitons dans cette partie analyser la neutralité dans les problèmes de type flowshop de permutation. Plusieurs instances du flowshop avec minimisation de la date d'achèvement de la dernière tâche ordonnancée sont utilisées dans la littérature (*cf.* section 1.5.2). D'une part, les instances aléatoires de Taillard [Tai93] sont les plus classiques ; d'autre part, les instances structurées de Watson [WBWH02] sont plus rarement utilisées mais présentent de part leur construction une alternative différente aux précédentes.
Dans le chapitre 2, deux paysages du problème de flowshop différenciés par leur relation de voisinage ont été étudiés. Pour les instances de Taillard, l'étude de paysage a montré que beaucoup de solutions avaient la même valeur de fitness. Le paysage de ces instances est-il neutre ? Est-ce aussi le cas pour les instances dites structurées du flowshop ?
Dans la suite de cette thèse, nous considérons pour le problème de flowshop uniquement l'opérateur de voisinage *insertion* 3.4.2.1 car c'est le plus utilisé dans la littérature et pour les instances de Taillard, les métaheuristiques sont alors performantes (*cf.* section 2.3). Le paysage du flowshop est alors défini par la relation de voisinage basée sur l'*insertion* et par la fonction objectif qui minimise la date d'achèvement de la dernière tâche ordonnancée sur la dernière machine.
Dans cette section, la structure neutre des paysages du flowshop des instances aléatoires de Taillard et des instances structurées est étudiée en utilisant les outils présentés précédemment.

3.2.1 Degrés de neutralité

Pour le problème de flowshop, les instances aléatoires de Taillard et les instances structurées présentent des propriétés structurelles différentes [JSW09]. Nous commençons l'étude de la neutralité du paysage du flowshop par le calcul des degrés de neutralité moyens du paysage des différentes instances pour analyser si la différence entre les structures de paysage peut s'expliquer entre autre par les degrés de neutralité.

3.2.1.1 Design expérimental

Le degré de neutralité moyen est estimé à partir d'une population initiale de 100 solutions, à laquelle sont ajoutées 100 nouvelles solutions tant que le degré de neutralité moyen ne se situe pas dans un intervalle de confiance à 99%. L'interprétation du degré de neutralité est dépendante de la taille du voisinage, donc de la taille de l'instance. Le *ratio du degré de neutralité* correspond au pourcentage du rapport entre le degré de neutralité et la taille du voisinage d'une solution, et est alors utilisé pour comparer les instances de tailles différentes entre elles. Le degré de neutralité moyen de l'instance est calculé en utilisant des solutions choisies aléatoirement dans l'espace de recherche et le degré de neutralité



(a) (b)

Figure 3.2 – Ratio du degré de neutralité moyen des instances de Taillard pour des solutions choisies aléatoirement (a) et pour les optima locaux (b).

moyen des optima locaux est calculé en utilisant des optima locaux obtenus en utilisant un First Improvement Hill Climbing (*cf.* 1.2.1.1).

3.2.1.2 Instances aléatoires de Taillard

Taillard a proposé pour le problème de flowshop de permutation plusieurs instances de tailles différentes afin de minimiser la date d'achèvement de la dernière tâche ordonnancée. Pour chaque nombre de jobs N de l'ensemble $\{20, 50, 100, 200, 500\}$ et chaque nombre de machines M de l'ensemble $\{5, 10, 20\}$ dix instances ont été générées. La figure 3.2 présente les résultats des ratios moyennés des dix instances pour (a) des solutions choisies aléatoirement et (b) des optima locaux. Dans les deux cas, le degré de neutralité croît avec le nombre de jobs et décroît avec le nombre de machines. Pour 5 machines, les degrés de neutralité paraissent assez élevés puisque pour 50 et 100 jobs, plus d'un tiers des voisins d'une solution sont neutres. D'après les résultats, les optima locaux présentent moins de neutralité que les autres solutions. Pour les instances avec 20 machines, les ratios sont très faibles.
Afin de regarder plus en détails, les tableaux 3.1 et 3.2 présentent les degrés de neutralité et les ratios correspondants pour respectivement des solutions choisies aléatoirement et des optima locaux pour la première instance de chaque taille. Au regard de ses tableaux, même si les ratios sont assez faibles pour certaines instances, le degré de neutralité n'en demeure pas moins élevé car le nombre de voisins neutres est très grand. Ces voisins neutres sont autant de solutions à évaluer qui, du point de vue de la qualité, ne font pas progresser le processus d'optimisation.
Les instances de Taillard sont les plus utilisées dans la littérature pour les problèmes de type flowshop. Dans le chapitre 2, l'étude des performances de plusieurs métaheuristiques sur ces instances a montré que le paysage jouait un rôle important sur leurs performances. Les durées de tâches des instances de Taillard sont générées dans l'intervalle $[1; 99]$ selon une loi uniforme. Il est communément admis qu'une telle génération des durées engendre beaucoup de solutions de même valeur (de part l'échelle des valeurs). Il est donc intéres-

Tableau 3.1 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage (en %) de solutions choisies aléatoirement pour la première instance de Taillard de chaque taille.

Jobs	20	50	100	200	500
Voisinage	361	2401	9801	39601	249001
Machines					
5	87 - 24%	720 - 30%	3038 - 31%		
10	32 - 9%	336 - 14%	1666 - 17%	7930 - 20%	
20	14 - 4%	168 - 7%	882 - 9%	3960 - 10%	24900 - 10%

Tableau 3.2 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage (en %) des optima locaux pour la première instance de Taillard de chaque taille.

Jobs	20	50	100	200	500
Voisinage	361	2401	9801	39601	249001
Machines					
5	65 - 18%	792 - 33%	3920 - 40%		
10	7 - 2%	96 - 4%	784 - 8%	6732 - 17%	
20	4 - 1%	24 - 1%	98 - 1%	396 - 1%	4980 - 2%

sant de se demander comment le choix de la taille de l'intervalle pour générer les durées opératoires influence la structure de neutralité de ces instances.

3.2.1.3 Autres instances aléatoires

Pour répondre à cette question, nous avons généré de nouvelles instances qualifiées de "type Taillard". Nous considérons toujours que les durées des tâches suivent une distribution uniforme. Néanmoins, nous avons testé plusieurs intervalles différents. En effet, il se peut que la présence d'un job ayant une durée d'exécution très importante permette de déplacer et de réinsérer un job sans que la date d'achèvement de la dernière tâche ne soit impactée (créant un voisin neutre). De même, nous nous demandons si les durées étaient proportionnellement plus proches, la neutralité serait plus élevée ou non. Pour cette étude, nous considérons les instances de taille telle que le nombre de jobs appartienne à l'ensemble $\{20; 50; 100; 200\}$ et le nombre de machines à l'ensemble $\{5; 10; 20\}$. Les instances avec 500 jobs ne sont pas considérées car les calculs sont très longs et les autres tailles suffisent pour conclure. Pour chacune des tailles, dix instances ont été générées suivant quatre distributions uniformes avec des intervalles d'étendue différente :

- type A : $\mathcal{U}([1; 10])$. L'intervalle est plus petit, beaucoup de durées opératoires auront la même valeur ;
- type B : $\mathcal{U}([100; 1000])$. L'intervalle est neuf fois plus grand, les jobs auront des durées opératoires avec peu de valeurs communes ;

– type C : $\mathcal{U}([300;400])$ et $\mathcal{U}([900;1000])$: le nombre de valeurs possibles est le même, mais les proportions et donc l'impact sur l'ordonnancement des jobs et donc des durées seront différents.

La figure 3.3 présente pour chaque nombre de machines les ratios des degrés de neutralité moyens pour les solutions choisies aléatoirement (colonne de gauche) et les optima locaux (colonne de droite). Une première constatation est que, pour tous les cas envisagés, les degrés de neutralité sont équivalents et peu discernables entre les instances de Taillard et celles de type B et C. En effet, les courbes sont toujours entremêlées et aucun comportement différent n'est discernable pour l'une d'entre elles. De plus, comme pour les instances de Taillard, les degrés de neutralité des instances de types B et C sont généralement toujours plus élevés pour les solutions choisies aléatoirement que pour les optima locaux. En revanche, pour les instances de type A, les ratios des degrés de neutralité moyen des instances à 20 machines pour des optima locaux et pour des solutions aléatoirement choisies dans l'espace de recherche et ceux des instances à 10 machines pour ces dernières sont bien plus élevés que pour les instances de Taillard et celles de type B ou C. Il est donc évident que le fait d'avoir toutes les durées opératoires dans un intervalle très resserré favorise la neutralité des instances surtout quand le nombre de machines augmente.

Cette étude montre que, contrairement à ce que nous pourrions attendre, l'intervalle choisit par Taillard, pour les générations de ces durées opératoires impacte peu sur le degré de neutralité des instances engendrées. Regardons à présent ce qu'il en est des instances structurées.

3.2.1.4 Instances structurées

Watson *et al.* [WBWH02] ont proposé des instances présentant des corrélations entre les durées opératoires des tâches pour les jobs et/ou les machines. Ces instances sont présentées comme plus proches des instances réelles. Pour chaque nombre de jobs (20, 50, 100 ou 200) et uniquement 20 machines, un paramètre $\alpha \in \{0.0; 0.1; 0.2; ...; 1.0\}$ permet de régler les trois types de corrélations entre les jobs et/ou les machines. Rappelons qu'il existe les instances jobs-corrélées (jc), les instances machines-corrélées (mc) et les instances jobs/machines-corrélées (mxc). Pour les onze valeurs possibles de α, le degré de neutralité moyen a été calculé pour trois instances de chaque taille.

La figure 3.4 présente les ratios de degré de neutralité pour les instances à 20 et 200 jobs pour les solutions choisies aléatoirement dans l'espace de recherche (colonne de gauche) et les optima locaux (colonne de droite). Les ratios des instances de Taillard dont les durées sont générées suivant une distribution uniforme $\mathcal{U}([1;99])$ et des instances "type Taillard" générées suivant une distribution uniforme $\mathcal{U}([1;10])$ sont aussi représentés pour pouvoir comparer les instances structurées aux instances dites aléatoires. Les conclusions étant similaires quelque soit le nombre de jobs, les résultats pour 50 et 100 jobs sont donnés dans l'annexe B.

Quand α est égal à zéro, les degrés de neutralité sont quasi-identiques entre les trois types d'instances. En effet, elles sont construites de la même manière et ressemblent donc aux instances aléatoires générées selon la distribution uniforme $\mathcal{U}([1;10])$. Par contre, pour les autres valeurs de α, ces instances sont plus neutres que les instances aléatoires. En outre, plus l'instance est structurée *i.e.* α est élevé, plus le ratio est grand. Comme pour les instances aléatoires, les degrés de neutralité sont plus faibles pour les optima locaux

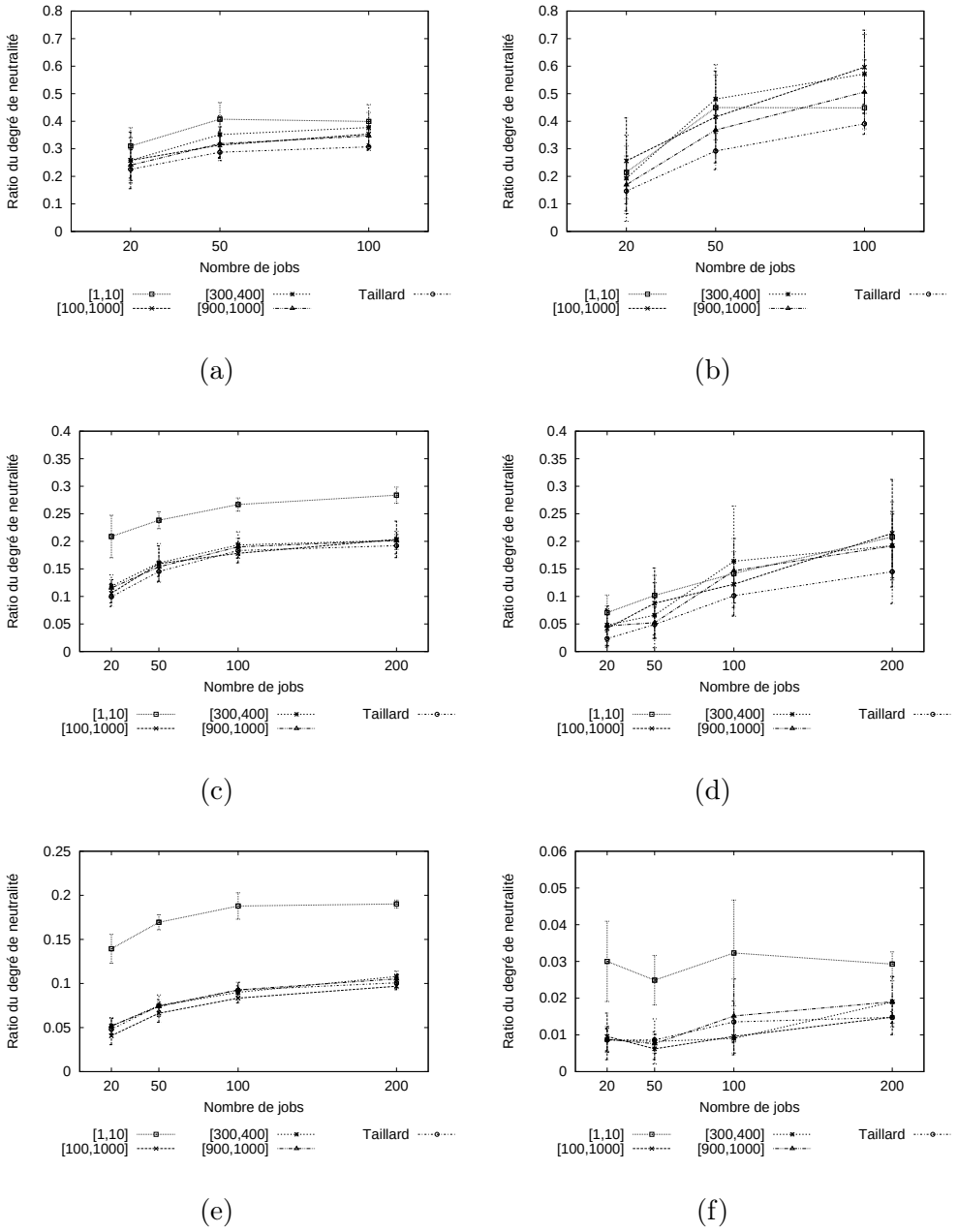


Figure 3.3 – Ratio du degré de neutralité moyen des instances de "type Taillard" pour des solutions choisies aléatoirement (a-c-e) et pour les optima locaux (b-d-f). Les résultats sont présentés pour 5 machines (a-b), 10 machines (c-d) et 20 machines (e-f).

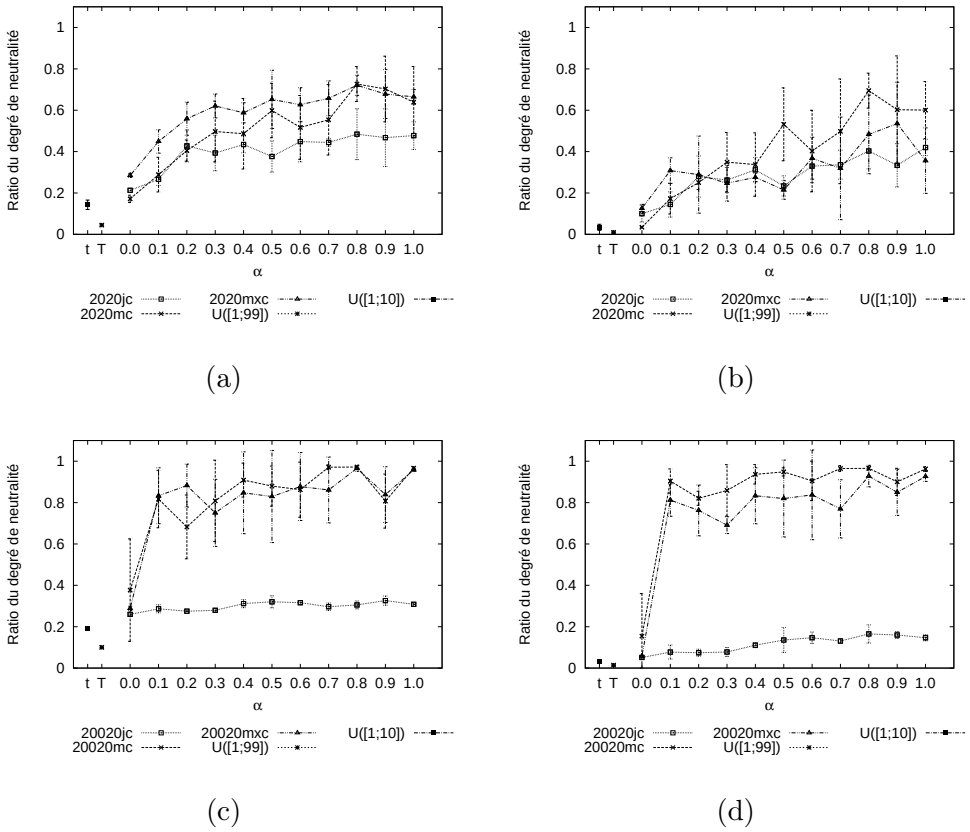


Figure 3.4 – Ratio du degré de neutralité moyen des instances structurées avec 20 machines pour des solutions choisies aléatoirement (a-c) et pour les optima locaux (b-d). Les résultats sont présentés pour 20 jobs (a-b), 200 jobs (c-d). Les valeurs de α sont en abscisse. Sont représentés pour comparaison l'instance "type Taillard" $\mathcal{U}([1;10])$ (t) et l'instance de Taillard $\mathcal{U}([1;99])$ (T).

que pour les autres solutions pour les instances jobs-corrélées. Les courbes des instances machines-corrélées et jobs/machines-corrélées s'entremêlent et montrent des ratios quasi-similaires et de plus en plus proches de un quand la valeur de α augmente. De plus, pour ces instances, plus le nombre de jobs est élevé et plus les ratios sont identiques entre les solutions choisies aléatoirement et les optima locaux. Par exemple pour 200 jobs et $\alpha = 1.0$, plus de 95% du voisinage d'une solution a la même valeur de fitness. Cette façon de structurer les instances présente donc deux types de neutralité : (i) un premier type avec une neutralité très proche de 1, pour les instances machines-corrélées et jobs/machines-corrélées et (ii) un deuxième type avec une neutralité plus proche de celles des instances aléatoires pour les instances jobs-corrélées.

3.2.1.5 Synthèse

Les instances aléatoires de Taillard ou de "type Taillard" et les instances structurées ont été analysées sous l'angle de la neutralité. Pour toutes ces instances, la neutralité existe. Cependant pour une même instance, il existe plusieurs niveaux de neutralité ; les optima locaux sont souvent moins neutres que les autres solutions. Pour les instances structurées, la neutralité peut être très élevée quelque soit la solution considérée. Pour les instances de Taillard, le ratio du degré de neutralité des optima locaux est quasi nul mais peut néanmoins représenter un grand nombre de solutions de l'espace de recherche pouvant bloquer les recherches locales qui ne tiennent pas compte de la neutralité du problème.
Quelque soit le degré de neutralité moyen d'une instance, il ne peut être ignoré. En effet, les solutions de même qualité sont voisines ou font partie d'un même plateau du paysage. Il semble donc important d'étudier cette caractéristique du paysage pour résoudre ces instances. L'étude des plateaux est donc l'étape suivante pour mieux comprendre la structure neutre du problème.

L'intérêt de ce chapitre est d'utiliser des informations du paysage pour concevoir et/ou paramétrer des métaheuristiques dans le cadre de problèmes neutres. Pour étayer cette étude, il nous faut choisir des instances ayant des structurations différentes et étant diversifiées en terme de neutralité. Les ratios des degrés de neutralité des instances de "type de Taillard" que nous avons proposé, étant soit proches de celles de Taillard soit des instances structurées pour α égal à zéro, nous avons décidé de ne pas les exploiter pour la suite. En revanche, les instances de Taillard montrent pour la plupart d'entre-elles des degrés de neutralité assez faibles alors que les instances machines-corrélées et jobs/machines-corrélées présentent des ratios de degré de neutralité moyen très élevés, quasi égaux à 1, que ce soit pour des solutions choisies aléatoirement ou des optima locaux. Les expérimentations ayant des durées très longues, nous ne pouvons pas utiliser toutes les instances structurées pour toutes les valeurs possibles de α. Étant donné que pour $\alpha = 0.8$, les instances structurées peuvent présenter de forts degrés de neutralité et sont bien différentes de celles de Taillard, nous choisissons de poursuivre notre étude uniquement pour cette valeur du paramètre α. Ainsi, nous considérerons pour les expérimentations et pour chaque type de corrélations des durées opératoires une instance de chaque taille (nombre de jobs) avec le paramètre $\alpha = 0.8$.

3.2.2 Structure du paysage neutre

Comme présenté dans la première section de ce chapitre, la structure d'un paysage neutre est étudiée grâce aux caractéristiques des plateaux. Nous avons vu précédemment, que le problème de flowshop présente un degré de neutralité jamais nul et donc possède de nombreux plateaux à tous niveaux que ce soit pour les instances de Taillard ou pour les instances structurées.

3.2.2.1 Design expérimental

Lors du calcul du degré de neutralité des optima locaux, la longueur de marche (longueur de la descente) nécessaire entre chaque solution de départ choisie aléatoirement et l'op-

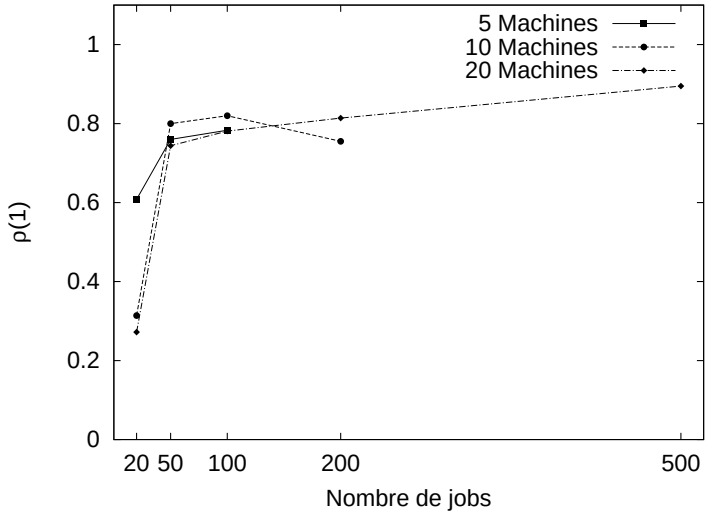


Figure 3.5 – Première valeur de l'autocorrélation $\rho(1)$ du degré de neutralité entre voisins d'un plateau en fonction du nombre de jobs pour les instances de Taillard.

timum local associé a été enregistrée. Pour chaque instance, la valeur maximale l_{max} de ces longueurs de marche est utilisée pour calculer la longueur maximale de marche neutre. En effet, pour étudier le compromis coût/qualité il est nécessaire d'avoir une longueur de marche neutre pouvant être comparable. 30 marches neutres partant de 30 solutions choisies aléatoirement sont effectuées. Ces marches sont de longueur P_{max} proportionnelle à l_{max}. Notons qu'étant donnée la valeur de l_{max}, la longueur des marches neutres est alors suffisamment grande pour caractériser les plateaux du paysage. Pour chaque solution d'une marche, tout son voisinage est évalué dans le but de connaître son nombre de voisins améliorants, neutres et dégradants. Chaque instance est donc échantillonnée par 30 de ses plateaux.

3.2.2.2 Instances aléatoires de Taillard

Dans cette partie, seule la première instance de Taillard pour chaque taille est utilisée. Pour chacune des instances considérées, 30 marches neutres de longueur $\mathrm{P}_{max} = 10 \times l_{max}$ sont effectuées.

Autocorrélation du degré de neutralité L'autocorrélation du degré de neutralité au long d'une marche neutre sur un plateau mesure le caractère non aléatoire du réseau selon la propriété de neutralité. La figure 3.5 présente la moyenne sur 30 marches neutres de la première valeur de l'autocorrélation $\rho(1)$ du degré de neutralité pour 5, 10 et 20 machines en fonction du nombre de jobs. Pour toutes les instances, la première valeur de l'autocorrélation du modèle nul est au-dessous de 0.01. Ainsi, l'autocorrélation du degré de neutralité au long de la marche est une conséquence de la succession des solutions visitées lors du déplacement de voisin en voisin sur le plateau. Pour les instances avec 50, 100, 200 et 500 jobs, cette valeur est très élevée et proche de 1 ce qui signifie que le degré de neutralité est très corrélé entre les voisins (plus grand que 0.7). Pour les instances 20-

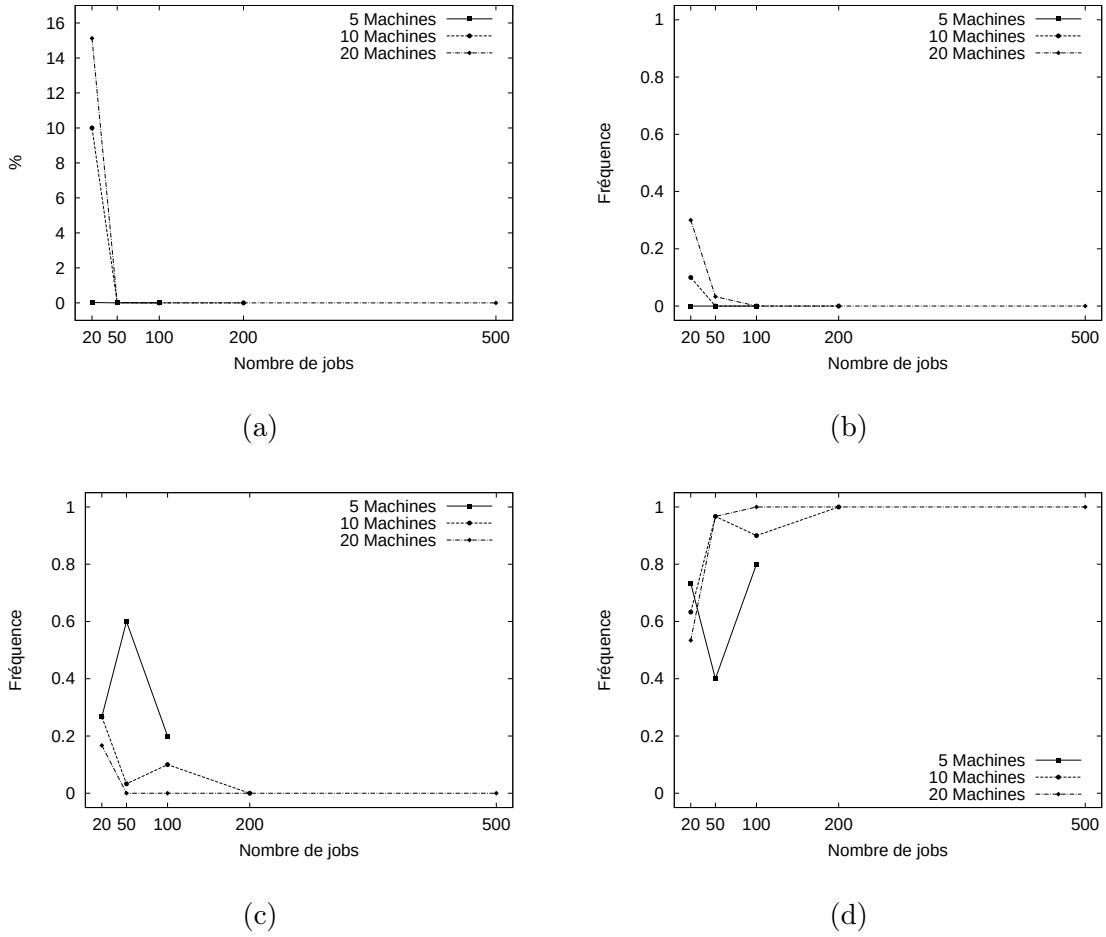


Figure 3.6 – Pourcentage moyen de solutions visitées au moins deux fois par les 30 marches neutres (a) et fréquences moyennes des types de plateaux rencontrés T1 (b), T2 (c) et T3 (d) pour les instances de Taillard.

jobs 10-, 20- machines, la valeur d'autocorrélation est autour de 0.4 et est donc moins significative. Ceci est dû au très faible degré de neutralité de ces instances (respectivement 7 et 4, *cf.* tableau 3.2). Néanmoins, le degré de neutralité d'une solution est tout de même partiellement corrélé à celui de ses voisins.
Nous pouvons donc déduire que les plateaux, définis comme des graphes de solutions voisines de même valeur de fitness, ne sont pas aléatoires. Le voisinage d'une solution donne des informations à traiter pour caractériser et se déplacer au mieux sur ces plateaux. Par conséquent, dans un processus d'optimisation, il semble efficace de parcourir les plateaux en utilisant ces informations entre voisins.

Échantillonnage des plateaux Une marche neutre sur un plateau se déplace aléatoirement de voisin en voisin et peut repasser deux fois sur la même solution. Il est nécessaire de

vérifier que ce ne soit pas le cas pour les marches neutres utilisées pour décrire le paysage neutre. La figure 3.6 (a) montre le pourcentage moyen de solutions visitées au moins deux fois par les 30 marches neutres utilisées pour caractériser le paysage de chaque instance. Pour les instances avec 50, 100, 200 et 500 jobs, il n'y a aucune solution qui soit dans ce cas là. Les marches neutres ont donc toutes visité des solutions différentes. Les informations récoltées au long de ces marches sont d'autant plus fiables pour caractériser leur propre plateau. Pour les instances 20- jobs 10- ou 20- machines, respectivement en moyenne 10% et 15% des solutions des marches neutres sont rencontrées au moins deux fois. Comme remarqué dans le paragraphe précédent, ces deux instances présentent chacune un très faible degré de neutralité. Les plateaux ne doivent pas être très larges et la marche revisite donc des solutions déjà connues.
Ces résultats sur les marches neutres et les solutions qui les composent montrent que les plateaux sont généralement très larges (sauf pour les problèmes de plus petite taille). Il est nécessaire de prendre en considération cette propriété du paysage lors de la résolution d'un problème avec des méthodes utilisant les voisinages.

Typologies des plateaux Nous avons défini trois typologies de plateaux : (T1) les plateaux dégénérés constitués d'un unique optimum local, (T2) les plateaux semblant être constitués uniquement d'optima locaux et (T3) les plateaux possédant au moins une porte. Ce qui nous intéresse ici c'est de connaître la proportion de chacun de ces types dans l'ensemble des plateaux échantillonnés. La figure 3.6 (b) montre la fréquence de plateaux de type T1, (c) la fréquence de plateaux de type T2 et (d) la fréquence de plateaux de type T3. Pour les instances avec 100, 200 et 500 jobs, les paysages ne semblent pas présenter de plateaux de type T1. Les 30 optima locaux trouvés appartiennent donc à un plateau ayant au moins deux solutions. La figure 3.6 (a) permet même d'affirmer que ces réseaux possèdent au moins autant de solutions que la taille de la marche neutre. Pour les instances 20- jobs 10-, 20- machines, respectivement en moyenne 10% et 30% des optima locaux n'ont aucun voisin de même qualité. Pour l'instance 50- jobs 20- machines seulement un plateau dégénéré (type T1) a été identifié. Le nombre de plateaux sur lesquels aucune porte n'a été trouvée (type T2) n'est significatif que sur les instances à 5 machines ou à 20 jobs (supérieur à 20%). Ceci peut s'expliquer par le ratio du degré de neutralité qui est assez élevé et qui engendre des plateaux larges sur lesquels il est facile de se perdre et de ne pas rencontrer de portes. Les instances 20- machines 50-, 100-, 200-, 500- jobs qui sont connues pour être les plus difficiles à résoudre de manière optimale ne présentent, ici, que des plateaux de type T3. Les optima locaux sont nombreux mais il est possible de s'échapper vers une solution avec une meilleure valeur de fitness. L'exploitation des plateaux peut être un bon moyen d'aider le processus d'optimisation.

Portes sur les plateaux Maintenant, nous considérons les marches neutres qui ont rencontré au moins une porte sur leur plateau. Nous pouvons alors savoir si les portes sont rapidement atteignables par une marche neutre se déplaçant aléatoirement sur le plateau à partir d'un optimum local. Le tableau 3.3 présente le minimum, la médiane, la moyenne et le maximum du nombre de solutions à visiter par plateau pour rencontrer une porte pour les marches neutres se déplaçant sur un plateau de type T3. Le nombre de marches neutres ayant trouvé une porte (type T3) et utilisé pour ce calcul est indiqué ainsi que le nombre

Tableau 3.3 – T3 donne le nombre de plateaux de type T3. P_{max} donne la longueur des marches neutres. nbS donne le nombre de solutions nécessaires à visiter pour rencontrer une porte en se déplaçant sur le plateau. LgM donne la longueur de marche moyenne pour trouver les optima locaux. Toutes ces valeurs sont données pour les différentes tailles des premières instances de Taillard.

	T3	P_{max}	nbS				LgM			
Instances	nb	nb	Min	Med	Moy	Max	Min	Med	Moy	Max
20/ 5/01	22	130	2	**10**	28.8	117	2	**7**	7.2	13
20/10/01	19	210	2	**2**	5.7	24	5	**13**	13	21
20/20/01	15	190	2	**2.5**	14.3	114	6	**11.5**	11.6	19
50/ 5/01	12	200	2	**18**	51.2	152	4	**10**	10.6	20
50/10/01	29	350	2	**4**	19.8	295	10	**24**	24.6	35
50/20/01	29	400	2	**2**	3.7	19	11	**27.5**	27.2	40
100/ 5/01	24	310	2	**4**	6.4	24	5	**16**	16.3	31
100/10/01	27	480	2	**8**	15.5	122	19	**33**	33.5	48
100/20/01	30	710	2	**4**	6.3	28	21	**42**	42.2	71
200/10/01	30	510	2	**6**	18.1	129	18	**32**	31.5	51
200/20/01	30	980	2	**3**	4.2	26	34	**66.5**	66.3	98
500/20/01	30	1540	2	**6**	11.4	107	71	**113**	112.9	154

de pas neutres P_{max} effectués. Pour chaque instance, au moins une marche neutre a réussi à trouver une porte en visitant seulement deux solutions. La médiane et la moyenne sont précisées pour avoir une idée de la répartition des valeurs trouvées. Pour les instances 20-machines, la moyenne est proche de la médiane ce qui montre qu'elle est représentative de l'ensemble. Ces instances qui sont considérées comme difficiles présentent des optima locaux dont il est facile de s'échapper en se déplaçant de voisin en voisin de même qualité. En général, quand une marche neutre rencontre une porte, il n'est pas nécessaire de faire beaucoup de pas. Il faut donc étudier le compromis entre le nombre de solutions à visiter sur le plateau pour rencontrer une porte et le nombre de pas à effectuer le long d'une méthode de descente pour trouver un optimum local qui peut être meilleur ou non.

Compromis coût/qualité Le compromis coût/qualité consiste à regarder le paysage du point de vue de la dynamique d'une recherche locale. Une fois un optimum local trouvé, est-il plus rapide de redémarrer à partir d'une autre solution pour trouver un nouvel optimum local meilleur ? Ou, est-il plus rapide de profiter de la neutralité du paysage pour se déplacer sur les plateaux pour rencontrer une porte pour ensuite être assuré de trouver une solution de meilleure qualité ? Pour ceci, nous comparons le nombre de solutions que visite une marche neutre afin de rechercher une porte et le nombre de pas nécessaires à une méthode de descente pour trouver un optimum local *i.e.* la longueur de marche. Le tableau 3.3 présente, en regard du nombre de solutions du plateau à visiter pour rencontrer une porte, la longueur de marche. Premièrement, pour toutes les instances, exceptée l'instance 20-jobs 5- machines, le nombre de solutions minimum à visiter pour rencontrer une porte est plus petit que la longueur de marche pour trouver un optimum local. Deuxièmement, pour

les instances 10- et 20- machines, il est incontestable qu'il est plus efficace de se déplacer sur les plateaux, car les optima locaux appartiennent généralement à des plateaux qui contiennent des portes rapidement atteignables.
Cette propriété de neutralité du paysage du flowshop ne doit donc pas être ignorée. En effet, pour chacune des instances, la majorité des 30 optima locaux trouvés après une méthode de descente appartient à un plateau non dégénéré. Ces plateaux sont larges mais les marches aléatoires se déplaçant sur ceux-ci conduisent assez rapidement pour la plupart à des portes. Ces solutions représentent de nouveaux départs pour qu'une recherche locale améliore la valeur de fitness courante.

3.2.2.3 Instances structurées

Nous nous intéressons maintenant au paysage neutre de quelques instances structurées du problème de flowshop. Ces instances ont l'intérêt d'être très neutres par rapport aux instances aléatoires de Taillard ce qui permet d'étudier plusieurs configurations de neutralité dans un même cadre. Nous avons choisi d'étudier plus spécifiquement pour chaque taille (20, 50, 100 ou 200 jobs et 20 machines) et chaque type de corrélation, uniquement une instance pour $\alpha = 0.8$. Pour une telle valeur du paramètre α, nous avons vu dans la section 3.2.1 que les instances machines-corrélées (mc) et jobs/machines-corrélées (mxc) présentent une neutralité très forte : en moyenne, plus de 80% des voisins d'une solution appartiennent au même réseau de neutralité. Les instances jobs-corrélées (jc) sont quant à elles moins neutres (entre 15% et 30%) mais sont néanmoins plus neutres que les instances aléatoires de Taillard de même taille. Ces 12 instances étant très neutres doivent avoir des plateaux très larges. Dans le but de caractériser au mieux la structure des plateaux du paysage de ces instances, nous avons donc exécuté 30 marches neutres de longueur $P_{max} = 100 \times l_{max}$. Pour l'instance 20020jc, par souci de temps et de mémoire, nous avons limité les 30 marches neutres à $10 \times l_{max}$ pas neutres.

Autocorrélation du degré de neutralité De même que précédemment, la figure 3.7 présente la moyenne sur 30 marches neutres de la première valeur de l'autocorrélation du degré de neutralité pour les trois types de corrélations (jc, mc et mxc) en fonction du nombre de jobs. Pour toutes les instances, la première valeur de l'autocorrélation du modèle nul est au-dessous de 0.01. Ainsi, l'autocorrélation du degré de neutralité au long de la marche est une conséquence de la succession des solutions visitées lors du déplacement de voisin en voisin sur le plateau. Pour chaque taille et type de corrélation, cette valeur est supérieure à 75%, le degré de neutralité est très corrélé entre les voisins. Les plateaux, définis comme des graphes de solutions voisines de même valeur de fitness, ne sont pas aléatoires. Par conséquent, dans un processus d'optimisation, il peut sembler efficace de parcourir ces plateaux non, par des marches aléatoires, car le voisinage d'une solution donnant des informations à traiter pour caractériser et se déplacer au mieux sur ces plateaux.

Échantillonnage des plateaux Pour chacune des instances, le pourcentage moyen de solutions visitées au moins deux fois par les 30 marches neutres est égal à zéro, ce qui signifie que les marches neutres caractérisent d'autant mieux les plateaux. Ces résultats sur

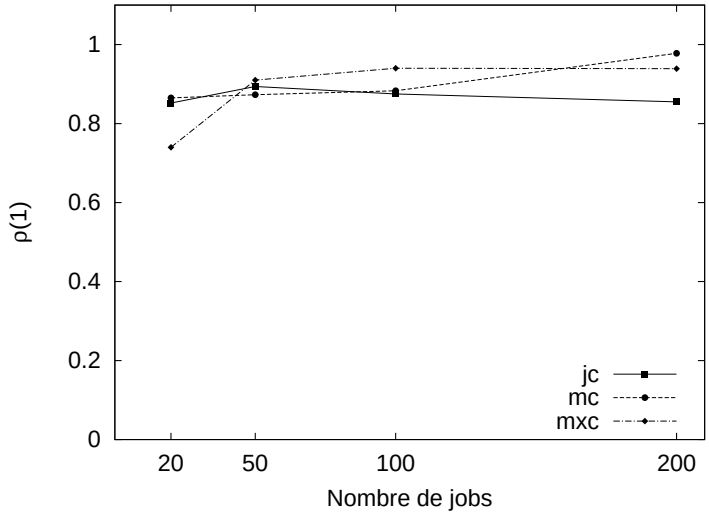


Figure 3.7 – Première valeur de l'autocorrélation $\rho(1)$ du degré de neutralité entre voisins d'un plateau en fonction du nombre de jobs pour les instances structurées.

les marches neutres et les solutions qui les composent montrent que les plateaux sont généralement très larges. Ces plateaux pouvant représenter autant de barrières au processus d'amélioration de la valeur de fitness courante, il est nécessaire de prendre en considération cette propriété du paysage lors de la résolution d'un problème avec des méthodes utilisant les voisinages.

Typologies des plateaux En plus des trois types de plateaux (T1, T2 et T3), notons T_{global} les plateaux constitués d'optima globaux (best-known de la littérature). Ce qui nous intéresse ici c'est de connaître la proportion de chacun de ces types dans l'ensemble des plateaux échantillonnés. Dans notre étude, aucun plateau de type T1 n'a été trouvé pour aucune des instances. Le tableau 3.4 donne le nombre de plateaux de type T_{global}, T2 et T3 visités par les 30 marches neutres. Pour les instances structurées, beaucoup de plateaux parcourus correspondent à ceux des optima globaux. Seules les instances 5020jc et 10020mxc présentent des plateaux de type T2 dont les marches neutres n'ont pas mené à des portes. Pour l'instance 20020mc, tous les optima trouvés après la méthode de descente sont globaux. Le paysage semble donc assez plat et l'instance facile à optimiser. Dans la majorité des cas, les plateaux, excepté T_{global}, présentent des portes.

Portes sur les plateaux Considérons uniquement les marches neutres ayant rencontré au moins une porte sur leur plateau. Nous pouvons alors savoir si les portes sont rapidement atteignables par une marche neutre se déplaçant aléatoirement sur le plateau à partir d'un optimum local. Le tableau 3.5 présente le minimum, la médiane, la moyenne et le maximum du nombre de solutions à visiter par plateau pour rencontrer une porte pour les marches neutres se déplaçant sur un réseau de type T3. Le nombre de marches neutres utilisées pour ce calcul est indiqué ainsi que le nombre de pas neutres P_{max} effectués. Pour ces instances, certains résultats sont à prendre avec précaution car pour la moitié des instances, du fait

Tableau 3.4 – Typologies des plateaux identifiés lors des 30 marches neutres pour les instances structurées. T_{global} correspond aux plateaux des optima globaux.

Instances	T_{global}	T2	T3
2020jc	20	0	10
5020jc	9	1	20
10020jc	0	0	30
20020jc	0	0	30
2020mc	2	0	28
5020mc	29	0	1
10020mc	22	0	8
20020mc	30	0	0
2020mxc	23	0	7
5020mxc	0	0	30
10020mxc	20	6	4
20020mxc	29	0	1

Tableau 3.5 – T3 donne le nombre de plateaux de type T3. P_{max} donne la longueur des marches neutres. nbS donne le nombre de solutions nécessaires à visiter pour rencontrer une porte en se déplaçant sur le plateau. LgM donne la longueur de marche moyenne pour trouver les optima locaux. Toutes ces valeurs sont données pour les instances structurées.

	T3	P_{max}	nbS				LgM			
Instances	nb	nb	Min	Med	Moy	Max	Min	Med	Moy	Max
2020jc	10	2600	2	**5**	9.3	31	6	**15**	15.1	26
5020jc	20	4900	2	**17**	47.8	248	20	**33**	32.9	49
10020jc	30	10400	2	**4.5**	6.2	17	40	**73**	73.9	104
20020jc	30	2060	2	**4**	4.6	12	125	**165.5**	165.9	206
2020mc	28	1700	2	**37**	98.2	863	1	**7**	7.3	17
5020mc	1	2200	24	**24**	24	24	4	**9**	9.9	22
10020mc	8	3900	5	**49**	72.4	272	9	**22**	22.1	39
20020mc	0	1800	-	**-**	-	-	-	**-**	-	-
2020mxc	7	1400	2	**14**	28.9	114	3	**8**	7.7	14
5020mxc	30	3400	2	**9.5**	13.8	49	9	**20**	20.6	34
10020mxc	4	2100	222	**621**	766.3	1609	7	**13**	13.5	21
20020mxc	1	4500	106	**106**	106	106	22	**32**	32	45

du large nombre de plateaux de type T_{global}, les valeurs présentées sont calculées sur un échantillon très peu représentatif de l'instance en entier. Pour les instances jobs-corrélées

100-, 200- jobs, il est très rapide de s'échapper du plateau car en médiane il suffit de visiter quatre solutions pour trouver une porte. En revanche, pour l'instance 20- jobs, seulement dix plateaux sont de type T3, donc les cinq pas neutres pour s'échapper du plateau ne sont pas très significatifs de la facilité à améliorer en suivant un chemin neutre. Pour l'instance 50- jobs, sur les 20 marches neutres qui ont rencontré au moins une porte, il a fallu visiter en moyenne 47.8 solutions pour trouver une porte (médiane = 17), ce qui montre une assez grande instabilité quant à la facilité de s'échapper des plateaux de son paysage. Pour l'instance jobs/machines-corrélées 50- jobs, les portes semblent assez faciles à atteindre car en médiane il est seulement nécessaire de visiter 9.5 solutions pour s'échapper du plateau par une porte. En revanche, pour l'instance machines-corrélées à 20 jobs, le nombre de pas neutres médian (= 37) est assez élevé et loin de la moyenne (98.2) ce qui n'encourage pas à exploiter les plateaux dans un processus d'optimisation.

Compromis coût/qualité Pour évaluer un compromis coût/qualité nous comparons le nombre de solution à visiter sur le plateau pour que la marche neutre rencontre une porte et le nombre de pas nécessaires à une méthode de descente pour trouver un optimum local *i.e.* la longueur de marche. Le tableau 3.5 présente, en regard du nombre de solutions à visiter pour rencontrer une porte (nbS), la longueur de marche (LgM).
Clairement, pour les trois instances où tous les plateaux regardés sont de type T3 (10020jc, 20020jc et 5020mxc), le nombre de solutions visitées sur un plateau pour rencontrer une porte est inférieur à la longueur de marche correspondante. L'exploitation des plateaux jusqu'à trouver une porte semble plus judicieuse que recommencer à chercher à partir d'une solution choisie aléatoirement dans une nouvelle partie de l'espace de recherche. En revanche, pour les instances machines-corrélées ou jobs/machines-corrélées (excepté l'instance 5020mxc), le paysage semble assez plat car les descentes jusqu'aux optima locaux sont courtes et que la plupart d'entre eux sont des optima globaux étant donné le nombre de plateau de type T_{global}. Pour ces instances, l'exploitation des plateaux ne semblent pas intéressante car il est rapide de trouver un optimum global à partir d'une solution choisie aléatoirement dans l'espace de recherche.

3.2.3 Synthèse

Dans cette section, plusieurs instances pour le flowshop ont été traitées. Les instances de Taillard sont dites aléatoires alors que les instances structurées sont générées de façon à avoir des corrélations entre les durées opératoires pour les jobs et/ou les machines. Quelques soient l'instance et la taille, le paysage présente de la neutralité. Par contre, selon le type de l'instance, aléatoire ou structurée, la structure du paysage est différente.
Le ratio du degré de neutralité des instances de Taillard augmente avec le nombre de jobs mais il diminue avec le nombre de machines. Pour les instances 5 machines, il peut atteindre 30% alors que pour les instances 20- machines, il est autour de 1%. Néanmoins, même un faible ratio impacte de manière importante le paysage. En effet, dans la majorité des cas, les optima locaux sont sur des plateaux où les portes peuvent être nombreuses et rapidement atteignables. Il semble donc intéressant d'utiliser la neutralité dans le processus d'optimisation.
Nous pensions que la neutralité de ces instances étaient dûes à la génération des durées

opératoires selon une loi uniforme dans l'intervalle $[|1; 99|]$. Des instances aléatoires suivant des tailles d'intervalles différentes ont donc été générées. En fait, les degrés de neutralité ne se sont pas avérés différents de ceux des instances de Taillard exceptés pour ceux de l'intervalle $[|1; 10|]$. Ces instances n'ont donc pas fait l'objet d'une étude plus approfondie. Quant aux instances structurées, elles sont très neutres. Pour les instances dont les machines sont corrélées et dont les machines et les jobs sont corrélés, la neutralité peut atteindre jusqu'à 97% de la taille du voisinage. Leurs plateaux sont donc immenses. Le best-known est trouvé par de simples méthodes de descentes. Pour les instances dont les jobs sont corrélés, la neutralité étant moins forte, les plateaux sont plus dispersés mais présentent de nombreuses portes rapidement atteignables. Là encore, la neutralité peut être intéressante à prendre en considération.
Afin de compléter la pertinence de cette analyse de structure neutre d'un problème d'optimisation combinatoire, un autre type de problème que le flowshop est étudié suivant la même méthodologie.

3.3 Neutralité des paysages-NKq

Les paysages-NK sont des problèmes de maximisation dits académiques. Les solutions sont représentées par un vecteur de N bits où les interactions entre les bits se règlent à l'aide de la valeur de K et définissent la rugosité du problème. Les paysages-NKq sont beaucoup moins utilisés mais permettent, par l'ajout du paramètre q de créer des paysages présentant une propriété de neutralité (*cf.* 1.5.1). Deux solutions sont voisines si elles ont un et un seul bit de différence *i.e.* la distance de Hamming entre deux voisins est égale exactement à un. Si le changement de bit ne fait pas varier la valeur de fitness de la solution, les solutions sont qualifiées de voisins neutres. Dans cette thèse, uniquement les paysages-NKq sont traités considérant trois tailles de problème à savoir $N \in \{64, 128, 256\}$. Le paramètre K appartient à l'ensemble $\{2, 4, 6, 8\}$ et q à l'ensemble $\{2, 3, 4\}$. Dans cette section, nous étudions la rugosité puis la neutralité des paysages-NKq puis la structure neutre de ces problèmes est décrite.

3.3.1 Rugosité

Dans le chapitre 1, nous avons défini la rugosité du paysage comme une mesure de la variabilité des valeurs de fitness des solutions voisines. La rugosité du paysage impacte la difficulté à résoudre un problème car les solutions voisines sont toutes différentes. La fonction d'autocorrélation [Wei90] permet de mesurer la rugosité en calculant les corrélations des valeurs de fitness entre les solutions distantes de d dans l'espace de recherche. Pour les paysages-$NK(q)$, le paramètre K règle cette rugosité, c'est l'épistasie du problème. La fonction d'autocorrélation des valeurs de fitness calculée le long d'une marche aléatoire mesure la régularité des valeurs de fitness entre les voisins.
Une marche aléatoire de 10 000 solutions a été exécutée afin de calculer les valeurs de l'autocorrélation des valeurs de fitness pour les instances présentées ci-dessus. Le tableau 3.6 donne la première valeur d'autocorrélation pour $N = 64$. Nous constatons que le paramètre q n'a pas d'influence sur la rugosité du problème. Par contre, cette rugosité décroît avec l'augmentation de la valeur de K. Pour $N = 128$ et $N = 256$, les valeurs sont toujours

supérieures à 93% et ne décroissent que très faiblement. Ceci est dû à la valeur de l'épistasie par rapport à la taille du problème. En effet pour une même valeur de K, plus N est grand et plus la régularité du paysage est conservée car la proportion de bits impactés par une variation diminue.

Tableau 3.6 – Première valeur de l'autocorrélation des valeurs de fitness entre voisins des paysages-NKq, $N = 64$.

K	2	4	6	8
q =2	0.95	0.92	0.89	0.85
3	0.94	0.91	0.89	0.86
4	0.94	0.91	0.89	0.85

3.3.2 Degrés de neutralité

Le degré de neutralité moyen d'une instance est calculé en moyennant le nombre de voisins neutres de solutions issues d'une population aléatoire de l'espace de recherche. Le degré de neutralité moyen des optima locaux est calculé à partir d'une population de solutions d'optima locaux. Étant donné que le temps d'exécution est rapide pour ces instances, nous nous permettons de calculer, pour chaque instance, ces degrés de neutralité avec une population initiale de 100 000 solutions. La taille de la population est suffisamment grande pour bien représenter l'instance et il a été vérifié que chaque degré de neutralité est dans un intervalle de confiance de 99%. L'interprétation du degré de neutralité moyen est dépendante de la taille du voisinage, ici la taille d'une solution. Le *ratio du degré de neutralité* correspond au pourcentage du rapport entre le degré de neutralité et la taille du voisinage d'une solution. Ce ratio permet alors de comparer les instances de tailles différentes.

Tableau 3.7 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage de solutions choisies aléatoirement pour les paysages-NKq, $N = 64$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	21 - 32%	17 - 26%	14 - 22%	12 - 17%
3	13 - 20%	10 - 16%	9 - 14%	8 - 12%
4	9 - 14%	7 - 12%	6 - 9%	6 - 9%

Le paramètre q définissant le nombre de valeurs différentes des tables de contribution, la neutralité est d'autant plus grande que q est petit. Le paramètre K règlant la rugosité du problème, il est évident que des solutions voisines deviennent de plus en plus différentes quand K augmente car il est alors moins facile de garder la même valeur de fitness. Les tableaux 3.7 et 3.8 présentent le degré de neutralité moyen pour chaque instance et les

Tableau 3.8 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage des optima locaux pour les paysages-NKq, $N = 64$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	16 - 25%	9 - 14%	6 - 9%	4 - 6%
3	10 - 15%	5 - 8%	3 - 5%	2 - 3%
4	7 - 10%	4 - 6%	3 - 5%	2 - 3%

ratios moyens des degrés de neutralité pour $N = 64$. Dans l'annexe C, ces valeurs sont présentées pour $N = 128$ et $N = 256$. Que ce soit pour les solutions choisies aléatoirement dans l'espace de recherche ou pour les optima locaux, nous constatons que le degré de neutralité diminue quand K et/ou q augmente. De plus, pour chaque instance, le degré de neutralité moyen est plus élevé pour des solutions choisies aléatoirement que pour les optima locaux. Cependant, quelque soit la taille du problème (N), les ratios des degrés de neutralité sont identiques pour les mêmes valeurs de K et de q. Le paramètre q est lié aux tables des contributions. La valeur de fitness d'une solution est la somme de la contribution de chaque bit. La taille du problème intervient donc proportionnellement dans la neutralité du problème ce qui peut expliquer que, pour des tailles différentes mais une valeur de q identique, les instances ont des ratios de degré de neutralité équivalents.
L'étude de la rugosité et de la neutralité des instances des paysages-NKq a montré que l'épistasie du problème joue sur la régularité des valeurs de fitness du paysage entre les différentes tailles de problème mais conduit néanmoins à des ratios similaires du degré de neutralité. Cette étude préliminaire incite à analyser la structure des plateaux suivant les paramètres q et K car leur impact sur le paysage est lié pour certaines caractéristiques (degré de neutralité) mais distincts pour d'autres (régularité des valeurs de fitness dans le voisinage).

3.3.3 Structure du paysage neutre

Comme présenté dans la première section de ce chapitre, l'étude de la structure d'un paysage neutre se fait en étudiant les caractéristiques des plateaux. 30 marches neutres sont effectuées avec une longueur P_{max} égale à 1000 ce qui est suffisant compte tenu de la longueur de marche moyenne d'une méthode de descente pour trouver un optimum local. Pour chaque solution d'une marche, tout son voisinage est évalué dans le but de connaître son nombre de voisins améliorants et neutres. Chaque instance est échantillonnée par 30 de ses plateaux.

Autocorrélation du degré de neutralité L'autocorrélation du degré de neutralité au long d'une marche neutre sur un plateau mesure le caractère non aléatoire du réseau selon cette propriété. La figure 3.8 présente la moyenne sur 30 marches neutres de la première valeur de l'autocorrélation du degré de neutralité pour $N = 64$ et $N = 256$ (pour $N = 128$, se référer à l'annexe C). Pour toutes les instances, la première valeur de l'autocorrélation du modèle nul est dans l'intervalle $[-0.01; 0.01]$. Ceci prouve que les

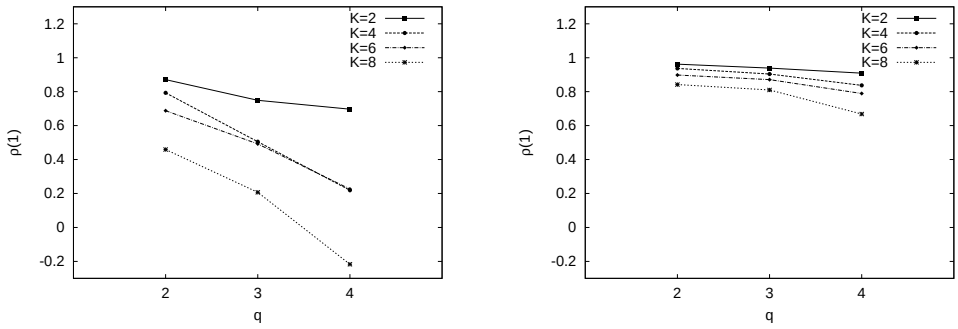


Figure 3.8 – Première valeur de l'autocorrélation $\rho(1)$ du degré de neutralité entre voisins d'un plateau pour les paysages-NKq avec N=64 (a) et N=256 (b).

corrélations des degrés de neutralité entre les voisins d'un même plateau doivent être prises en compte. Pour les paysages-NKq, les degrés de neutralité sont équivalents contrairement à la rugosité liée aux valeurs de fitness. L'autocorrélation établit une passerelle entre ces deux résultats. En effet, moins le paysage est neutre (q augmente) et plus il est rugueux (K augmente), moins la corrélation est forte entre le nombre de voisins neutres de deux solutions voisines de même qualité. Sur les figures, ceci est visible par l'impression de voir une diagonale descendante. De plus, pour $N = 64$ et $N = 128$, pour $K = 8$ et $q = 4$, les degrés de neutralité entre voisins sont anti ou non corrélés, ce qui implique que le plateau a une structure aléatoire qui va être plus difficilement exploitable sans des connaissances supplémentaires. Quand N augmente cette diagonale est de plus en plus horizontale et proche de 1. L'épistasie joue sur la rugosité du paysage et sur la régularité du degré de neutralité sur les plateaux. Dans notre étude, nous avons choisi de limiter les valeurs de K à 8 pour étudier les effets de ce paramètre sur la neutralité. Ici, plus le rapport entre l'épistasie (K) et la taille du problème (N) est grand, plus la rugosité va impacter le paysage et donc la structure neutre du problème. Dans la suite de ce chapitre nous nous contentons du cas $N = 64$ car il présente des structures plus différentes ; il est donc plus intéressant pour évaluer les performances des métaheuristiques sur ces types de problèmes. Les autres résultats de l'étude de la structure des plateaux pour $N = 128$ et $N = 256$ sont tout de même donnés à titre indicatif dans l'annexe C.

Échantillonnage des plateaux Afin de vérifier que les solutions rencontrées lors de la marche neutre soient différentes, la figure 3.9 (a) montre le pourcentage moyen de solutions rencontrées au moins deux fois par les 30 marches neutres utilisées pour caractériser le paysage de chaque instance. Ce pourcentage augmente avec K et q. Par exemple, pour $K = 8$ et $q = 4$, en moyenne un tiers des solutions rencontrées sur les différents plateaux sont rencontrées une nouvelle fois par la marche neutre. Par contre, pour $q = 2$ et $K = 2, 4, 6$, aucune des 1000 solutions ne sont identiques : les plateaux des paysages de ces instances sont donc très grands. Pour ces instances des problèmes de paysages-NKq, les paysages neutres et peu rugueux présentent des plateaux larges alors que les paysages

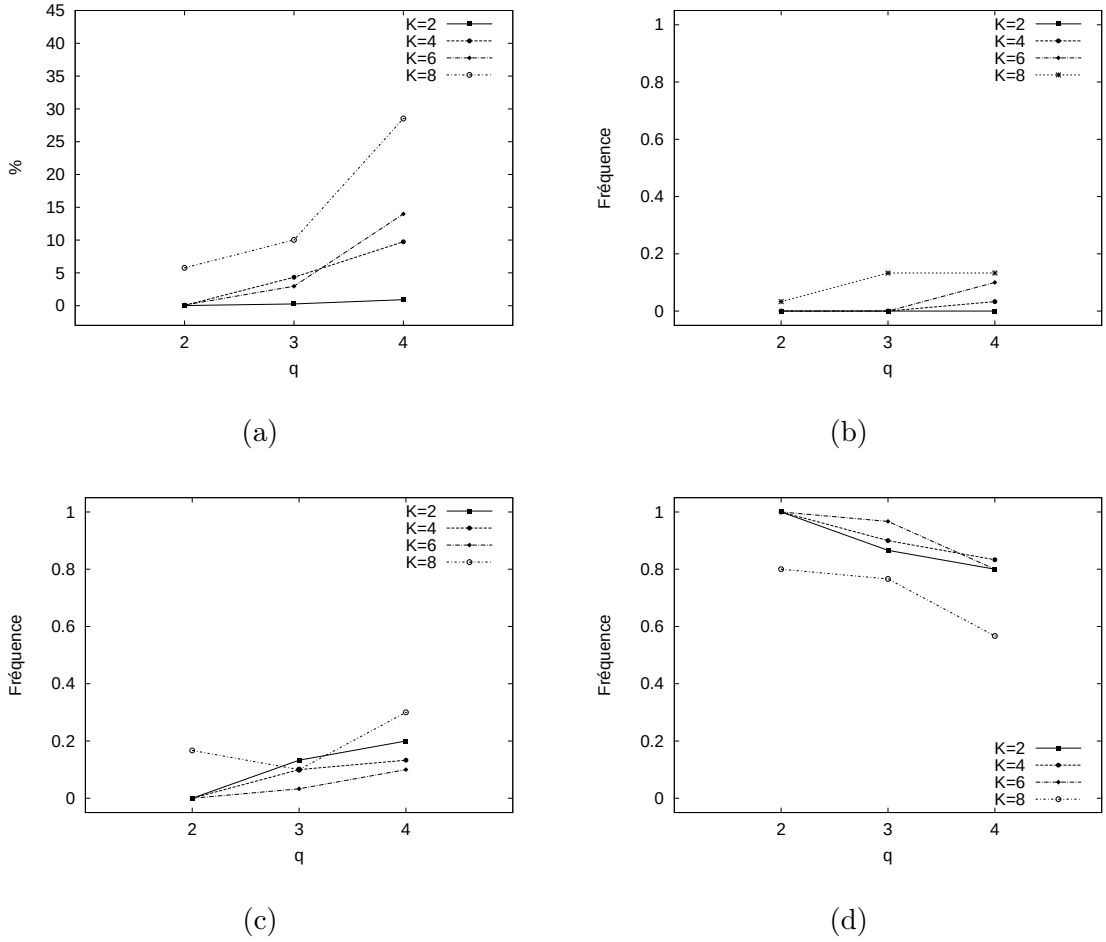


Figure 3.9 – Pourcentage moyen de solutions visitées au moins deux fois par les 30 marches neutres (a) et fréquences moyennes des types de plateaux rencontrés T1 (b), T2 (c) et T3 (d) pour les paysages-NKq ($N = 64$).

rugueux ayant un faible degré de neutralité ont des plateaux plus petits.
Ces résultats sur les marches neutres et les solutions qui les composent montrent que les plateaux peuvent être très denses. Lors de la résolution de tels problèmes, cette propriété de structure doit être prise en considération.

Typologies des plateaux Nous avons défini trois typologies de plateaux : (T1) les plateaux dégénérés constitués d'un unique optimum local, (T2) les plateaux semblant être constitués uniquement d'optima locaux et (T3) les plateaux possédant au moins une porte. Ce qui nous intéresse ici c'est de connaître la proportion de chacun de ces types dans l'ensemble des plateaux échantillonnés. La figure 3.9 (b) montre la fréquence des plateaux de type T1, (c) la fréquence des plateaux de type T2 et (d) la fréquence des plateaux de type T3. Nous constatons que sur les paysages très neutres avec peu de rugosité ($q = 2$

Tableau 3.9 – T3 donne le nombre de plateaux de type T3. P_{max} donne la longueur des marches neutres. nbS donne le nombre de solutions nécessaires à visiter pour rencontrer une porte en se déplaçant sur le plateau. LgM donne la longueur de marche moyenne pour trouver les optima locaux. Toutes ces valeurs sont données pour des instances des paysages-NKq.

Instances		T3	P_{max}	nbS				LgM			
		nb	nb	Min	Med	Moy	Max	Min	Med	Moy	Max
q=2	K=2	30	10^4	2	**4**	5.9	30	4	**17**	17.5	32
	K=4	30	10^4	2	**3**	5.1	28	3	**16**	16.4	31
	K=6	30	10^4	2	**3**	7.3	50	3	**15**	15.3	31
	K=8	24	10^4	2	**2**	5.2	38	2	**15**	14.6	29
q=3	K=2	26	10^4	2	**4**	12.3	108	6	**22**	22	42
	K=4	27	10^4	2	**2**	3.9	32	5	**21**	21	43
	K=6	29	10^4	2	**3**	7.3	50	4	**19**	19	40
	K=8	23	10^4	2	**2**	2.8	9	2	**18**	18.1	38
q=4	K=2	24	10^4	2	**4**	9.5	100	7	**25**	24.7	45
	K=4	25	10^4	2	**3**	3	8	6	**23**	23.3	47
	K=6	24	10^4	2	**2**	5.1	61	5	**21**	21.3	42
	K=8	17	10^4	2	**2**	4.2	15	3	**20**	20	44

et $K = 2, 4, 6$), les marches neutres ont rencontré au moins une porte. Avec ce type de paysage, il est quasiment certain qu'une recherche locale exploitant la neutralité sera capable même en se déplaçant aléatoirement de trouver une solution améliorante. Pour les instances où $q = 3$ et $K = 2, 4, 6$, aucun plateau dégénéré n'a été trouvé. Les 30 optima locaux, départs des marches neutres, possèdent donc des voisins neutres. Ainsi, une recherche locale trouvant un optimum local peut continuer à se déplacer sur le paysage en acceptant ces solutions de même qualité. Néanmoins pour ces instances, il existe des marches neutres (6 au maximum) qui se sont enfoncées dans le plateau et n'ont pas trouvé de portes. Pour les instances où $K = 8$, nous avons vu que les plateaux semblent plutôt petits par rapport aux autres instances. De même, la rugosité est importante et le degré de neutralité entre les voisins n'est pas corrélé. Cependant ces instances possèdent plus de la moitié de leurs plateaux avec des portes. Les déplacements neutres ne sont donc pas à laisser de côté pour ce type d'instance, car certains peuvent conduire à des solutions améliorantes. D'après ces résultats, il peut être intéressant d'exploiter ces plateaux pour se déplacer sur le paysage et tenter de trouver une porte.

Portes sur les réseaux Pour les instances considérées, au minimum la moitié des plateaux visités possèdent au moins une solution porte. Une recherche locale peut donc exploiter cette propriété de la structure de paysage mais il faut s'assurer que ces portes sont rapides à trouver et ainsi ne pas perdre de temps à chercher en vain. La figure 3.9 présente le minimum, la médiane, la moyenne et le maximum du nombre de solutions à

visiter par plateau pour rencontrer une porte pour les marches neutres se déplaçant sur un réseau de type T3. Le nombre de marches neutres utilisées pour ce calcul est indiqué ainsi que le nombre de pas neutres P_{max} effectués. Pour chaque instance, au moins une marche neutre a réussi à trouver une porte en visitant seulement deux solutions. Ceci signifie qu'en seulement deux mouvements une solution améliorante peut être trouvée. La médiane et la moyenne sont précisées pour avoir une idée de la répartition des valeurs trouvées. En moyenne, les marches neutres trouvent très rapidement des portes que ce soit pour des instances très neutres ou non ou des instances rugueuses ou non. Il faut donc étudier le compromis entre le nombre de solutions à visiter sur le plateau pour rencontrer une porte et le nombre de pas à effectuer le long d'une méthode de descente pour trouver un optimum local.

Compromis coût/qualité Afin d'étudier le compromis coût/qualité nous comparons le nombre de solutions à visiter pour qu'une marche neutre rencontre une porte et le nombre de pas nécessaires à une méthode de descente pour trouver un optimum local *i.e.* la longueur de marche. Le tableau 3.9 présente, en regard du nombre de solutions du plateau à visiter pour rencontrer une porte, la longueur de marche. Pour toutes les instances considérées, le nombre de pas minimum pour trouver un optimum local avec un First Improvement Hill Climbing est supérieur ou égal au nombre de solutions visitées par plateau pour trouver une porte. Il est donc évident que profiter de la neutralité du paysage pour continuer le processus d'optimisation sans dégrader la solution est bénéfique par rapport à redémarrer la recherche d'une solution choisie aléatoirement dans l'espace de recherche.

3.3.4 Synthèse

Dans cette section, plusieurs instances pour des paysages-NKq ont été traitées. Ce problème est intéressant car il est possible de régler à l'aide des paramètres K et q respectivement, le degré de rugosité et le degré de neutralité de l'instance suivant la taille des solutions N. Dans cette étude, $N \in \{64; 128; 256\}$, $K \in \{2; 4; 6; 8\}$ et $q \in \{2; 3; 4\}$. Plus la valeur de K augmente, plus l'instance est rugueuse. De plus, plus la valeur de q augmente, moins l'instance est neutre. Pour une même valeur de K, l'augmentation de la valeur de N entraîne un moins fort impact de K sur la rugosité du problème. Par contre, à valeurs de K et q identiques, les degrés de neutralité sont identiques quelque soit la valeur de N. Seule la structure de l'instance avec $N = 64$ a été étudiée dans la suite car c'est celle qui présentait le plus de différences selon les valeurs de K et q.
Les conclusions sont les suivantes. Plus l'instance est rugueuse et moins elle est neutre, plus les plateaux sont petits. Qu'importe l'instance, si le plateau possède des portes, celles-ci sont rapidement atteignables. Par contre, le nombre de tels plateaux est dépendant du degré de neutralité. Plus la neutralité est élevée et plus il est facile de trouver un plateau avec des portes.

Cette analyse pour caractériser la structure neutre de deux problèmes d'optimisation combinatoire, le problème de flowshop et les paysages-NKq, incite donc à exploiter la neutralité des optima locaux tout en veillant à ne pas perdre de temps à explorer des plateaux sans

porte au cours du processus d'optimisation.

3.4 Métaheuristiques exploitant la neutralité

Peu de métaheuristiques exploitent explicitement la propriété de neutralité que peut présenter le paysage de certains problèmes d'optimisation. Les recherches locales pour la plupart améliorent la solution courante en se déplaçant vers une solution de qualité strictement meilleure jusqu'à rencontrer un optimum local. Elles abordent alors une stratégie de remplacement de la solution courante en acceptant de dégrader la qualité dans le but de trouver une meilleure solution que le précédent optimum local. Le *Netcrawler*, quant à lui, est une méthode de descente (*cf.* section 1.2.1.1) qui se déplace vers la première solution de qualité meilleure ou équivalente à la solution courante. Cette méthode exploite explicitement la propriété de neutralité d'un problème mais ne fait pas de différence entre les optima locaux ou les autres solutions. Dans cette section, nous proposons une recherche locale itérée qui tient compte de la propriété de neutralité des optima locaux du paysage. Notre but est de concevoir une recherche locale générique qui soit simple à mettre en œuvre, tel un Netcrawler, et qui soit efficace pour les problèmes traités telle une méthode dédiée.

3.4.1 NILS : ILS basée sur la neutralité

Les recherches locales itérées sont des méthodes utilisant efficacement la relation de voisinage pour se déplacer vers des solutions meilleures et une stratégie, appelée perturbation, qui permet de sortir d'un optimum local. L'étude de la neutralité du paysage a montré, pour les problèmes de flowshop et paysages-NKq, que les optima locaux ont en moyenne moins de voisins neutres que les autres solutions. Contrairement au Netcrawler, nous proposons d'utiliser la neutralité du paysage uniquement au niveau des optima locaux. Ainsi, la recherche locale ne tient pas compte des solutions neutres pour se déplacer jusqu'à un optimum local. Puis à partir de ce dernier, l'étape de perturbation exploite la neutralité du voisinage en autorisant des déplacements neutres.

3.4.1.1 Perturbation basée sur une marche neutre

En général, il existe deux façons possibles de s'échapper d'un optimum local quand le paysage présente de la neutralité : soit une marche neutre est exécutée sur le plateau jusqu'à trouver une solution porte soit un saut dans l'espace de recherche est effectué en appliquant successivement un autre opérateur de voisinage, *kick-move* en anglais. Ainsi l'utilisation des mouvements neutres favorise l'*exploitation* des propriétés de neutralité et aide à trouver une meilleure solution ; contrairement au kick-move, qui présuppose que les portes sont rares et difficiles à atteindre et donc, favorise l'*exploration* d'une autre partie de l'espace de recherche plus prometteuse.
La stratégie de perturbation que nous proposons profite du compromis entre exploration et exploitation des plateaux. Cette perturbation basée sur les marches neutres est décrite par l'algorithme 2. La perturbation commence par exécuter à partir de l'optimum local trouvé une marche neutre ayant une longueur maximale. Ce nombre maximal de pas sur le plateau est un paramètre de cette perturbation dénoté par MNS (Maximal number of

Algorithme 2 Perturbation de NILS

```
solution s tel que |V_n(s)| > 0
nbPas ← 0, meilleur ← faux
tant que nbPas < MNS et non meilleur faire
  choisir s' ∈ V(s) tel que f(s') ≤ f(s)
  si f(s') < f(s) alors
    meilleur ← vrai
  fin si
  s ← s'
  nbPas ← nbPas+1
fin tant que
si non meilleur alors
  s ← kick(s)
fin si
```

steps). Au long de la marche neutre, le voisinage de chaque solution rencontrée est évalué dans un ordre aléatoire jusqu'à ce qu'une solution de qualité équivalente ou meilleure soit trouvée pour remplacer la solution courante. Cette étape de perturbation s'arrête dès qu'une solution améliorante est rencontrée ou quand le nombre de pas maximal de déplacement sur le plateau (MNS) est atteint. Dans ce dernier cas, un kick-move sur la solution courante conduit à une nouvelle solution de l'espace de recherche.

3.4.1.2 Description de la méthode

NILS (Neutrality-based Iterated Local Search) est une recherche locale itérée basée sur une méthode de descente et sur la perturbation exploitant la neutralité, décrite dans le paragraphe précédent. Le First Improvement Hill Climbing (FIHC) est la méthode de descente choisie car elle permet de se déplacer plus rapidement vers une solution améliorante puisque le premier meilleur voisin trouvé remplace la solution courante. Le voisinage de la solution est évalué dans un ordre aléatoire pour ne pas favoriser certains mouvements. Comme toute recherche locale itérée, un critère d'acceptation est défini. Tout optimum local trouvé est accepté comme la nouvelle solution courante. La meilleure solution de tous ces optima locaux est conservé en mémoire tout au long de la recherche. NILS itère successivement deux étapes : (i) une étape d'exploitation de la neutralité qui effectue un déplacement sur le plateau, (ii) une étape d'amélioration de la solution. La propriété de neutralité que peut présenter le problème à optimiser est prise en considération lors de la phase (i). Les marches neutres sur les plateaux successifs permettent de traverser de larges parties de l'espace de recherche. Néanmoins, les portes peuvent être difficiles à trouver et leur voisins améliorants peu nombreux. Dans une telle situation, une technique de *kick-move* est utilisée : la solution est sortie de son voisinage en étant déplacée vers une autre partie de l'espace de recherche. La deuxième phase (ii) permet d'améliorer strictement la qualité de la solution courante. Ainsi, FIHC améliore chaque solution jusqu'à trouver un optimum local puis, une marche neutre est exécutée jusqu'à trouver une porte à partir de laquelle la méthode de descente pourra être appliquée. Le compromis entre l'exploration et l'exploitation des plateaux est directement réglé par le seul paramètre de NILS : le

paramètre MNS.

3.4.1.3 Dynamique de NILS

La figure 3.10 présente en parallèle la dynamique de NILS et du Netcrawler (NC) pour atteindre l'optimum global (en rouge, en bas à droite) dans un problème de minimisation. Ces représentations se veulent comme des modèles généraux de la dynamique des méthodes. À partir d'une même solution de l'espace de recherche (en haut à gauche), NILS commence par améliorer strictement la qualité de la solution courante, puis se retrouve sur un optimum local (en vert). Notons que cette première partie correspond à la dynamique du FIHC. Puis, NILS se déplace neutralement sur le plateau jusqu'à trouver une porte (en orange), une solution dont un voisin améliorant a été trouvé. Ensuite, NILS reprend la dynamique du FIHC à partir de ce nouveau voisin et ainsi de suite jusqu'à atteindre l'optimum global. Clairement, nous avons décrit la dynamique principale de NILS en omettant le kick-move qui ne tient pas le rôle le plus significatif. Parallèlement à la dynamique de NILS, la figure 3.10 permet de comprendre celle de NC et d'en noter les principales différences. NC accepte toute solution meilleure ou équivalente. Il se déplace neutralement s'il trouve d'abord un voisin équivalent ou en améliorant dans le cas où il trouve un voisin meilleur. NC ne voit donc pas les optima et stoppe sa recherche seulement s'il est sur l'optimum global ou sur le plateau dégénéré d'un optimum local.

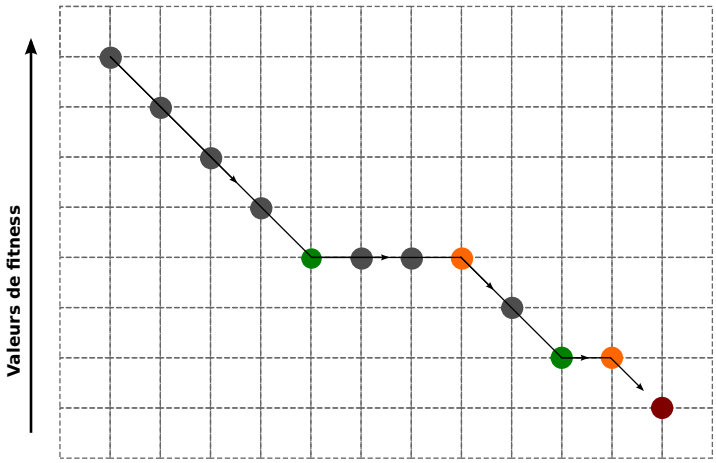
NILS est une métaheuristique qui se veut efficace et facile à mettre en œuvre. Dans la suite de cette section, les performances de NILS sont étudiées sur les deux problèmes neutres : le flowshop et les paysages-NKq. L'analyse de la structure neutre de ces deux problèmes a été menée au cours de ce chapitre et a révélé que pour certaines instances, l'exploration et l'exploitation des plateaux peuvent conduire à des portes. Utiliser NILS pour résoudre ces instances semble donc tout à fait favorable. De plus, pour chacun des problèmes, NILS est au moins comparé au NC afin de prouver d'une part l'intérêt d'utiliser la neutralité, d'autre part l'apport de NILS sur la résolution des problèmes neutres.

3.4.2 NILS sur le problème de flowshop

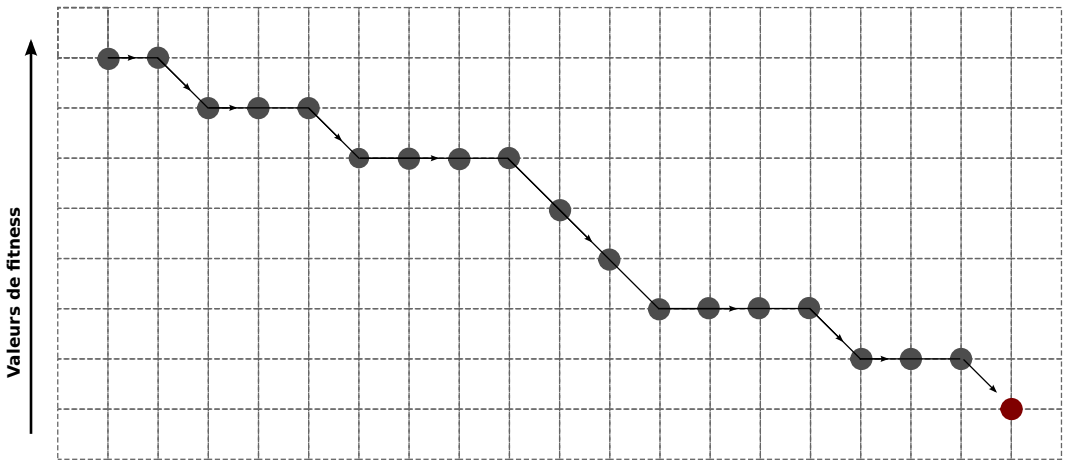
La section 3.2 a montré que les instances du flowshop présentent un paysage neutre. Prenant la relation de voisinage basée sur l'opérateur *insertion*, la neutralité est plus ou moins forte selon le type d'instances : aléatoire ou structurée. C'est donc cette relation de voisinage qui est utilisée par la méthode de descente et la marche neutre de NILS.

3.4.2.1 Design expérimental

Pour commencer, les performances de NILS ont été étudiées sur les instances aléatoires proposées par Taillard car elles sont les plus utilisées de la littérature [Stü98, RS07]. La première instance de Taillard pour chaque taille a été utilisée : le nombre de jobs N appartient à l'ensemble $\{20, 50, 100, 200, 500\}$ et le nombre de machines M à $\{5, 10, 20\}$. Ensuite, NILS a été exécuté sur les instances structurées du Flowshop pour les trois types de corrélation (jobs-correlées, machine-corrélées, jobs/machines-corrélées) et les différents nombres de jobs $\{20, 50, 100, 200\}$.



NILS



NC

Figure 3.10 – Dynamiques de NILS et de NC pour un problème de minimisation. À partir d'une solution de l'espace de recherche (en haut à gauche), la trajectoire des mouvements sur le paysage est explicitée jusqu'à atteindre l'optimum global (en rouge, en bas à droite). Les optima locaux de NILS sont de couleur verte et les portes sur les plateaux de couleur orange.

Le degré de neutralité étant relatif à la taille du voisinage, des expérimentations préliminaires ont montré que le paramètre MNS devait être adapté à chaque taille. Pour plus de clarté dans la présentation des résultats, nous présentons pour chaque nombre de jobs, les 5 valeurs de MNS issues d'un pré-réglage qui nous paraissent les plus significatives pour analyser les performances et l'influence de ce paramètre :
– pour $N = 20$, $MNS \in \{0, 10, 20, 50, 100\}$;
– pour $N = 50$, $MNS \in \{0, 50, 150, 900, 1800\}$;
– pour $N = 100$, $MNS \in \{0, 100, 5000, 20000, 40000\}$;
– pour $N = 200$, $MNS \in \{0, 200, 4000, 30000, 60000\}$;
– pour $N = 500$, $MNS \in \{0, 5000, 15000, 80000, 150000\}$.
Pour étudier ses performances, NILS a été exécuté 30 fois pour chaque taille et respectivement chacune des valeurs de MNS proposées. Remarquons que $MNS = 0$ correspond à une recherche locale itérée (ILS) sans que la neutralité soit considérée. Ceci permet donc de comparer NILS avec une ILS plus classique. La distribution des 30 valeurs de fitness trouvées pour chaque taille et valeur de MNS n'est pas symétrique. Sa moyenne et son écart-type étant alors statistiquement dénués de sens, les quartiles, dont la médiane, des 30 valeurs de fitness sont présentées pour chaque taille.
Dans le but de laisser chaque configuration de NILS converger, la condition d'arrêt correspond à un nombre d'évaluations maximal, égal à 2.10^7.
Dans l'étape de perturbation, un kick-move est utilisé quand aucune meilleure solution n'a été trouvée avant le nombre maximal de pas neutres atteint. Ruiz *et al.* [RM05] ont montré que le kick-move dépendait des instances. Cependant, notre travail n'a pas pour but de trouver les meilleurs paramétrages pour le kick-move, nous avons choisi une valeur qui donne des performances raisonnablement bonnes pour toutes les instances. Le kick-move correspond dans nos expérimentations à trois fois l'application de l'opérateur de voisinage *échange* (*cf.* section). Cet opérateur permet de casser le paysage vu par NILS et se déplacer dans une nouvelle partie de l'espace de recherche.

3.4.2.2 Instances aléatoires de Taillard

3.4.2.2.1 Résultats expérimentaux Dans cette section, nous présentons les performances de NILS sur les instances de Taillard pour le problème de flowshop de permutation et l'intérêt d'intégrer la neutralité dans le processus de recherche quand le paysage est neutre. Notons, NILS$_x$, la configuration de NILS telle que le paramètre $MNS = x$. Nous comparons les performances obtenues avec $MNS = 0$ et ceux avec $MNS > 0$. Le tableau 3.10 présente pour chaque taille de problème ($M \times N$), la meilleure valeur de fitness trouvée quelque soit la valeur du paramètre MNS en comparaison à la meilleure valeur connue de la littérature [1]. De plus, la déviation entre la meilleure valeur de NILS et le best-known divisée par celui-ci est précisée en pourcentage. Rappelons que le flowshop est un problème de minimisation. De plus, il est important de souligner que les best-known correspondent à des solutions généralement trouvées par des méthodes exactes ou des métaheuristiques différentes selon la taille et l'instance considérées. Pour les instances 5- machines ou 20-jobs et l'instance 100- jobs 20- machines, le best-known de la littérature est atteint par au moins une configuration de NILS testées. Pour les plus faciles d'entre elles, toutes les

1. `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html`

Tableau 3.10 – Meilleures performances de NILS$_*$ pour les instances de Taillard selon les nombres de jobs (N) et de machines (M). Les valeurs en gras indiquent que le best-known est atteint par au moins une configuration de NILS. Δ représente en pourcentage les déviations non nulles entre les valeurs de NILS et les best-known.

Instances	M	N	Best-known	NILS$_*$	$\Delta_{(\%)}$
20/ 5/01	5	20	1278	**1278**	-
50/ 5/01		50	2724	**2724**	-
100/ 5/01		100	5493	**5493**	-
20/10/01	10	20	1582	**1582**	-
50/10/01		50	2991	3025	1.14
100/10/01		100	5770	**5770**	-
200/10/01		200	10862	10872	0.09
20/20/01	20	20	2297	**2297**	-
50/20/01		50	3847	3894	1.22
100/20/01		100	6202	6271	1.11
200/20/01		200	11181	11303	1.08
500/20/01		500	26059	26235	0.68

configurations testées le trouve au moins une fois. Ceci soutient qu'utiliser les marches neutres conduit à des performances au moins aussi bonnes que celles d'un modèle de recherche locale itérée classique. Ces instances étant très difficiles, le best-known n'est que rarement trouvé par une métaheuristique générique. Ainsi, les performances pour ces instances semblent assez loin du best-known. Mais, au regard de la déviation au best-known, elles sont en réalité très correctes.

Le tableau 3.11 présente les médianes des 30 valeurs trouvées par NILS$_0$ et de celles trouvées par NILS$_{max}$ (valeur maximale de MNS). Pour les instances 5- machines ou 20- jobs, le best-known est atteint par au moins la moitié des 30 exécutions de NILS quelque soit la configuration testée. Pour les autres instances, les médianes sont meilleures pour NILS$_{max}$ que pour NILS$_0$. Par exemple, pour les instances 20- machines (excepté 20-jobs), les performances de NILS$_{max}$ dépassent clairement celles de NILS$_0$. Les meilleures performances sont donc atteintes en utilisant des marches neutres sur les plateaux existants du paysage. Ceci valide bien l'intérêt d'utiliser la propriété de neutralité d'un tel paysage.

La comparaison de ces performances avec le degré de neutralité des instances permet de donner des conclusions additionnelles. Pour les instances 20- machines, les degrés de neutralité des optima locaux sont assez bas conduisant à des ratios très faibles (autour de 5%). Cependant, les performances de NILS sont très bonnes dans cette situation ce qui peut être surprenant à première vue. En fait, pour ces instances, le degré de neutralité moyen est suffisant pour qu'une marche aléatoire neutre traverse une large partie plate de l'espace de recherche, les plateaux. De plus, les instances présentant un ratio de degré de neutralité moyen assez élevé voient leur densité de porte sur le plateau diminuée et ainsi une recherche purement aléatoire sur le plateau est assez inefficace pour trouver

Tableau 3.11 – Performances de NILS pour les instances de Taillard selon les nombres de jobs (N) et de machines (M). NILS_0 donne la médiane des 30 valeurs trouvées pour $MNS = 0$. NILS_{max} donne la médiane des 30 valeurs trouvées pour la valeur maximale de MNS. La meilleure médiane parmi NILS_0 et NILS_{max} est en caractère gras.

Instances	M	N	NILS_0	NILS_{max}
20/ 5/01	5	20	**1278**	**1278**
50/ 5/01		50	**2724**	**2724**
100/ 5/01		100	**5493**	**5493**
20/10/01	10	20	**1582**	**1582**
50/10/01		50	3042	**3025**
100/10/01		100	5784	**5771**
200/10/01		200	10885	**10872**
20/20/01	20	20	**2297**	**2297**
50/20/01		50	3925	**3917**
100/20/01		100	6375.5	**6315.5**
200/20/01		200	11405.5	**11334.5**
500/20/01		500	26401.5	**26335.5**

rapidement une porte.

3.4.2.2.2 Influence du paramètre MNS Puisque les marches neutres sur les plateaux *a priori* d'optima locaux semblent efficaces pour trouver des portes *i.e.* des solutions améliorantes lors de l'évaluation du voisinage de celle-ci, nous nous demandons si la longueur de ces marches, réglée par le paramètre MNS, doit être nécessairement grande. Nous souhaitons étudier les performances des configurations de NILS avec les valeurs de MNS données lors de la description de notre design expérimental. Il est évident que la valeur du paramètre MNS dépend de la taille du voisinage de la solution qui est fortement liée à la taille de la solution (le nombre de jobs). La figure 3.11 présente les performances trouvées pour les exécutions de NILS pour les instances 50-, 100-, 200- jobs et 10-, 20- machines. Les résultats pour les autres instances sont données en annexe D.
Pour les instances 10- machines (haut), la médiane des valeurs de fitness commence par diminuer quand la valeur de MNS augmente, puis se stabilise : les résultats deviennent équivalent pour des valeurs de MNS élevées. Puisque la distribution des performances n'est pas normale, nous avons utilisé le test statistique de Mann-Whitney qui teste l'hypothèse (H_0) d'égalité de deux échantillons. Ce test a donc validé que les performances étaient comparables et identiques pour des configurations de NILS avec des valeurs de MNS élevées. Il est donc possible d'affirmer qu'augmenter la valeur du paramètre MNS dans ces cas ne détériore pas les performances de NILS. Par conséquent, ce paramètre peut être réglé à de très grandes valeurs.
Pour les instances 20- machines (bas), la meilleure des performances trouvées pour chacune vient d'une configuration de NILS qui exploite les plateaux. Pour les instances 200-, 500-

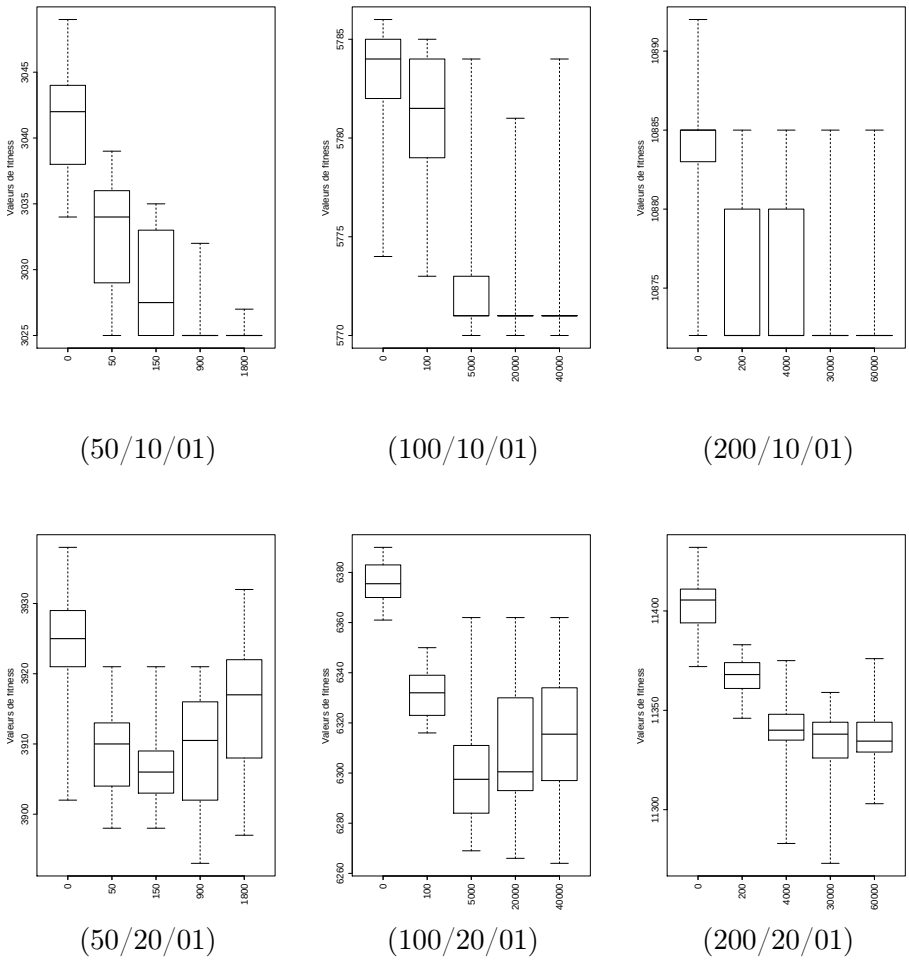
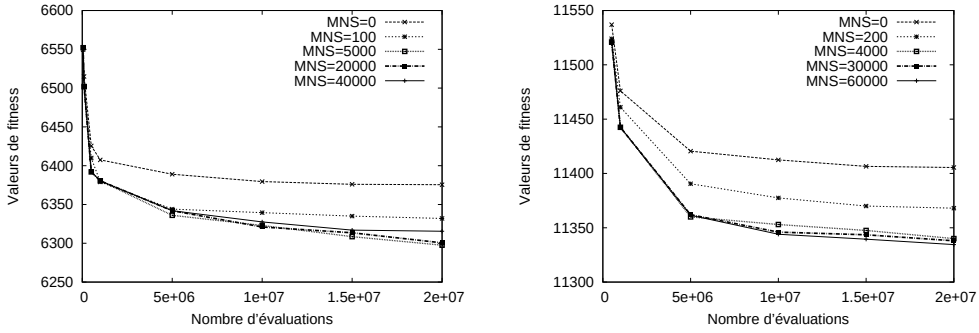


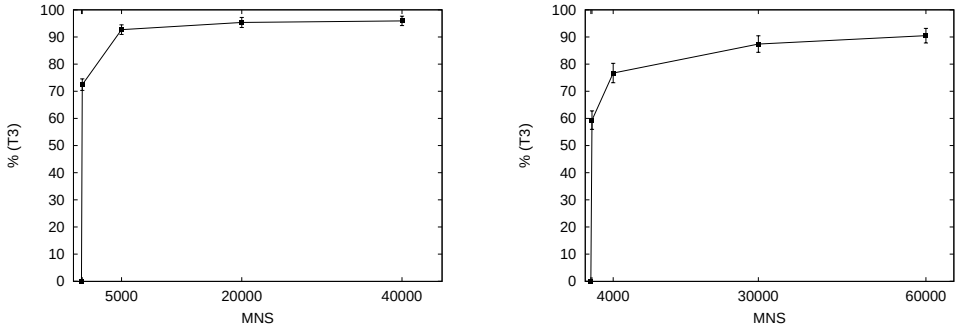
Figure 3.11 – Boîtes à moustaches des 30 valeurs de fitness trouvées après 2.10^7 évaluations pour différentes valeurs de MNS, pour les instances de Taillard, de haut en bas, 10- et 20- machines et de gauche à droite, 50-, 100- et 200- jobs.

jobs (*cf.* annexe D), les conclusions sont les mêmes que les précédentes pour 10- machines. Néanmoins, pour les instances 50- et 100- jobs, les médianes des performances diminuent quand les valeurs de MNS deviennent trop élevées.

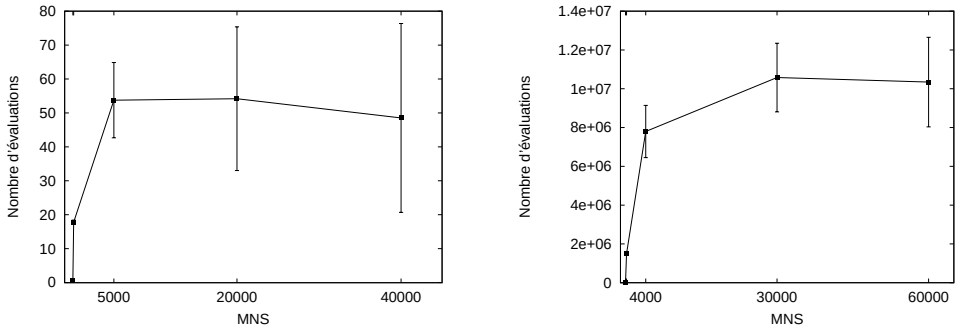
L'allure des boîtes à moustaches donnent l'impression qu'il existe une valeur optimale pour le paramètre MNS. Ceci est confirmé par le test de Mann-Whitney qui montre que les performances de NILS_{max} sont moins bonnes que celles avec une valeur de MNS plus petite. Par conséquent, dans certains cas, il apparaît que la sur-exploitation de la neutralité liée à des déplacements trop longs sur le même plateau peut s'avérer coûteux. Ici, nous comprenons que le compromis entre l'exploitation de la neutralité et l'exploration de l'espace de recherche doit être considéré scrupuleusement étant donné qu'il existerait une valeur *optimale* pour le paramètre MNS.



(a)



(b)



(c)

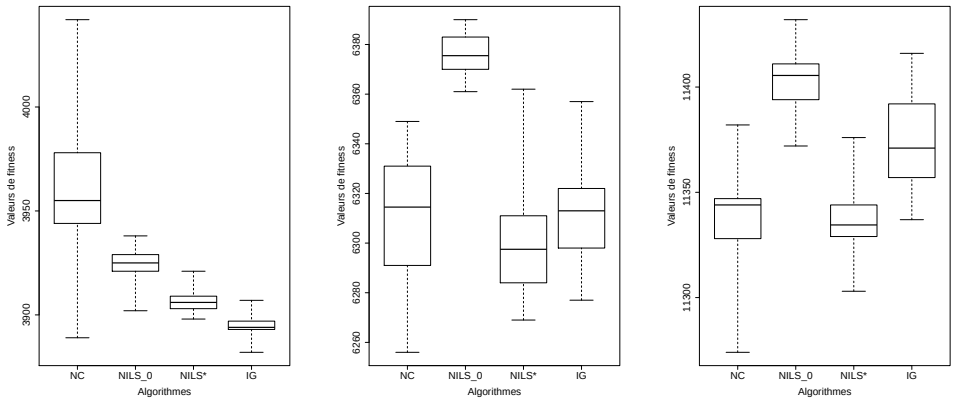
Figure 3.12 – Résultats pour les instances 100/20/01 (gauche) et 200/20/01 (droite). Médiane des performances en fonction du nombre d'évaluations (a). Pourcentage moyen (et son écart-type) du nombre de plateaux où la marche neutre a trouvé une porte (b). Nombre d'évaluations perdues (moyenne et écart-type) où la marche neutre n'a pas trouvé de portes en fonction de la valeur de MNS (c).

3.4.2.2.3 Bénéfice *vs.* coût des marches neutres Nous venons de voir que, pour les instances 20- machines, l'analyse des performances des différentes configurations de NILS en fonction de la valeur du paramètre MNS amène à deux conclusions. Les per-

formances peuvent se trouver dégradées quand la valeur de MNS est trop élevée ou au contraire, elles peuvent stagner. Notons, que pour ce dernier cas, nous avons testé des valeurs de MNS plus grandes, et nous avons constaté que les performances restaient quasi identiques. La figure 3.12 (a) présente l'évolution de la médiane des performances en fonction du nombre d'évaluations pour les instances 100-, 200- jobs 20- machines et confirme ces dernières remarques. De plus, pour étudier le compromis entre l'exploitation des plateaux et l'exploration de l'espace de recherche, nous avons représenté dans la figure 3.12 (b) et (c) en fonction des valeurs du paramètre MNS respectivement le pourcentage de plateaux qui ont permis de trouver une porte (type T3) et le nombre total d'évaluations qui ont été utilisées par les marches neutres n'ayant pas trouvé de portes. Dans les deux cas, le pourcentage de plateaux de type T3 visités augmente mais de plus en plus faiblement. De même, le nombre moyen d'évaluations perdues a tendance à diminuer mais son écart-type est de plus en plus élevé, ce qui montre que le processus de recherche devient de plus en plus instable. Ces remarques confirment qu'avoir une marche neutre très longue est très intéressant en terme de performances mais qu'il peut être nécessaire d'utiliser la perturbation pour éviter de perdre trop d'évaluations pendant les déplacements sur les plateaux.

3.4.2.2.4 Comparaison avec la littérature Le problème de flowshop de permutation est un problème classique de la littérature. Pour les instances de Taillard, Ruiz *et al.* [RS07] ont proposé une recherche locale gloutonne itérée (Iterated Greedy local search, IG) spécifique au problème et l'ont comparée aux métaheuristiques classiques de la littérature. Ils ont aussi entre autre trouvé de nouvelles meilleures solutions pour quelques instances. IG repose sur l'heuristique NEH [NEH83] qui est une procédure de construction et déconstruction de la solution. Elle consiste à supprimer n jobs de la solution et à les réinsérer un à un de façon à minimiser à chaque insertion la date d'achèvement de la dernière tâche ordonnancée. Cette heuristique est ainsi utilisée dans la phase de perturbation de leur ILS et inspire la recherche locale gloutonne qu'ils utilisent. Cette méthode est complètement dédiée au flowshop et présente quelques subtilités algorithmiques telles que la structure de données utilisée lors de la procédure NEH [Tai90].
Néanmoins, nous n'avons trouvé dans la littérature que peu de travaux sur la résolution du problème de flowshop où les méthodes de résolution tiennent compte explicitement de la structure du problème. Citons par exemple les travaux de Jin *et al.* [JSW09] qui proposent une recherche tabou conçue pour bénéficier des propriétés de la structure du problème. La comparaison avec une recherche tabou classique sur les instances structurées de la littérature montre qu'il est avantageux de tirer partie de la structure inhérente du problème. Au début de ce chapitre, nous avons montré que le problème du flowshop présente un paysage ayant une propriété de neutralité. L'analyse de la structure du paysage neutre, nous a amené à conclure qu'utiliser cette propriété de neutralité pouvait donner de bonnes performances. Le Netcrawler [Bar01] est une méthode de descente acceptant de se déplacer vers un voisin de même qualité. Telle une méthode de descente, cette recherche locale est sans difficulté à implémenter et à mettre en œuvre.

NILS est une métaheuristique simple à utiliser et donne des résultats intéressants. Dans tout travail de conception, il est important de se comparer aux méthodes existantes. Nous



(50/20/01) (100/20/01) (200/20/01)

Figure 3.13 – Comparaison des performances du Netcrawler (NC), de NILS sans neutralité (NILS_0), de NILS avec le meilleur réglage testé de MNS (NILS_*) et de la recherche locale gloutonne itérée (IG) pour les instances de Taillard, de gauche à droite, 50-, 100-, 200- jobs 20- machines.

proposons ici de comparer les performances de NILS avec une méthode dédiée telle la méthode IG et une méthode très générique tel le Netcrawler (NC). Les instances de la littérature les plus difficiles à résoudre étant celles à 20 machines, nous nous contentons de comparer leurs performances sur les instances 50-, 100-, 200- jobs 20- machines de Taillard. Ces instances présentent des ratio de neutralité peu élevés ce qui n'influencent pas *a priori* les méthodes exploitant les propriétés de neutralité. L'intérêt d'exploiter la neutralité est montré par la comparaison des performances avec NILS_0 qui correspond à une recherche locale classique qui perturbe la solution et l'envoie dans une autre partie de l'espace de recherche dès qu'un optimum local est trouvé. Pour chaque instance, nous avons retenu le meilleur réglage du paramètre MNS, dénoté par NILS_*.

Ces quatre méthodes ont été exécutées 30 fois sur les trois instances considérées pour cette étude comparative. La figure 3.13 présente les performances obtenues par chacun des algorithmes sur les 30 exécutions réalisées.

La figure montre qu'il n'est pas possible de conclure globalement. En effet, suivant la taille du problème, nous n'observons pas les meilleures performances pour la même méthode. En revanche, NC présente toujours au moins une des meilleures performances mais a des performances très inégales visibles par l'étendue des boîtes à moustaches. Pour l'instance 50-jobs, NC montre qu'une sur-exploitation de la neutralité peut conduire à des performances très irrégulières et qu'il faut donc modérer son utilisation. De plus sur cette instance, ses performances sont en moyenne moins bonnes qu'une ILS classique (de type NILS_0). Néanmoins, NILS_* présente des performances concurrentes à une méthode dédiée telle que IG même si ses performances sont un peu en dessous. Pour les instances 100-, 200- jobs, il est utile de considérer la neutralité car elle permet d'obtenir de meilleurs résultats que NILS_0 et même identiques voire meilleurs que IG.

Cette étude comparative montre que les méthodes dédiées (IG) peuvent trouver de bons

Tableau 3.12 – Meilleures performances de NILS$_*$ pour les instances structurées. Le best-known de la littérature est présenté ainsi qu'en indice sa borne inférieure correspondante. La valeur en gras précise que NILS a trouvé un nouveau best-known. En indice, le nombre de fois où le best-known a été trouvé est indiqué.

Instances	Best-known	NILS$_*$
2020jc	2714_{2712}	2714_{30}
5020jc	4613_{4613}	4613_{30}
10020jc	6959_{6959}	6959_{30}
20020jc	11237_{11216}	$\mathbf{\underline{11223}}_{2}$
2020mc	2536_{2527}	2536_{30}
5020mc	5237_{5237}	5237_{30}
10020mc	10840_{10837}	10840_{30}
20020mc	20418_{20418}	20418_{30}
2020mxc	2518_{2518}	2518_{30}
5020mxc	4802_{4799}	4802_{30}
10020mxc	10342_{10342}	10342_{30}
20020mxc	19435_{19435}	19435_{30}

résultats mais qu'elles peuvent avoir néanmoins des difficultés à avoir des performances robustes. De plus, il faut faire attention à bien exploiter la neutralité car NC, par exemple, même s'il trouve de bonnes solutions, présente lui aussi des performances de qualité assez diverses. NILS$_*$ présente aussi un intervalle assez important entre sa meilleure et sa pire performances, mais la moitié des performances des solutions se situe dans un plus petit intervalle comparé à NC. En conclusion sur ces instances aléatoires, nous pouvons dire que NILS est une méthode qui tient compte de la propriété de neutralité d'un problème et représente un bon compromis entre une méthode (sur-)exploitant la neutralité pour laquelle aucun paramètre supplémentaire n'est nécessaire et une méthode dédiée qui est adaptée au problème à résoudre et donc plus difficile à concevoir.

3.4.2.3 Instances structurées

Les instances structurées sont des instances qui peuvent présenter un très fort degré de neutralité. Pour les instances que nous considérons ($\alpha = 0.8$), le degré de neutralité peut atteindre jusqu'à 95% pour les instances machines-corrélées et jobs/machines-corrélées. NILS étant une recherche locale exploitant la neutralité, il est intéressant d'étudier ses performances sur ce type de structure neutre. Notons toujours NILS$_x$, la configuration de NILS telle que le paramètre $MNS = x$. Le tableau 3.12 présente pour chaque instance structurée, la meilleure valeur de fitness trouvée quelque soit la valeur du paramètre MNS en comparaison du best-known de la littérature [2] (à titre indicatif, la borne inférieure est aussi donnée). Pour l'instance 20020jc, une des configurations de NILS trouve un nouveau

2. `http://www.cs.colostate.edu/sched/generator`

Tableau 3.13 – Performances de NILS pour les instances structurées. NILS_0 donne la médiane des 30 valeurs trouvées pour $MNS = 0$. NILS_{max} donne la médiane des 30 valeurs trouvées pour la valeur maximale de MNS. La meilleure médiane parmi NILS_0 et NILS_{max} est en caractère gras. En indice, le nombre de fois où le best-known a été trouvé est indiqué.

Instances	NILS_0	NILS_{max}
2020jc	2714_{30}	2714_{30}
5020jc	4613_{30}	4613_{30}
10020jc	6960_{4}	$\mathbf{6959}_{30}$
20020jc	11249_{0}	$\mathbf{11225.5}_{2}$
2020mc	2536_{30}	2536_{30}
5020mc	5237_{30}	5237_{30}
10020mc	10840_{30}	10840_{30}
20020mc	20418_{30}	20418_{30}
2020mxc	2518_{30}	2518_{30}
5020mxc	4802_{24}	4802_{30}
10020mxc	10342_{30}	10342_{30}
20020mxc	19435_{30}	19435_{30}

best-known. Sa valeur de fitness est même trouvée deux fois parmi toutes les exécutions. Clairement, le best-known est très facile à atteindre car pour les autres instances, il existe au moins une configuration de NILS qui trouve à chaque fois le best-known. L'analyse de la structure des plateaux (*cf.* section 3.2.2.3) ayant montré que pour la plupart de ces instances, les plateaux des optima globaux étaient facilement accessibles par une simple méthode de descente ce résultat n'est pas des plus surprenant. Néanmoins, le peu de fois où la méthode de descente ne trouve pas le meilleur plateau, l'exploitation du plateau ou le kick-move permettent de l'atteindre. En effet, l'étude des marches neutres a révélé que des portes pouvaient être trouvées en se déplaçant sur les plateaux des optima locaux. Sinon, les descentes successives et rapides à partir de toute solution de l'espace de recherche sont d'autant de tirages qui conduisent directement au best-known.

Comparons les performances obtenues avec $MNS = 0$ et ceux avec la valeur maximale testée de MNS. Le tableau 3.13 présente les médianes des 30 valeurs trouvées par NILS_0 et de celles trouvées par NILS_{max} (valeur maximale de MNS).

Pour l'instance 20020jc, NILS_{max} trouve le nouveau best known deux fois. Pour les autres instances, il trouve pour les 30 exécutions le best-known. La comparaison de NILS_0 et NILS_{max} se concentre sur les instances 10020jc, 20020jc et 5020mxc car pour les autres, ils trouvent tous les deux le best-known pour les 30 exécutions. Ces trois instances sont les seules où l'analyse des plateaux grâce aux marches neutres a montré que le best-known ne pouvait pas être trouvé facilement uniquement avec une méthode de descente classique. En effet, les 30 marches neutres se déplaçaient sur des plateaux des optima locaux où des portes pouvaient effectivement être trouvées en peu de pas. Ainsi, les performances de

NILS_0 sur ces trois instances sont moins bonnes que celles de NILS_{max}. Pour l'instance 5020mxc, le best-known n'est trouvé que 24 fois sur 30 comparé à NILS_{max}. Pour l'instance 10020jc, il n'est trouvé que 4 fois et bien sûr, il n'est jamais trouvé pour l'instance 20020jc car un nouveau best-known a été découvert par NILS_{max}.
Pour les instances structurées, l'analyse de la structure neutre a montré que dans certains cas, les plateaux des optima globaux étaient facilement atteignables par une méthode de descente classique mais que pour d'autres, ce n'était pas le cas. Cette étude des performances de NILS prouve que l'exploitation des plateaux pour ce type de neutralité est intéressante pour optimiser. En effet, les résultats obtenus avec $\text{NILS}_{MNS>0}$ sont équivalents voire supérieurs à ceux d'une recherche locale itérée classique (NILS_0).
Afin de poursuivre l'évaluation de NILS, nous l'utilisons pour résoudre le problème des paysages-NKq.

3.4.3 NILS sur les paysages-NKq

Nous avons vu dans la section 3.2 que les instances neutres des paysages-NKq présentent divers degrés de neutralité et de rugosité. L'analyse de la structure neutre du paysage a prouvé l'existence de plusieurs plateaux sur lesquels une marche aléatoire rencontre au moins une porte après peu de pas (maximum 3).

3.4.3.1 Design expérimental

Pour étudier les performances de NILS sur les paysages-NKq, les mêmes instances que pour l'analyse de la structure neutre du paysage ont été utilisées à savoir $N = 64$, $K \in \{2, 4, 6, 8\}$ et $q \in \{2, 3, 4\}$. Pour chaque instance, six valeurs de MNS plus ou moins grande $(0, 2, 4, 8, 16, 64)$ ont été testées pour étudier l'impact de cette longueur sur les performances de NILS. Pour rendre cette étude la plus robuste possible et parce que les expérimentations sur ces instances sont assez rapides, comparées à celles du problème de flowshop par exemple, NILS a été exécuté 100 fois sur chaque instance et pour chacune des valeurs du paramètre MNS. $MNS = 0$ correspond à une recherche locale itérée (ILS) où le dernier optimum local trouvé est utilisé pour redémarrer la recherche dans une partie proche de celui-ci. La distribution des 100 valeurs de fitness trouvées pour chaque instance et valeur de MNS n'étant pas symétrique, les médianes des performances sont données plutôt que la moyenne.
Dans le but de laisser chaque configuration de NILS converger, la condition d'arrêt correspond à un nombre d'évaluations maximal, égal à 10^5.
Dans l'étape de perturbation, un kick-move est utilisé quand aucune meilleure solution n'a été trouvée avant le nombre maximal de pas neutres atteint. Ici, quatre bits de l'optimum local courant sont changés ce qui permet de faire varier la valeur de fitness de la solution et ainsi redémarrer à nouveau la recherche avec la méthode de descente dans une nouvelle partie de l'espace de recherche.

3.4.3.2 Résultats expérimentaux

Le tableau 3.14 présente pour chaque instance, la meilleure valeur de fitness trouvée quelque soit la valeur du paramètre MNS en comparaison avec le best-overall, la meilleure

Tableau 3.14 – Meilleures performances de NILS$_*$ pour les paysages-NKq et différentes valeurs de K et de q. La valeur en gras indique que le best-overall n'est atteint par aucune configuration de NILS.

Instances		Best-overall	NILS$_*$
$q=2$	$K=2$	0.9219	0.9219
	$K=4$	0.9688	0.9688
	$K=6$	0.9844	0.9844
	$K=8$	0.9844	0.9844
$q=3$	$K=2$	0.8516	0.8516
	$K=4$	0.9219	0.9219
	$K=6$	0.9219	0.9219
	$K=8$	0.9219	0.9219
$q=4$	$K=2$	0.8281	0.8281
	$K=4$	0.8802	0.8802
	$K=6$	0.8854	0.8854
	$K=8$	0.8802	**0.875**

solution connue parmi toutes nos expérimentations sur ces instances. Au moins une configuration de NILS trouve le best-overall pour toutes les instances excepté pour $K=8$ et $q=4$. Néanmoins, d'après les résultats des marches neutres pour cette dernière instance, il était prévisible que même en exploitant la neutralité il serait difficile de la résoudre car la rugosité est élevée et les plateaux sont petits et près de la moitié n'ont pas de portes.

Le tableau 3.15 présente pour chaque instance les performances médianes de NILS$_0$, NILS$_4$ et NILS$_{64}$. Ces trois paramétrages de NILS ($MNS = 0, 4, 64$) permettent de voir l'importance de la valeur du paramètre MNS. Clairement NILS$_{64}$ obtient pour la plupart des instances des performances médianes moins bonnes que NILS$_0$ et NILS$_4$ (valeurs soulignées). Pour les instances où $K=8$, NILS$_4$ est meilleur que NILS$_0$ (valeurs en gras). Les plateaux de ces instances sont plus petits mais leur exploration conduit à de bons résultats. Se déplacer sur les plateaux est donc un bon moyen d'améliorer sans jamais dégrader la solution courante.

Pour les paysages-NKq, l'exploitation des plateaux est plus discutable que pour le problème de flowshop. En effet, une trop importante exploration des plateaux (comme pour NILS$_{64}$) n'est pas satisfaisante par rapport à l'utilisation d'un kick-move. Néanmoins, autoriser quelques déplacements neutres à partir d'un optimum local trouvé permet d'obtenir des performances meilleures (comme pour NILS$_4$). L'analyse de la structure des plateaux (*cf.* section 3.3.3) montrait que des portes étaient facilement atteignables dès lors que le plateau en possède. NILS$_4$ est donc un bon compromis entre l'exploration trop intensive des plateaux qui perd la recherche et la non considération des plateaux.

Tableau 3.15 – Performances de NILS pour les paysages-NKq et différentes valeurs de K et de q. NILS_0 donne la médiane des 100 valeurs de fitness trouvées pour $MNS = 0$. NILS_4 donne la médiane des 100 valeurs de fitness trouvées pour $MNS = 4$. NILS_{64} donne la médiane des 100 valeurs de fitness trouvées pour la valeur $MNS = 64$. La meilleure médiane parmi NILS_0, NILS_4 et NILS_{64} est en caractère gras quand les trois sont différentes sinon, les meilleures médianes sont soulignées.

Instances		NILS_0	NILS_4	NILS_{64}
$q = 2$	$K = 2$	0.9219	0.9219	0.9219
	$K = 4$	0.9688	0.9688	0.9688
	$K = 6$	0.9688	0.9688	0.9688
	$K = 8$	0.9375	**0.9531**	0.9375
$q = 3$	$K = 2$	0.8438	0.8438	0.8438
	$K = 4$	0.9141	0.9141	0.9141
	$K = 6$	<u>0.8984</u>	<u>0.8984</u>	0.8906
	$K = 8$	0.875	**0.8828**	0.875
$q = 4$	$K = 2$	0.8281	0.8281	0.8281
	$K = 4$	<u>0.8802</u>	<u>0.8802</u>	0.8646
	$K = 6$	<u>0.8593</u>	<u>0.8593</u>	0.8542
	$K = 8$	0.849	**0.8516**	0.844

3.4.3.3 Comparaison avec la littérature

Pour les paysages-NKq, les performances des différents paramétrages de NILS ont montré que NILS_4 obtient les meilleurs résultats parmi les configurations de NILS avec neutralité mais que NILS_0 obtient des performances très peu différentes de celles de NILS_4. Ces résultats amènent à se demander s'il est vraiment nécessaire d'utiliser la neutralité des optima locaux ou encore, si au contraire, il est nécessaire de considérer la neutralité à tous les niveaux de la recherche.

Ainsi, les performances de NILS sont comparées à celles d'un Netcrawler (NC). Afin de rendre la comparaison plus juste, nous permettons à NC de redémarrer d'une solution de l'espace de recherche quand il ne converge plus après un certain nombre d'itérations, préalablement défini par des expérimentations.

En outre, les performances de NILS sont comparées à celles d'un First Improvement Hill Climbing (FIHC), redémarré de n'importe quelle solution de l'espace de recherche dès qu'un optimum local est trouvé. Ainsi, l'intérêt de la perturbation est montré.

La figure 3.14 présente les performances médianes de NILS_4, de NILS_0, de NC et de FIHC pour les différentes instances testées avec les paramètres de neutralité et rugosité (q, K). Dans tous les cas, FIHC donne les pires performances. Les paysages des instances sont trop rugueux et/ou neutres pour qu'une simple utilisation des méthodes de descente ne suffit à donner de bons résultats. Si ce n'est pour les instances très neutres ($q = 2$ et $K = 3, 4$), NC est moins performant que les 2 configurations de NILS. Exploiter la neutralité pour toutes les solutions de la recherche n'est donc pas le meilleur moyen de résoudre ces problèmes.

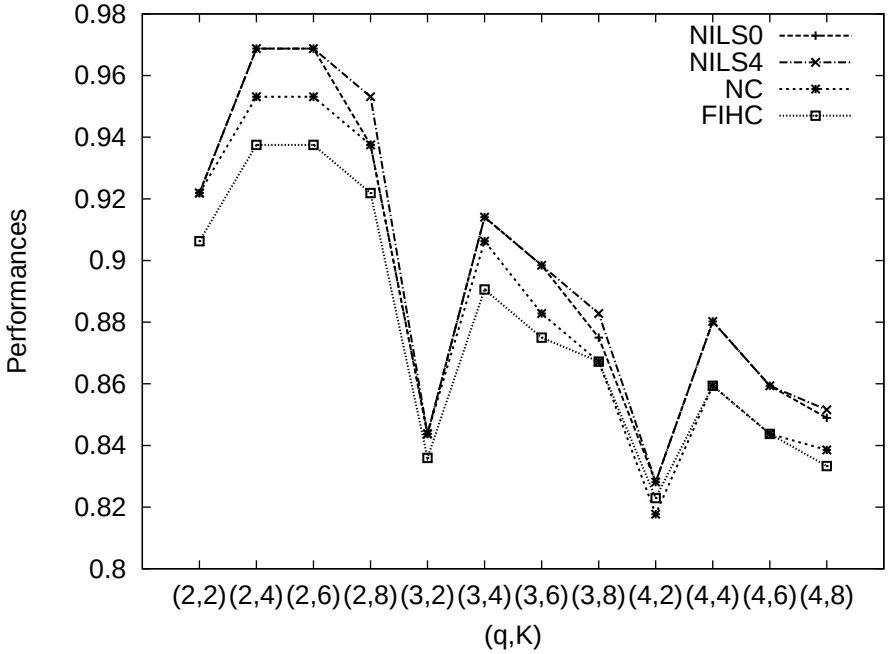


Figure 3.14 – Comparaison des performances médianes de NILS sans neutralité ($NILS_0$), NILS avec neutralité ($NILS_4$), du Netcrawler (NC) et du First Improvement Hill Climbing (FIHC) en fontion du couple (q, K) pour les paysages-NKq.

Néanmoins, utiliser des indicateurs pour guider et diriger la recherche sur les réseaux de neutralité de l'ensemble du paysage pourrait être une bonne piste pour exploiter ces réseaux de neutralité qui sont nombreux.

3.4.4 Discussion

Dans cette section, nous venons de présenter NILS, une recherche locale exploitant la neutralité des optima locaux pour se déplacer d'un nombre de pas fixé par le paramètre MNS jusqu'à trouver une porte. NILS a été analysé sur deux problèmes neutres dont les structures avaient été caractérisées dans les sections précédentes. Le flowshop de permutation est un problème neutre qui présente de larges plateaux sur lesquels des portes peuvent être trouvées. Les paysages-NKq sont des problèmes dits académiques où les degrés de rugosité et de neutralité du paysage sont réglés par respectivement les paramètres K et q. De plus, selon les instances, les performances de NILS ont été comparées à des méthodes classiques ou performantes de la littérature.

Les instances de Taillard pour le problème de flowshop sont des instances raisonnablement neutres. L'étude des plateaux des premières instances de chaque taille ayant montré que les portes étaient atteignables en ne visitant que très peu de solutions sur un plateau, l'exploitation de la neutralité était indiquée. Effectivement, les performances de NILS sur ces instances sont comparables à celles du Netcrawler et d'une recherche gloutonne IG,

dédiée à la résolution du flowshop. Le paramètre MNS doit être, néanmoins, réglé avec parcimonie car de trop courtes ou trop longues marches sur le plateau peuvent s'avérer inefficace.
Les instances structurées du flowshop présentent des degrés de neutralité plus fort que les précédentes. Même si, l'étude du paysage neutre de ces instances a révélé que pour la plupart des instances, le best-known pouvait être trouvé avec des simples descentes, certaines étaient plus difficiles à résoudre. Pour ces dernières, NILS a prouvé qu'exploiter les plateaux conduit au best-known pour ce type de problème et structure neutres.
Pour le problème du flowshop, exploiter la neutralité des optima locaux conduit à des performances identiques voire supérieures aux autres méthodes. Il est donc intéressant d'exploiter les plateaux des optima locaux dans la dynamique de la recherche.

Les instances des paysages-NKq sont plus ou moins rugueuses et neutres. NILS est plus performant que le Netcrawler et qu'une méthode de descente stricte classique (FIHC). Néanmoins, NILS sans neutralité ($MNS = 0$) a des performances statistiquement identiques à celle de NILS avec $MNS = 4$, les configurations de NILS ayant une valeur de MNS plus grande étant moins bonnes. Dans cette étude, le kick-move correspond à 4 mouvements dans l'espace de recherche et permet ainsi de sauter directement sur une solution dans le voisinage d'ordre 4. Or, en regardant de plus près les résultats sur les marches neutres, (*cf.* section 3.3.3, tableau 3.9) nous nous apercevons que pour beaucoup d'instances, il est nécessaire de visiter 4 solutions sur le plateau pour atteindre une porte. Ainsi, avec ce paramétrage le kick-move permet potentiellement de se déplacer directement sur ces portes, ce qui expliquerait les résultats de NILS sans neutralité.

3.5 Conclusion

Dans ce chapitre, les définitions liées aux paysages neutres ont été données ainsi que quelques outils pour aider à les analyser. Un paysage est neutre quand la majorité des solutions de l'espace de recherche a des voisins de même qualité qu'elles-mêmes. Ces solutions forment alors des réseaux de neutralité aussi appelés plateaux. L'analyse des paysages selon les critères de neutralité, des problèmes de flowshop et paysages-NKq, a montré que le degré de neutralité pouvait être plus ou moins élevé. Les marches neutres sont des déplacements de voisins en voisins de même qualité. Elles permettent d'appréhender et étudier la structure des plateaux d'un paysage en collectant des informations. Par exemple, elles sont capables d'indiquer si partant d'un optimum local, par des déplacements sur le plateau, il est rapide de rencontrer des portes, solutions du plateau dont au moins un des voisins est de meilleure qualité. Nous avons montré que les instances de Taillard pour le flowshop sont des instances neutres ayant de larges plateaux présentant de nombreuses portes. Les instances structurées pour le flowshop ont, quant à elles, un très fort degré de neutralité et leurs paysages sont très plats. Les instances des paysages-NKq présentent une neutralité de l'ordre de grandeur de celle des instances de Taillard pour le flowshop. Néanmoins, la structure est différente étant donné la taille de l'espace et du voisinage. Les plateaux, sont alors plus petits, mais les portes sont toujours assez rapidement atteignables en se déplaçant sur ces plateaux.

De ces observations, une recherche locale exploitant la neutralité a été conçue. NILS (Neutrality-based Iterated Local Search) est une recherche locale itérée où la perturbation se déplace aléatoirement sur le plateau du dernier optimum local trouvé. Ce déplacement a un nombre limité de pas, réglé par le paramètre MNS, pour éviter que la recherche se perde sur les plateaux. Dès lors qu'un voisin améliorant est trouvé, l'exploitation de la recherche s'arrête et ce voisin devient la solution courante. Sinon, si un tel voisin n'est jamais trouvé après s'être déplacé d'un nombre de pas égal à MNS, une perturbation aléatoire est appliquée pour envoyer la solution courante dans une nouvelle partie de l'espace de recherche. Les performances de cette méthode ont été évaluées sur les instances décrites précédemment. Pour les instances de Taillard pour le flowshop, NILS a montré qu'il était aussi performant qu'une méthode dédiée de la littérature [RS07] ou que le Netcrawler, une méthode de descente acceptant les voisins neutres [Bar01]. Ce type de problème est favorable à l'exploitation de la neutralité. Pour les instances structurées, NILS a trouvé une nouvelle meilleure solution (best-known) pour une instance et a démontré sa robustesse pour trouver ceux des autres instances pour chacune de ses exécutions. Pour les paysages-NKq, les résultats sont assez discutables sur l'intérêt d'utiliser la neutralité mais ils montrent, néanmoins, que l'exploitation des plateaux donnent des performances au moins aussi bonnes que sans.

Ce chapitre montre comment l'analyse de la structure neutre d'un paysage a conduit à concevoir une recherche locale efficace et simple pour ce type de problème. Dès lors, que les plateaux du paysage possèdent des portes rapidement atteignables par une marche neutre l'utilisation de NILS amène à de bonnes performances.
Cette étude montre l'intérêt d'exploiter la neutralité du paysage à partir des optima locaux. Néanmoins, le Netcrawler qui autorise les déplacements neutres montre de bonnes performances qui pourrait être accrues en utilisant plus les informations sur la structure des plateaux. Guider la solution à travers les plateaux, ne serait-elle pas intéressante et possible pour ce type de problème ? C'est la question à laquelle nous allons essayer d'apporter des éléments de réponses dans le chapitre suivant.

Chapitre 4

Évolvabilité : guider une recherche locale

Publication en lien avec le chapitre :

– M.-É. Marmion, L. Jourdan, C. Dhaenens, A. Liefooghe, S. Verel, The Road to VEGAS : Guiding the Search over Neutral Networks, In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2011, ACM Press, pages 1979–1986, 2011

Sommaire

Dans le chapitre précédent, nous avons montré que les problèmes d'optimisation combinatoire présentent des niveaux de neutralité différents. Cette neutralité conduit à un paysage ayant de nombreux plateaux, réseaux de solutions voisines de même qualité. Ces plateaux représentent un frein dans la dynamique de la plupart des recherches locales puisque ne considérant pas les voisins ayant la même qualité, ces recherches locales s'arrêtent ou acceptent de détériorer la qualité de la solution courante pour tenter de trouver par la suite une meilleure solution. Or, nous avons montré sur deux types de problèmes neutres (flowshop et paysages-NKq) qu'il existe de nombreux plateaux desquels il est possible de s'échapper par des portes, solutions qui possèdent des voisins de meilleures qualités qu'elles-mêmes.
Le Netcrawler (NC) est une méthode de descente acceptant toute solution améliorante ou équivalente. Cette recherche locale profite ainsi de la neutralité et se déplace sur les plateaux. Dans le chapitre précédent, les performances du Netcrawler sur les problèmes de flowshop de permutation et paysages-NKq se sont révélées satisfaisantes. Exploiter la neutralité des solutions de l'espace de recherche représente un moyen facile d'améliorer sans jamais dégrader la recherche. Cependant, les plateaux peuvent être larges et le Netcrawler peut alors perdre, en vain, beaucoup de temps à les explorer. Dans ce cas, il semble intéressant de se demander si utiliser une stratégie pour guider la recherche à travers les plateaux ne permettrait pas d'augmenter les performances d'une telle méthode. De plus, dans le cas où un voisin neutre est accepté, le fait de l'utiliser immédiatement afin d'évaluer son voisinage nous paraît un choix assez arbitraire vis-à-vis des autres solutions déjà rencontrées sur le même plateau et pour lesquelles il reste des voisins à explorer.
Naturellement plusieurs questions se posent : Pourquoi toujours prendre la dernière solution acceptée pour évaluer son voisinage ? Pourquoi ne pas tenir compte des solutions précédemment rencontrées sur le plateau ? Comment choisir parmi ces solutions ? Certaines sont-elles des portes ? Comment guider la recherche vers les portes ? Ces questions seront notre fil conducteur pour proposer une recherche locale autonome exploitant la structure neutre des problèmes et l'ensemble des connaissances sur les solutions d'un même plateau.

4.1 Évolvabilité

Pour exploiter intelligemment les plateaux existants dans les problèmes neutres, il est nécessaire de pouvoir différencier les solutions du plateaux les unes par rapport aux autres et donc de connaître ou calculer d'autres informations que la valeur de fitness pour celles-ci.

4.1.1 Définitions

L'*évolvabilité* est un terme générique pour parler de la capacité d'un système à s'adapter au cours de son évolution. Ce concept a été repris pour les systèmes biologiques pour définir "la capacité pour des variations aléatoires à produire parfois une amélioration" [Alt94]. Ainsi, l'évolvabilité est utilisée comme une seconde mesure pour apprécier les qualités d'un individu et choisir le plus prometteur entre deux. Selon Wagner [Wag05], ces qualités sont basées sur la capacité de l'individu à produire des descendants robustes et adaptés à leur milieu. Cette définition est alors reprise dans le contexte de l'optimisation combinatoire [SHLO02, Ver05, VCC06] pour mesurer la capacité d'une solution à produire des

voisins performants.
Dans le cas où le paysage est neutre, l'évolvabilité représente alors une nouvelle mesure de qualité entre les solutions d'un même plateau. La plupart des mesures existantes de l'évolvabilité sont basées sur les valeurs de fitness des voisins d'une solution. Citons, par exemple, la moyenne des valeurs de fitness des solutions voisines ou le minimum, le maximum, la médiane ou encore la probabilité d'améliorer en évaluant le voisin d'une solution. L'autocorrélation de l'évolvabilité [VCC06] sur un plateau correspond à la fonction d'autocorrélation de la mesure de l'évolvabilité le long d'une marche aléatoire neutre. Elle permet ainsi de mesurer si l'information est aléatoire ou non sur les réseaux de neutralité. Quand la corrélation est large, les solutions proches les unes des autres sur un plateau ont des évolvabilités proches elles aussi. Dans ce cas, l'évolvabilité peut être utilisée pour guider une recherche sur les plateaux tout comme la valeur de fitness guide la recherche le long du paysage [Wei90].
Sur un plateau, une recherche locale efficace doit s'orienter rapidement vers une porte. Elle doit donc sélectionner sa solution voisine du plateau la plus prometteuse *i.e.* celle qui est une porte ou qui l'y conduira le plus vite possible. L'évolvabilité peut être utilisée pour faire ce choix entre les solutions d'un plateau. Dans d'autres situations, comme la sélection du meilleur opérateur dans les algorithmes évolutionnaires, des mesures sont utilisées pour apprécier la qualité d'un opérateur par rapport à un autre [FSS10]. La sélection d'opérateurs s'apparente à la sélection du meilleur voisin sur un plateau. Nous proposons d'adapter une stratégie de choix d'opérateurs utilisée dans les recherches adaptatives pour aider à choisir une solution sur un plateau [DFSS08].

4.1.2 Stratégies proposées

Les recherches adaptatives sont de plus en plus étudiées du fait de la croissance de la complexité des méthodes de recherche et des problèmes à optimiser. Le but de ces méthodes est d'adapter leurs mécanismes, souvent par le biais des paramètres, aux conditions de changements des problèmes. Le contrôle de paramètres correspond aux paramétrages *on-line*, *i.e.* au cours du processus d'optimisation, par exemple, de la représentation du problème, de différents opérateurs (voisinages, mutation, croisement, sélection), de la sélection des opérateurs, du taux d'application des opérateurs, etc. [EMSS07]. À partir d'un historique de la recherche, ces méthodes adaptatives sélectionnent alors le prochain paramétrage. Plusieurs règles existent pour effectuer cette sélection : les probabilités de *matching* (stratégies de décision), les poursuites adaptatives [Thi05] qui affectent une probabilité de succès à chaque opérateur, le bandit manchot à plusieurs bras [DFSS08]...
Le bandit manchot à plusieurs bras est un modèle séquentiel d'apprentissage, principalement étudié dans la théorie des jeux, utilisant un compromis entre exploration et exploitation. A bras indépendants sont considérés, chacun d'eux possédant un score qui suit une distribution inconnue. Une stratégie de sélection optimale du bras le plus prometteur est de maximiser un score qui se cumule au long de la recherche. La stratégie de la borne supérieure de confiance (Upper Confidence Bound, UCB) [ACBF02] est asymptotiquement optimale et est utilisée pour la sélection adaptative d'opérateurs [DFSS08].
Dans ce cas, pour chaque bras, qui représente un opérateur, un gain empirique est affecté reflétant sa qualité. Le score est alors calculé par la stratégie de la borne supérieure de

confiance :

$$\text{score}_{i,t} = \hat{r}_{i,t} + C\sqrt{\frac{2\ln\sum_k n_{k,t}}{n_{i,t}}} \tag{4.1}$$

où à l'instant t, $\hat{r}_{i,t}$ est le gain empirique mesuré de l'opérateur i, $n_{i,t}$ est le nombre de fois où l'opérateur i a été utilisé. L'opérateur (bras) qui a le meilleur score est sélectionné par la méthode pour poursuivre la recherche.
Dans la formulation de l'UCB (équation 4.1), le gain représente la qualité de l'opérateur. Ce gain étant une estimation de la qualité, la seconde partie de l'équation représente une mesure du bruit associé au gain. C est un paramètre indépendant de l'instance du problème considéré, représentant un facteur qui règle le compromis entre exploitation et exploration de tous les opérateurs pour que le calcul soit optimal.

Dans ce chapitre, nous adaptons cette stratégie de sélection d'opérateurs à la sélection de solutions. En effet, cette stratégie de sélection d'opérateurs a prouvé qu'elle était performante dans ce domaine. De plus, les plateaux du paysage étant des freins à la progression de la recherche, il est souhaitable de trouver le chemin le plus rapide pour atteindre une porte en se déplaçant neutralement. Les solutions du plateau représentent alors des bras du bandit manchot à tester. Le gain associé à une solution reflète la qualité de celle-ci et donc correspond à une mesure d'évolvabilité. La solution ayant le meilleur score est la plus évolvable du plateau (au bruit près). Le choix de cette solution entraîne l'évaluation d'un de ses voisins non encore connu. En résumé, cette méthode permet de considérer uniquement l'ensemble des solutions découvertes du plateau afin d'évaluer un voisin de la solution la plus évolvable d'entre elles, afin d'identifier les portes et rapidement trouver un de ses voisins améliorants.

4.2 VEGAS : comment profiter efficacement des plateaux ?

Dans le chapitre 3, l'intérêt d'exploiter les plateaux pour trouver une solution meilleure a été montré. Intuitivement, plus le degré de neutralité est élevé et plus le plateau sera large et donc long à parcourir. Pour les problèmes étudiés au chapitre 3, les optima locaux ont généralement un degré de neutralité plus faible que n'importe qu'elle autre solution de l'espace de recherche. Les solutions de mauvaises qualités ont des plateaux très larges qui peuvent faire barrage à la progression de la recherche locale. Il est alors nécessaire de pouvoir la diriger rapidement vers les bonnes solutions. VEGAS (Varying Evolvability-Guided Adaptive Search) est conçue pour tenir compte de la neutralité à tout niveau de la recherche en étant guidée à chaque moment par la solution connue la plus évolvable.

4.2.1 Considérer plusieurs solutions d'un même plateau

Pour commencer, définissons les termes aidant à la compréhension de la méthode. On dit qu'une solution est *évaluée* si sa valeur de fitness a été calculée. On dit qu'une solution est *visitée* si tous ses voisins ont été évalués, dans le cas contraire elle est *non-visitée*. Le voisinage d'une solution est évalué dans un ordre aléatoire.

Algorithme 3 VEGAS

$\mathcal{S} = \{s_0\}$
tant que $\exists s \in \mathcal{S}$ tel que s est non-visitée **faire**
  $s \leftarrow$ **sélectionner**$(\mathcal{S})$
  Choisir $s' \in \mathcal{N}(s)$ aléatoirement (sans répétition)
  si $f(s') < f(s)$ **alors**
    $\mathcal{S} \leftarrow \{s'\}$
  sinon si $f(s') = f(s)$ **alors**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{s'\}$
  fin si
  $\forall s \in \mathcal{S}$ **Mettre à jour** score(s, s')
fin tant que
Retourner $s \in \mathcal{S}$

Considérons une recherche locale qui améliore itérativement une solution courante s en explorant son voisinage. Dès qu'un voisin strictement meilleur est évalué, il est accepté et devient la nouvelle solution courante. Néanmoins, dès qu'un voisin neutre est évalué, une nouvelle solution du plateau courant est connu. L'idée principale de VEGAS est de considérer toutes ces solutions connues du plateau courant au lieu de se contenter d'une unique solution (la dernière acceptée, pour une recherche locale classique). On note $\mathcal{S}$, un sous-ensemble des solutions connues du plateau. VEGAS exploite les solutions de $\mathcal{S}$ dans le but d'explorer le plateau pour trouver une porte. VEGAS s'arrête quand toutes les solutions de $\mathcal{S}$ sont visitées. Il est évident que VEGAS se comporte comme un First Improvement Hill Climbing quand le problème n'est pas neutre.

4.2.2 Guider le déplacement sur le plateau

L'ensemble $\mathcal{S}$ étant défini, il est nécessaire de définir une stratégie pour choisir une solution parmi toutes ses solutions. En effet, pour poursuivre l'exploration du plateau, $|\mathcal{S}| > 1$ solutions sont alors envisageables. Une méthode de sélection permet alors de choisir la solution ayant le meilleur score parmi $\mathcal{S}$ pour évaluer un de ses voisins. Ces solutions ayant la même valeur de fitness, pour les différencier, nous proposons d'utiliser une mesure d'évolvabilité. L'évolvabilité reflétant la capacité d'une solution à conduire vers une meilleure solution de l'espace de recherche, cette mesure permet de calculer le score de chaque solution de $\mathcal{S}$. Ce score est ensuite utilisé par la méthode de sélection pour choisir la prochaine solution de $\mathcal{S}$ à tester. L'exploration des plateaux est alors guidée par cette méthode de sélection. L'algorithme 3 présente la structure générale de VEGAS. Toute solution évaluée ayant la valeur de fitness du plateau est enregistrée dans $\mathcal{S}$. Puis une méthode de sélection retourne la nouvelle solution à explorer. Ainsi, un nouveau voisin $s' \in \mathcal{V}(s)$ est évalué. Si s' est strictement de meilleure qualité que les solutions du plateau, alors l'ensemble $\mathcal{S}$ devient égal au singleton $\{s'\}$. Sinon, si s' est un voisin neutre, il est ajouté à $\mathcal{S}$. Considérant ce nouveau voisin (neutre ou moins bon), le score de toutes les solutions de $\mathcal{S}$ est recalculé et la méthode de sélection retourne alors la prochaine solution courante.
VEGAS est une recherche locale adaptative dont la méthode de sélection est l'un de ses principales composants. Par exemple, si cette méthode de sélection retourne toujours la

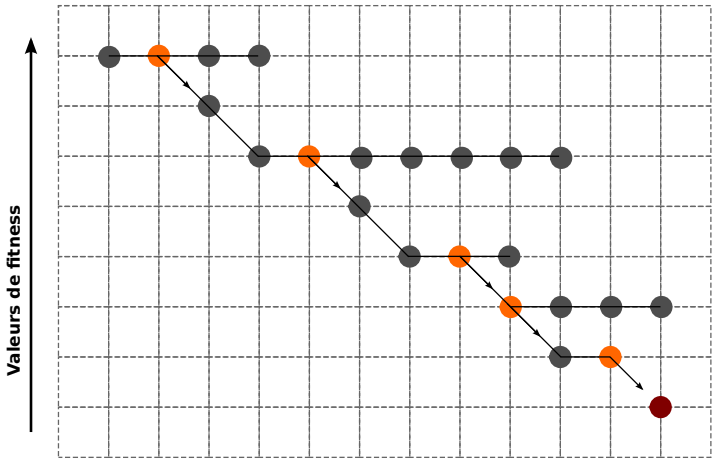
dernière solution neutre trouvée sur le plateau, cet algorithme se comporte comme un Netcrawler. Dans notre étude, nous proposons de baser la méthode de sélection selon un bandit manchot à plusieurs bras. Chaque solution connue du plateau $\mathcal{S}$ constitue un des bras du bandit. Celle qui a le meilleur score est choisie pour être la nouvelle solution courante. Le score est donné par l'équation de l'UCB (équation 4.1). Ainsi, dès qu'un nouveau voisin s' est évalué dans le voisinage d'une solution $s \in \mathcal{S}$, s' est enregistré pour mettre à jour la valeur du score de chaque solution de $\mathcal{S}$ selon le gain et la fréquence de sélection d'une solution. Le gain empirique du calcul du score correspond à une mesure d'évolvabilité. Le paramètre C de l'équation de l'UCB permet de faire varier le compromis entre exploration et exploitation des solutions du plateau. Quand C est petit, il donne plus d'importance à l'exploitation : la recherche considère plutôt les solutions les plus évolvables, donc se base sur le gain. Quand C est grand, il donne plus d'importance à l'exploration : la recherche considère plutôt les solutions très peu visitées du plateau.

4.2.3 Dynamique de VEGAS

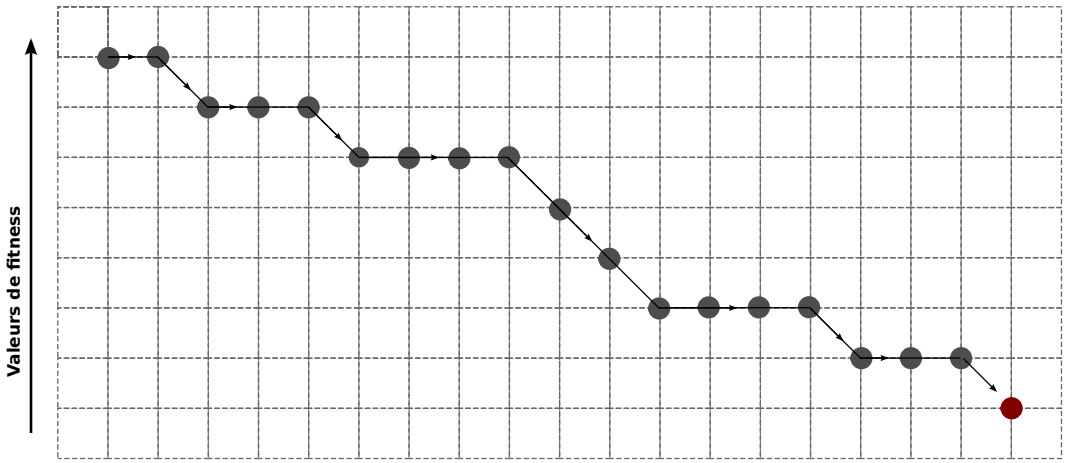
La figure 4.1 présente en parallèle la dynamique de VEGAS et du Netcrawler (NC) pour atteindre l'optimum global (en rouge, en bas à droite) dans un problème de minimisation. Ces représentations se veulent comme des modèles généraux de la dynamique des méthodes. VEGAS considère plusieurs solutions sur le même plateau et la recherche se déplace dès qu'un voisin améliorant d'une porte (en orange) est trouvé. VEGAS s'arrête quand l'optimum global est trouvé. Parallèlement à la dynamique de VEGAS, la figure 4.1 permet de comprendre celle de NC et d'en noter les principales différences. NC accepte toute solution meilleure ou équivalente et donc, se déplace sur les plateaux. Contrairement à VEGAS, seule la dernière solution trouvée sur le plateau est considérée. Ces deux représentations montrent que VEGAS et NC tiennent compte de tous les niveaux de neutralité mais considèrent respectivement soit un plateau de solution soit une unique solution.

4.2.4 Mise en œuvre de VEGAS

VEGAS est une méthode qui exploite la neutralité d'un problème d'optimisation tout en profitant de l'évolvabilité des solutions pour se guider sur les plateaux. Dans le chapitre 3, les structures neutres des problèmes de flowshop de permutation et des paysages-NKq ont été étudiées. Nous avons observé que profiter de la neutralité du problème donne des résultats satisfaisants par rapport à des méthodes dédiées ou n'exploitant pas cette structure neutre. De plus, pour les paysages-NKq, il est possible de faire varier la structure neutre et rugueuse du problème. Ceci aide à appréhender les performances d'une méthode sur une telle structure neutre. Afin de comprendre le fonctionnement de VEGAS, il a été analysé, tout d'abord, sur les paysages-NKq puis sur les instances de Taillard du flowshop. VEGAS est une méthode adaptative guidée par le score, donc par l'évolvabilité, des solutions connues du plateau. Pour apprécier l'efficacité du guidage, il est alors intéressant de comparer VEGAS à une méthode qui choisirait aléatoirement une solution parmi celles du plateau. AS_{alea} est une recherche adaptative aléatoire qui correspond à une configuration de VEGAS où la méthode de sélection est remplacée par un tirage aléatoire d'une solution du plateau. De même, si la dernière solution neutre trouvée était systématiquement choisie, VEGAS se comporterait comme un Netcrawler (NC). Dans cette étude des perfor-



VEGAS



NC

Figure 4.1 – Dynamiques de VEGAS et de NC pour un problème de minimisation. À partir d'une solution de l'espace de recherche (en haut à gauche), la trajectoire des mouvements sur le paysage est explicitée jusqu'à atteindre l'optimum global (en rouge, en bas à droite). Les portes sur les plateaux de VEGAS sont de couleur orange.

mances de VEGAS sur les problèmes des paysages-NKq et de flowshop, les performances de plusieurs configurations de VEGAS sont analysées puis les performances de VEGAS sont comparées à celles de AS_{alea} et NC.

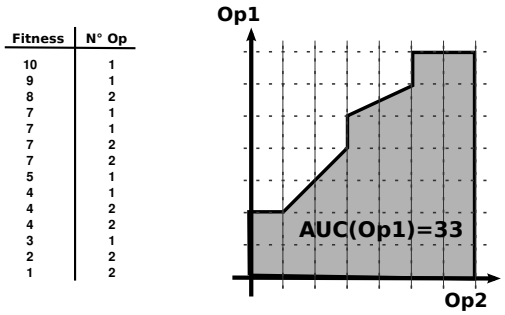


Figure 4.2 – Calcul de l'AUC pour l'opérateur op_1.

4.3 VEGAS sur les paysages-NKq

Les paysages-NKq sont une catégorie de problèmes pour lesquels il est possible de régler le degré de rugosité et le degré de neutralité du problème. Dans le chapitre 3, nous avons étudié plusieurs instances des paysages-NKq pour $N = 64$. Plusieurs valeurs de K et de q ont été testées pour obtenir des degrés de neutralité différents. La section 3.4.3.3 a montré qu'il est intéressant d'utiliser la structure de neutralité du problème pour mieux le résoudre. Ces performances ne pourraient-elles pas être accrues si la recherche était guidée lors de l'exploitation de la neutralité ? Pour tenter de répondre à cette question, VEGAS a été exécuté sur différentes instances des paysages-NKq.

4.3.1 Mesure d'évolvabilité

Pour utiliser VEGAS, il est nécessaire de définir une mesure d'évolvabilité pour calculer le gain de chaque solution. Dans les recherches adpatatives de sélection d'opérateurs, différentes mesures ont été proposées : l'amélioration de la valeur de fitness moyenne entre les populations parents et enfants d'un algorithme génétique, l'amélioration de la meilleure valeur de fitness dans une fenêtre de temps [DFSS08] ou plus récemment, l'aire sous la courbe (AUC : Area Under the Curve) [FSS10]. Cette dernière mesure compare les valeurs de fitness des solutions trouvées à la suite de l'utilisation de chaque opérateur. Pour classer les opérateurs les uns par rapport aux autres, la mesure utilise une méthode de rang plutôt que de normaliser l'amélioration de la qualité. La figure 4.2 donne un exemple de calcul de l'AUC pour deux opérateurs. Les valeurs de fitness des solutions trouvées sont associées à l'un des opérateurs puis rangées dans une liste décroissante. Une courbe est dessinée à partir du point $(0, 0)$ puis selon cette liste : quand l'opérateur op_1 est concerné, la courbe suit l'axe op_1 (en ordonnée) sinon elle suit l'autre axe (en abscisse). En cas d'égalité des opérateurs, autant d'importance est donnée à chacun des axes (diagonale). Pour chaque opérateur, l'aire sous la courbe (AUC) est calculée. La qualité de l'opérateur est alors égale à son AUC divisée par la somme totale des aires.
Fialho et al. [FSS10] ont utilisé cette mesure pour définir le gain dans le calcul du score (équation 4.1) et ont montré qu'elle était très efficace. Tout comme pour le bandit manchot à plusieurs bras, nous adaptons cette mesure aux solutions connues du plateau. Dans VEGAS, la méthode de sélection choisie une solution parmi les bras pour évaluer un de

ses voisins non encore connu. Si celui-ci n'est pas améliorant, sa valeur de fitness est alors associée à cette solution et joue un rôle dans le calcul de son gain. Plus une solution a des voisins de meilleures qualités que les autres, plus elle est évolvable *i.e.* , son gain sera plus élevé.

4.3.2 Performance et analyse du paramètre C

Nous cherchons à analyser les performances de VEGAS. Il est à noter que VEGAS n'a qu'un unique paramètre C qui intervient dans le calcul du score de chaque solution du plateau et règle ainsi le compromis entre l'exploration et l'exploitation des solutions du plateau. Dénotons VEGAS_n, la configuration de VEGAS où $C = n$.

4.3.2.1 Design expérimental

Les performances de VEGAS sur les paysages-NKq ont été analysées sur les mêmes instances que celles utilisées dans le chapitre 3, à savoir $N = 64$, $K \in \{2, 4, 6, 8\}$ et $q \in \{2, 3, 4\}$. Le voisinage correspond à une différence de un bit dans la représentation des solutions.
Dans le cadre de la sélection d'opérateurs, Fialho *et al.* [FSS10] ont testé plusieurs valeurs du paramètre C. Ce travail étant une adaptation pour la sélection de solutions, les mêmes valeurs ont été utilisées.
La valeur du paramètre C est dans l'ensemble $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 500\}$. Chaque configuration de VEGAS a été exécutée 100 fois sur chacune des instances.
Le critère d'arrêt correspond au nombre d'évaluations et est égal à 10^5. Comme la méthode peut s'arrêter avant d'avoir atteint ce nombre maximal d'évaluations lorsque tous les bras sont visités, VEGAS est intégré dans une recherche locale itérée qui redémarre d'une solution aléatoire.

4.3.2.2 Résultats expérimentaux

4.3.2.2.1 Impact du paramètre C Le tableau 4.1 présente pour chaque instance la meilleure valeur connue (best-known) et la meilleure valeur trouvée par l'ensemble des configurations de VEGAS, notée VEGAS_*. Ces résultats montrent qu'au moins une configuration de VEGAS atteint le best-known pour toutes les instances excepté pour celle $K = 8, q = 3$. VEGAS est donc une métaheuristique intéressante à étudier car elle peut permettre de trouver le best-known.
C est le paramètre principal de VEGAS qui règle le compromis entre exploration et exploitation des solutions du plateau. Le tableau 4.2 présente les performances médianes de trois configurations de VEGAS : (i) VEGAS-exploitation ou $\text{VEGAS}_{C<1}$ (représentée par $\text{VEGAS}_{0.01}$) pour une configuration qui favorise les solutions évolvables du plateau, (ii) $\text{VEGAS}_{C=1}$ pour une configuration qui oscille entre exploration et exploitation des solutions connues du plateau et, (iii) VEGAS-exploration ou $\text{VEGAS}_{C>1}$ (représentée par VEGAS_{100}) pour une configuration qui favorise les solutions peu visitées du plateau. En plus des médianes, le nombre de fois sur 100 où le best-known a été trouvé est précisé en indice.

Tableau 4.1 – Meilleures performances de VEGAS$_*$ pour les paysages-NKq et différentes valeurs de K et de q. La valeur en gras indique que le best-known n'est atteint par aucune configuration de VEGAS. Rappelons que c'est un problème de maximisation.

Instances		Best-known	VEGAS$_*$
$q=2$	$K=2$	0.9219	0.9219
	$K=4$	0.9688	0.9688
	$K=6$	0.9844	0.9844
	$K=8$	0.9844	0.9844
$q=3$	$K=2$	0.8516	0.8516
	$K=4$	0.9219	0.9219
	$K=6$	0.9219	0.9219
	$K=8$	0.9219	**0.9141**
$q=4$	$K=2$	0.8281	0.8281
	$K=4$	0.8802	0.8802
	$K=6$	0.8854	0.8854
	$K=8$	0.8802	0.8802

Tout d'abord, en comparant les résultats obtenus pour toutes les instances, l'efficacité de VEGAS semble ne pas être très sensible à la valeur du paramètre C. Cette appréciation a été confirmée par le test des rangs non-paramétrique apparié de Wilcoxon qui montre qu'aucune tendance générale ne peut être trouvée quant à une configuration de VEGAS qui donnerait les meilleures performances en médiane quelque soit l'instance. Néanmoins, pour certaines d'entre elles, on remarque que VEGAS-exploration ($C > 1$) obtient de meilleurs résultats que VEGAS-exploitation ($C < 1$) (valeurs en gras). Cette dernière configuration favorise les solutions évolvables qui sont censées mener efficacement aux portes. Or, ces résultats infirment cette idée et remettent donc en cause le choix de la mesure d'évolvabilité dans cette étude.
L'alternance du choix entre solutions très évolvables et solutions peu visitées (VEGAS$_1$) conduit à des performances médianes identiques à VEGAS-exploration, excepté pour le paramétrage $K = 4, q = 4$ (valeurs en gras). De plus, le nombre de fois où la meilleure solution est atteinte pour chacune des configurations amène une autre conclusion. En effet, sur 100 exécutions, VEGAS-exploration trouve au moins une fois le best-known ce qui n'est pas le cas des autres configurations. De même, la fréquence où la configuration VEGAS-exploration trouve le best-known est, en général, plus grande que pour les deux autres (valeurs soulignées). L'analyse des performances selon la valeur du paramètre C amène donc à conclure que favoriser l'exploration du plateau conduit à des résultats meilleurs et plus robustes.

4.3.2.2.2 Neutralité et performances La dynamique d'une métaheuristique est intéressante à étudier car elle peut expliquer les performances selon la valeur du paramètre. VEGAS est une recherche locale tenant compte de la neutralité du problème ayant un

Tableau 4.2 – Performances de VEGAS pour les paysages-NKq et différentes valeurs de K et de q. VEGAS$_{C<1}$ (VEGAS-exploitation) donne la médiane des 100 valeurs de fitness trouvées pour $C = 0.01$. VEGAS$_1$ donne la médiane des 100 valeurs de fitness trouvées (exploitation et exploration des solutions). VEGAS$_{C>1}$ (VEGAS-exploration) donne la médiane des 100 valeurs de fitness trouvées pour $C = 100$. Pour chacune des valeurs, le nombre de fois où le best-known est atteint, est précisé en indice. La meilleure performance en médiane est en caractère gras. La configuration de VEGAS qui atteint le plus le best-overall est soulignée.

Instances		VEGAS$_{C<1}$ (Exploitation)	VEGAS$_1$	VEGAS$_{C>1}$ (Exploration)
$q = 2$	$K = 2$	0.9219_{52}	0.9219_{54}	$\underline{0.9219_{56}}$
	$K = 4$	0.9531_{45}	$\mathbf{0.9688}_{60}$	$\underline{\mathbf{0.9688}_{65}}$
	$K = 6$	0.9609_{5}	$\mathbf{0.9688}_{7}$	$\underline{\mathbf{0.9688}_{9}}$
	$K = 8$	0.9531_{2}	$\underline{0.9531_{4}}$	0.9531_{1}
$q = 3$	$K = 2$	0.8359_{3}	0.8359_{4}	$\underline{0.8359_{9}}$
	$K = 4$	0.9141_{4}	0.9141_{7}	$\underline{0.9141_{12}}$
	$K = 6$	0.8829_{0}	$\mathbf{0.8906}_{0}$	$\underline{\mathbf{0.8906}_{1}}$
	$K = 8$	0.8750_{0}	$\mathbf{0.8828}_{0}$	$\underline{\mathbf{0.8828}_{1}}$
$q = 4$	$K = 2$	0.8203_{18}	$\mathbf{0.8229}_{25}$	$\underline{\mathbf{0.8229}_{29}}$
	$K = 4$	0.8698_{9}	0.8698_{12}	$\underline{\mathbf{0.8750}_{15}}$
	$K = 6$	0.8489_{1}	$\underline{\mathbf{0.8542}_{2}}$	$\mathbf{0.8542}_{1}$
	$K = 8$	0.8437_{0}	$\mathbf{0.8489}_{0}$	$\underline{\mathbf{0.8489}_{1}}$

paramètre principal C qui règle le compromis entre l'exploitation de la solution la plus évolvable et l'exploration de la solution la moins visitée.
La neutralité d'une instance joue sur la dynamique de VEGAS puisqu'elle conduit à des plateaux qui peuvent être très large mais desquels il faut s'échapper. Pour les problèmes-NKq, la section 3.3 a montré que le degré de neutralité décroît quand la valeur de K et/ou q augmente. La figure 4.3 donne le nombre moyen de solutions évaluées sur chacun des plateaux de VEGAS$_{100}$ par rapport au degré de neutralité. On observe que ce nombre croît exponentiellement avec le degré de neutralité. Les deux pics correspondent à $K = 2$ et $q = \{3, 4\}$ où, même si la neutralité est importante, la rugosité est très faible ce qui favorise des plateaux très larges. VEGAS est impacté par la neutralité du problème car plus la neutralité est importante, plus le nombre de solutions évaluées sur chaque plateau est grand et donc, plus il y a de choix dans les solutions à évaluer.

4.3.2.2.3 Compromis exploration-exploitation L'analyse des performances selon la valeur du paramètre C a montré qu'explorer un grand nombre de solutions sur les différents plateaux conduit en général à de meilleures performances. Pour analyser plus finement cette observation, certaines statistiques ont été calculées pour étudier la dynamique de la recherche : (i) le nombre de plateaux parcourus et, (ii) le nombre de solutions

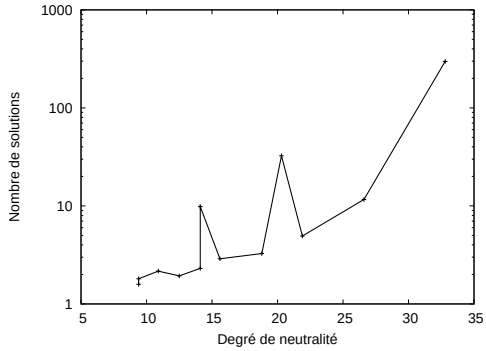
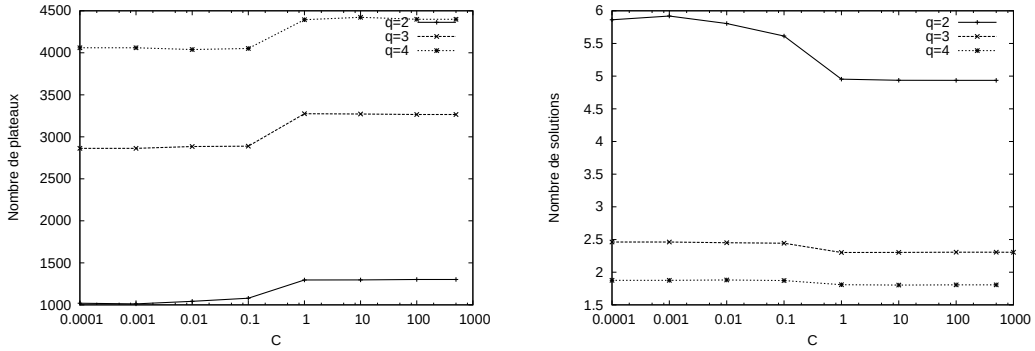


Figure 4.3 – Nombre moyen de solutions évaluées par plateaux par rapport au degré de neutralité pour VEGAS_{100}.



(a) (b)

Figure 4.4 – Nombre moyen de plateaux (a) et nombre moyen de solutions évaluées sur chacun d'eux (b). Ces nombres sont donnés en fonction de la valeur du paramètre C pour $q \in \{2, 3, 4\}$ et $K = 6$.

évaluées sur chacun de ces plateaux qui correspond à la taille connue des plateaux explorés. La figure 4.4 présente ces deux quantités pour les différentes valeurs de q et C testées pour l'instance $K = 6$. Les résultats sont similaires pour les autres valeurs de K et ne sont pas présentés ici. Premièrement, il existe une différence très nette entre VEGAS-exploration et VEGAS-exploitation. En effet, pour $C > 1$ les valeurs sont identiques. De même pour $C < 1$. Les deux courbes peuvent être séparées environ pour $C = 0.5$ en deux parties homogènes. Deuxièmement, plus le nombre moyen de plateaux est grand, moins le nombre de solutions évaluées par plateau est grand. Ceci prouve qu'il existe une réelle différence entre les dynamiques de VEGAS selon les valeurs du paramètre C. En effet, quand le paramètre C est réglé pour explorer les différents plateaux, beaucoup de plateaux sont explorés mais la portion connue est alors faible. Quand le paramètre C est réglé pour exploiter les plateaux, peu de plateaux sont explorés mais beaucoup de solutions sont évaluées. Cette dynamique peut expliquer que VEGAS-exploration donne, dans cer-

tains cas, de meilleures performances que VEGAS-exploitation. Par conséquent, le réglage du paramètre C impacte, outre le compromis exploration/exploitation, la dynamique de VEGAS.

4.3.3 Comparaison avec d'autres approches

VEGAS est une recherche locale qui tient compte de la neutralité du problème à chaque instant de la recherche. Sa dynamique est donc proche de celle du Netcrawler (NC) qui accepte toute nouvelle solution de valeur de fitness meilleure ou identique à la solution courante et de AS_{alea} qui comme VEGAS enregistre les solutions du plateau mais choisit l'une d'entre elles au hasard pour évaluer l'un de ses voisins.
Dans la section précédente, VEGAS_{100} a montré les meilleures performances. Nous l'utilisons donc comme configuration de VEGAS pour cette étude comparative.

4.3.3.1 Design expérimental

Les mêmes instances que précédemment sont utilisées pour cette étude comparative. Les trois méthodes étudiées (VEGAS_{100}, AS_{alea} et NC) démarrent d'une solution aléatoire et s'arrêtent dès que 10^5 solutions ont été évaluées. Ces méthodes peuvent s'arrêter avant d'avoir atteint le nombre d'évaluations maximal, elles sont donc intégrées dans une recherche locale itérée qui redémarre d'une solution aléatoire. NC pouvant se perdre sur un plateau très large, une étude préliminaire a permis d'identifier pour chaque instance un nombre maximal d'évaluations qu'il ne semble pas intéressant de dépasser. Quand ce nombre est atteint, NC est alors redémarré.

4.3.3.2 Résultats expérimentaux

Dans le but de comparer les performances de ces trois méthodes en fonction des paramètres K et q, les valeurs de fitness ont été normalisées en utilisant la moyenne et l'écart-type des valeurs de fitness trouvées par les trois méthodes pour chaque instance. Ainsi, la performance moyenne est autour de zéro et les comportements extrêmes sont accrus. La moyenne $\bar{f}$ et l'écart-type σ_f sont calculés à partir des 100 valeurs de fitness trouvées par chacune des 3 méthodes. Pour une valeur de fitness f, sa valeur normalisée $\tilde{f}$ est égale à $\frac{f - \bar{f}}{\sigma_f}$.
Cette étude comparative entre les trois méthodes utilise ces performances normalisées. Pour chaque valeur de K, les performances sont tracées en fonction de q sur la figure 4.5. Pour $K \in \{4, 6, 8\}$, VEGAS_{100} et AS_{alea} sont meilleurs que NC. La comparaison des performances de VEGAS_{100} et AS_{alea} est plus difficile, car même si VEGAS_{100} est toujours meilleur que AS_{alea}, leurs performances sont très proches et peuvent ne pas être statistiquement différentes. Pour $K = 2$, la figure 4.5 (a) montre qu'aucune conclusion générale ne peut être donnée car NC est meilleur que VEGAS_{100} et AS_{alea} pour $q = 3$, mais moins bon pour $q = 4$ tandis que les performances de NC et de VEGAS_{100} semblent être égales pour $q = 2$. Le test des rangs non-paramétrique apparié de Wilcoxon (à 95%) a alors été utilisé pour tester l'hypothèse nulle d'égalité des performances de VEGAS_{100},

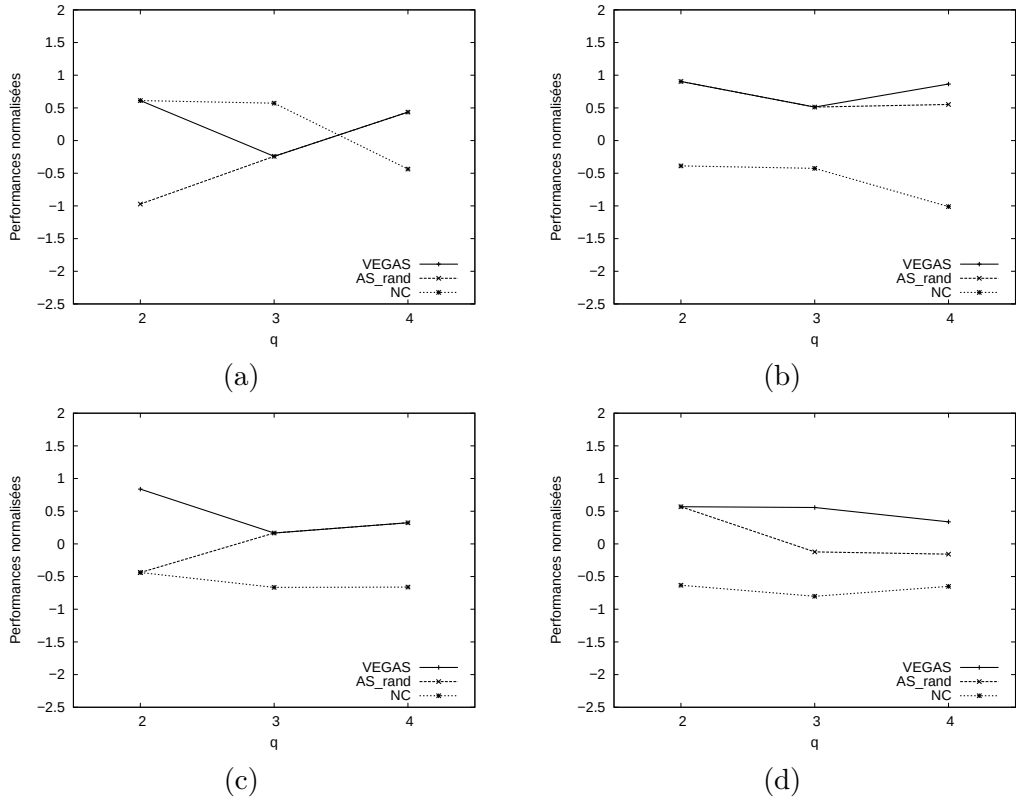


Figure 4.5 – Performances normalisées médiane de VEGAS_{100}, AS_{alea} et NC en fonction de q pour (a) $K = 2$, (b) $K = 4$, (c) $K = 6$, (d) $K = 8$

AS_{alea} et NC et ainsi analyser plus finement les résultats. Le tableau 4.3 donne les résultats de ces tests.

Ces résultats confirment que :

– pour $K = 4, 6, 8$, VEGAS_{100} et AS_{alea} ont des performances significativement meilleures que NC. Ceci montre que la manière dont on explore les plateaux est importante. En effet, une méthode qui a le choix parmi plusieurs solutions du plateau donne de meilleurs résulats que le Netcrawler qui sélectionne toujours la dernière solution évaluée du plateau.

– VEGAS_{100} n'est jamais moins bon que AS_{alea} et même parfois meilleur. Donc, guider le choix de la solution à considérer sur les plateaux permet d'obtenir de meilleures performances qu'une sélection aléatoire.

4.3.4 Synthèse

Cette analyse sur les paysages-NKq montre que guider l'exploration de la recherche sur les plateaux est un bon moyen d'obtenir de bonnes performances, au moins pour des degrés de neutralité raisonnables. Ce choix ne doit pas être arbitraire comme pour le Netcrawler qui choisit toujours la dernière solution, et doit suivre une loi de probabilité (aléatoire ou

Tableau 4.3 – Test des rangs non-paramétrique apparié de Wilcoxon pour les 100 exécutions de VEGAS_{100}, AS_{alea} et NC pour les paysages-NKq. $=$ indique que les méthodes ne sont pas significativement différentes à 95%, $>$ indique que la méthode de la ligne est meilleure que celle de la colonne et $<$ indique le contraire.

		$q=2$		$q=3$		$q=4$	
		VEGAS_{100}	AS_{alea}	VEGAS_{100}	AS_{alea}	VEGAS_{100}	AS_{alea}
$K=2$	AS_{alea}	$=$		$=$		$<$	
	NC	$>$	$>$	$>$	$>$	$<$	$=$
$K=4$	AS_{alea}	$=$		$<$		$=$	
	NC	$<$	$<$	$<$	$<$	$<$	$<$
$K=6$	AS_{alea}	$<$		$=$		$=$	
	NC	$<$	$<$	$<$	$<$	$<$	$<$
$K=8$	AS_{alea}	$=$		$<$		$=$	
	NC	$<$	$<$	$<$	$<$	$<$	$<$

plus complexe, comme avec la méthode des bandits). De même, cette étude montre qu'il est intéressant de considérer plusieurs solutions sur le plateau et que choisir les solutions dont peu de voisins ont été évalués est une bonne méthode pour s'échapper du plateau et améliorer la solution courante.
Dans cette étude, l'aire sous la courbe a été utilisée comme mesure de l'évolvabilité des solutions d'un même plateau. Or, VEGAS obtient de meilleurs résultats quand $C > 1$ et donc, lorsque l'évolvabilité n'est pas beaucoup prise en compte. Cette mesure ne semble donc pas être adaptée pour représenter l'évolvabilité d'une solution pour les instances des paysages-NKq.
Nous souhaitons donc dans la partie suivante analyser le comportement de VEGAS sur un autre problème (ici, le flowshop) avec une mesure d'évolvabilité plus classique.

4.4 VEGAS sur le problème de flowshop

Le problème de flowshop de permutation est un autre problème neutre, précédemment étudié dans le chapitre 3, pour lequel il s'agit de minimiser la date d'achèvement de la dernière tâche. Dans cette section, seules les instances de Taillard sont considérées. En effet, l'étude du paysage a révélé qu'il présentait de nombreux plateaux assez larges où de nombreuses portes sont accessibles uniquement avec un déplacement aléatoire sur le plateau. Des expérimentations ont montré que les méthodes qui tiennent compte explicitement de la neutralité obtiennent des performances équivalentes à une métaheuristique dédiée, une des meilleures de la littérature [RS07]. Exploiter les plateaux semble donc être un bon moyen d'obtenir de bons résultats sans connaissances *a priori*. VEGAS exploite la neutralité et guide la recherche sur les plateaux dans le but de trouver rapidement une

porte. Pour utiliser VEGAS, il est nécessaire de définir une mesure d'évolvabilité pour calculer le score de chaque solution du plateau. Or, dans la section précédente, l'aire sous la courbe [FSS10] a été utilisée comme mesure d'évolvabilité mais les résultats expérimentaux ont indiqué qu'elle n'est pas adaptée à la sélection de solution. Pour l'étude de VEGAS sur le flowshop, nous considérons une mesure d'évolvabilité plus classique.

4.4.1 Mesure d'évolvabilité

L'évolvabilité représente une seconde mesure pour appréhender la structure locale du paysage et continuer la recherche plus efficacement. Plusieurs auteurs [SHLO02, VCC06] ont utilisé la moyenne des valeurs de fitness des voisins, une mesure classique d'évolvabilité. Des expérimentations ont montré que la moyenne des valeurs de fitness des voisins est corrélée suivant que les solutions soient des portes ou non sur un même plateau. Cette mesure semble donc être adaptée à ce problème pour représenter l'évolvabilité des solutions du plateau. De plus, cette mesure étant plus classique, elle sera plus facile à étudier et à comprendre.
Néanmoins, les instances peuvent être très grandes et le voisinage complet d'une solution très coûteux à évaluer. Nous proposons donc de calculer incrémentalement l'évolvabilité de chaque solution du plateau. Quand une nouvelle solution du plateau est trouvée, son évolvabilité est calculée à partir de K voisins. Puis, quand l'un de ses voisins est évalué, sa valeur de fitness est utilisée pour calculer sa nouvelle évolvabilité. Pour que la valeur de l'évolvabilité soit significative, il est nécessaire que ces K premiers voisins ne soient pas neutres. Pour s'assurer que ces K voisins ne soient pas neutres avec une confiance α égale à 95%, pour chaque instance, la valeur de K est choisie telle que :

$$K > \frac{ln(1-\alpha)}{ln(d)}$$

où d est le ratio du degré de neutralité.
L'évolvabilité correspond au gain empirique du score donné par l'équation (4.1). De façon à ce que le gain reste indépendant par rapport aux instances, nous avons choisi d'ordonner toutes les évolvabilités du plateau dans l'ordre croissant. Le problème de flowshop étant un problème de minimisation, la solution ayant la plus petite évolvabilité est considérée comme la plus évolvable du plateau. Ainsi, à partir de cet ordre les solutions sont affectées d'un gain calculé grâce à l'ordre de la liste. La meilleure solution, en terme d'évolvabilité, a un gain égal à 1 et la dernière un gain égal à 0. Les solutions intermédiaires sont alors réparties équitablement dans l'intervalle $[0; 1]$.

4.4.2 Nouveaux paramètres

L'espace de recherche associé au problème de flowshop de permutation est de taille $N!$, où N est le nombre de jobs à ordonnancer. Plus N est grand et plus les plateaux risquent d'être nombreux et larges donc coûteux à explorer entièrement. Dans le cas, où VEGAS se trouve sur l'un de ces plateaux, il est nécessaire de prévoir un processus qui autorise un nombre de bras maximum au bandit manchot et qui stoppe la recherche dans le cas où trop de solutions ont été visitées sur le même plateau. Ainsi, on ajoute un paramètre A

qui correspond à la valeur maximale du nombre de bras autorisés sur le bandit manchot et P le nombre maximal de solutions qu'il est possible de visiter sur un même plateau.
Lors de la recherche, tant que le bandit ne possède pas A bras, toute nouvelle solution neutre trouvée devient un nouveau bras du bandit manchot. Dès lors que le nombre de bras maximal est atteint, si l'évolvabilité de la nouvelle solution neutre est meilleure que celle d'un des bras alors elle le remplace sinon cette solution est oubliée. Quant à l'exploration du plateau, elle se poursuit jusqu'à ce qu'un nombre P de solutions ait été évaluées sur le plateau. Dans cette situation, la recherche est alors redémarrée d'une solution choisie aléatoirement dans l'espace de recherche.

4.4.3 Expérimentations

Cette version de VEGAS pour les problèmes de grandes tailles (dont le flowshop de permutation) possède trois paramètres :
– C, qui règle le compromis entre exploration et exploitation des solutions des plateaux ;
– P, qui correspond au nombre maximal de solutions évaluées sur un même plateau ;
– A, qui autorise un nombre maximal de bras au bandit manchot.
Dénotons $\text{VEGAS}_{c,p,a}$, la configuration de VEGAS où $C = c$, $P = p$ et $A = a$.

4.4.3.1 Design expérimental

La première instance aléatoire de Taillard pour chaque taille de problème (nombre de jobs $N \in \{20; 50; 100; 200; 500\}$ et nombre de machines $M \in \{5; 10; 20\}$) a été utilisée pour tester VEGAS sur le flowshop. Le voisinage d'une solution est défini par l'opérateur *insertion*. Le critère d'arrêt est le nombre d'évaluations égal à 2.10^7. Comme la méthode peut s'arrêter avant d'avoir atteint ce nombre d'évaluations maximal quand tous les bras sont visités, VEGAS est intégré dans une recherche locale itérée qui redémarre d'une solution aléatoire.
Des expérimentations préliminaires ont permis de montrer que analyser les performances de VEGAS avec C prenant ses valeurs dans l'ensemble $\{0.01, 1, 100\}$ suffisait pour répondre à la question du compromis entre exploration et exploitation.
Le nombre maximal de solutions autorisées à être évaluées sur un plateau est P. Ce nombre se rapproche du paramètre MNS de la méthode NILS (*cf.* section 3.4.2). Trois valeurs (P_1, P_2, P_3) ont été prises pour paramétrer VEGAS. Nous avons aussi testé dans le cas où un nombre infini de solutions peut être visité. En pratique, ceci implique que la valeur du paramètre est égale au critère d'arrêt (P_4). Les valeurs sont choisies en fonction du nombre de jobs :
– pour $N = 20$, $P \in \{20, 50, 100, 2.10^7\}$
– pour $N = 50$, $P \in \{150, 900, 1800, 2.10^7\}$
– pour $N = 100$, $P \in \{5000, 20000, 40000, 2.10^7\}$
– pour $N = 200$, $P \in \{4000, 30000, 60000, 2.10^7\}$
– pour $N = 500$, $P \in \{15000, 80000, 150000, 2.10^7\}$
Le nombre de bras maximal A correspond en fait au nombre de solutions d'un plateau qui peuvent nous permettre d'obtenir une porte. Se basant sur l'analyse de la structure neutre de la section 3.2.1, la valeur médiane et moyenne du nombre de pas neutres pour atteindre une porte se déplaçant sur le plateau est *a priori* un bon indicateur pour la valeur du

nombre de bras. Nous avons aussi testé des choix plus arbitraires, à savoir le nombre de jobs de l'instance car la taille du voisinage en est dépendant, 100 et 3 qui sont des valeurs tests qui permettent de conclure plus finement. On note, pour chaque instance :
– A_1, la moyenne du nombre de solutions à visiter sur le plateau pour atteindre une porte ;
– A_2, la médiane du nombre de solutions à visiter sur le plateau pour atteindre une porte ;
– A_3, le nombre de jobs ;
– A_4, la valeur 100 ;
– A_5, la valeur 3 ;
où les valeurs de A_1 et A_2 sont issues de l'analyse de paysage (*cf.* section 3.2.2).

4.4.3.2 Résultats expérimentaux

Pour présenter les performances de VEGAS, on note VEGAS$_*$ la configuration de VEGAS qui obtient la meilleure performance pour chacune des instances. Pour présenter les résultats, on utilise le triplet (c, p, a) pour expliquer les performances d'un jeu de paramètres (C, P, A).

Tableau 4.4 – Meilleures performances de VEGAS$_*$ pour les instances de Taillard selon les nombres de jobs (N) et de machines (M). Les valeurs en gras indiquent que le best-known est atteint par au moins une configuration de VEGAS. Δ représente en pourcentage les déviations non nulles entre les valeurs de VEGAS et les best-known.

Instances	M	N	Best-known	VEGAS$_*$	$\Delta_{(\%)}$
20/ 5/01	5	20	1278	**1278**	-
50/ 5/01		50	2724	**2724**	-
100/ 5/01		100	5493	**5493**	-
20/10/01	10	20	1582	**1582**	-
50/10/01		50	2991	3024	1.10
100/10/01		100	5770	**5770**	-
200/10/01		200	10862	10870	0.07
20/20/01	20	20	2297	**2297**	-
50/20/01		50	3847	3886	1.01
100/20/01		100	6202	6292	1.45
200/20/01		200	11181	11320	1.24
500/20/01		500	26059	26280	0.85

4.4.3.2.1 Performances de VEGAS Le tableau 4.4 présente pour chaque instance la meilleure valeur trouvée par l'une des configurations de VEGAS, notée VEGAS$_*$. Rappelons que le flowshop est un problème de minimisation. De plus, il est important de souligner que les best-known correspondent à des solutions généralement trouvées par des méthodes exactes ou des métaheuristiques différentes selon la taille et l'instance considérées. Ces résultats montrent qu'au moins une configuration de VEGAS atteint le best-known pour les instances dites plus faciles (5- machines ou 20- jobs). De même, au moins une configuration

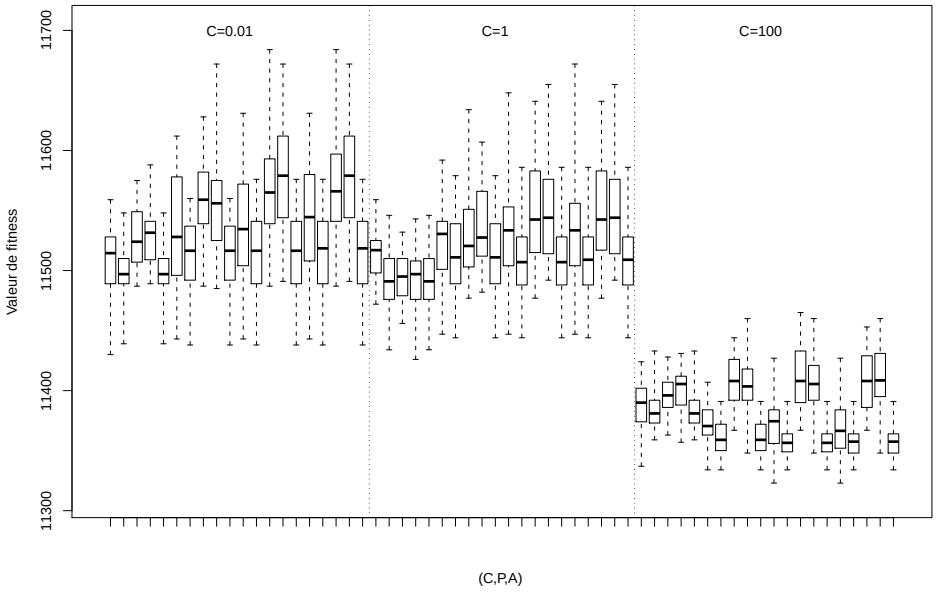


Figure 4.6 – Boîtes à moustaches des 30 valeurs de fitness trouvées après 2.10^7 évaluations, pour les 60 paramétrages possibles (C, P, A) de l'instance 200/20/01.

de VEGAS trouve le best-known de l'instance 100- jobs 10- machines qui est une instance assez difficile étant donnés la taille de l'espace de recherche et le nombre d'optima locaux (*cf.* section 2.3.1). Lorsque VEGAS ne trouve pas le best-known, on remarque avec les valeurs des déviations aux best-known que ses performances sont en réalité très correctes.

4.4.3.2.2 Impact du paramètre C C est le paramètre principal de VEGAS qui règle le compromis entre exploration et exploitation des solutions du plateau. La figure 4.6 présente les performances pour tous les couples (C, P, A) possibles de l'instance 200-jobs 20- machines. Les résultats sont présentés en trois groupes différents : (i) VEGAS-exploitation pour $C = 0.01$, pour une configuration qui favorise les solutions évolvables du plateau, (ii) VEGAS$_{C=1}$ pour une configuration qui oscille entre exploration et exploitation des solutions connues du plateau et, (iii) VEGAS-exploration pour $C = 100$ pour une configuration qui favorise les solutions peu visitées du plateau. Clairement, les configurations de VEGAS où $C = 100$ donnent de bien meilleurs résultats que celles où $C = 0.01$ ou $C = 1$. Ces résultats incitent donc à guider l'exploration des plateaux en choisissant parmi les solutions peu visitées. Tout comme pour les paysages-NKq, l'évolvabilité choisie n'aide pas ici à guider l'exploration des plateaux. Ces études montrent donc que cette mesure est difficile à définir.

4.4.3.2.3 Impact des paramètres A et P Les performances de VEGAS sont analysées en fonction des paramètres P et A qui respectivement contrôlent le nombre de

solutions visitées sur un même plateau et le nombre de bras du bandit manchot. Pour ceux-ci, on ne considère que les triplets $(100, p, a)$ car les performances obtenues avec $C = 0.01$ et $C = 1$ ne sont pas satisfaisantes. Il existe donc vingt paramétrages possibles (4 valeurs de P fois 5 valeurs de A). Tous ces paramétrages (triplets $(100, p, a)$) ont été comparés deux à deux avec le test des rangs non-paramétrique apparié de Wilcoxon. Le tableau 4.5 donne les résultats de ces tests.
Clairement, les configurations de VEGAS où $P = P_2, P_3$ (un nombre assez grand de solutions du plateau peut être visité) donnent les meilleurs résultats. En effet, les performances sont en moyenne égales ou supérieures à celles de $P = P_1$ ou $P = P_4 = 2.10^7$. En conclusion, il est nécessaire de limiter la valeur du paramètre P, . Ainsi, la recherche évite de se perdre sur des trop grands plateaux.
Le tableau 4.5 montre que certains des paramétrages où $A = A_1$, $A = A_2$ ou $A = A_5 = 3$ donnent de meilleures performances que les autres (résultats grisés). Les résultats obtenus avec $A = A_2$ sont plus foncés, car ce paramétrage est plus souvent meilleur à tous les autres. A_5 est une valeur arbitraire mais qui est proche de la plupart des médianes du nombre de solutions à visiter sur un plateau pour trouver une porte. C'est pourquoi, ce paramétrage donne de relativement bonnes performances. Ici, on observe que l'étude de la structure du paysage aide à paramétrer VEGAS, en donnant l'ordre de grandeur de la valeur pour un bon paramétrage.

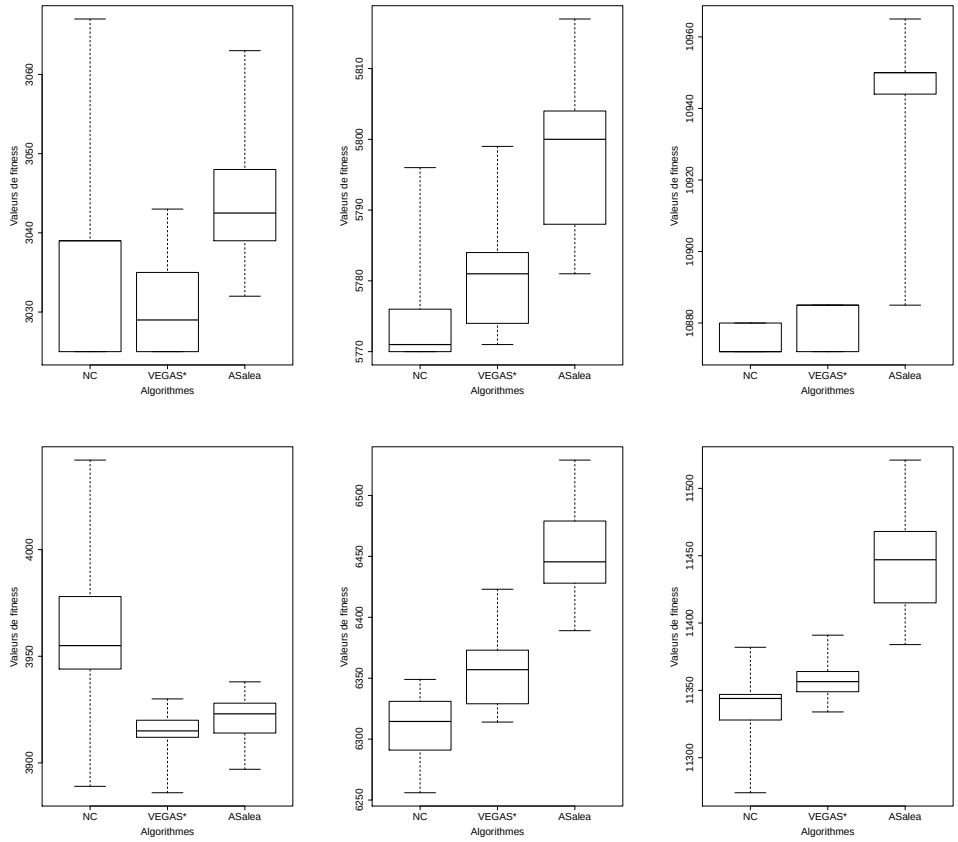


Figure 4.7 – Comparaison des performances du Netcrawler (NC) de $\text{VEGAS}_{100,P_3,A_2}$ et de AS_{alea} pour les instances de Taillard, de gauche à droite, 50-, 100- 200- jobs 10- machines (haut) 20- machines (bas).

Tableau 4.5 – Résultats des tests de Wilcoxon entre les différents paramétrages (C, P, A).

		P_1					P_2					P_3					P_4			
		A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4
P_1	A_2	=																		
	A_3	=	=																	
	A_4	=	=	=																
	A_5	=	=	=	=															
P_2	A_1	=	=	=	=	=														
	A_2	=	=	=	>	=	=													
	A_3	=	=	=	=	=	<	<												
	A_4	=	=	=	=	=	=	=	=											
	A_5	>	=	=	=	=	=	=	>	=										
P_3	A_1	=	=	=	=	=	=	=	>	=	=									
	A_2	>	>	>	>	=	=	=	>	>	=	=								
	A_3	=	=	=	=	=	=	=	=	=	=	=	<							
	A_4	=	=	=	=	=	=	<	=	=	=	=	<	=						
	A_5	>	=	=	=	=	=	=	>	=	=	=	=	>	=					
P_4	A_1	=	=	=	=	=	=	<	=	=	=	=	<	=	=	=				
	A_2	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=			
	A_3	=	<	=	=	<	<	<	=	<	<	<	<	=	<	<	<	<		
	A_4	=	<	=	=	<	<	<	=	<	<	<	<	=	<	<	<	<	=	
	A_5	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	>	>

4.4.4 Comparaison avec la littérature

D'après le tableau 4.5, le meilleur paramétrage de VEGAS parmi ces expérimentations est $C = 100$, $P = P_3$ et $A = A_2$ (la médiane du nombre de solutions à visiter pour atteindre une porte en se déplaçant sur un plateau). C'est donc avec ce paramétrage que VEGAS est comparé au Netcrawler (NC) et à AS_{alea} où une solution est choisie au hasard parmi celles connues du plateau pour qu'un de ses voisins soit évalué, le nombre maximal de bras et de solutions visitées sur un même plateau sont paramétrés comme pour VEGAS.
Le tableau 4.7 présente les 30 valeurs obtenues pour NC, VEGAS et AS_{alea} pour les instances 50-, 100-, 200- jobs 10-, 20- machines. Clairement, pour toutes les instances, AS_{alea} a de très mauvaises performances par rapport à VEGAS. Il est donc nécessaire d'avoir une méthode non aléatoire pour guider l'exploration du plateau.
VEGAS est meilleur que le Netcrawler uniquement pour les instances 50- jobs. Ici, on observe la difficulté à bien définir le guidage de VEGAS.

4.4.5 Synthèse

Cette étude sur les performances de VEGAS sur le problème de flowshop montre que l'analyse de paysage est utile pour connaître l'ordre de grandeur optimal pour les valeurs des paramètres P et A. Cependant, la mesure d'évolvabilité choisie ne permet pas encore d'atteindre les performances escomptées car les meilleures performances de VEGAS sont pour $C = 100$. VEGAS est tout de même une métaheuristique générique relativement performante car pour les instances, dites faciles, le best-known est toujours trouvé et pour les autres instances, la déviation par rapport au best-known reste très faible.
De plus, ces expérimentations montrent que considérer plusieurs solutions sur un même plateau est efficace s'il existe une stratégie de sélection d'une solution prometteuse.

4.5 Conclusion

Ce chapitre est né des observations sur la structure neutre que présente certains problèmes d'optimisation. Pour ce type de problème, les expérimentations du chapitre 3 ont montré que les recherches locales tenant compte de la neutralité, tel le Netcrawler, sont efficaces. Le Netcrawler accepte toute solution améliorante ou équivalente à la solution courante. Il se déplace donc le long des plateaux mais en ne considérant que la dernière solution acceptée. Dans ce chapitre, nous avons proposé une recherche locale qui explore les plateaux et qui, sur chacun de ces plateaux, tient compte de plusieurs des solutions trouvées.
Les plateaux sont larges. Il est nécessaire de pouvoir différencier les solutions par leur qualité à guider la recherche vers une porte du plateau. L'évolvabilité est définie comme la capacité d'une solution à produire des voisins performants. Dans cette étude, nous reprenons le concept d'évolvabilité comme la capacité qu'a une solution à mener rapidement à une porte.

VEGAS est une recherche locale qui explore le plateau par une méthode de guidage. Ainsi, dès qu'un voisin de qualité est trouvé, il devient une nouvelle solution à exploiter du plateau. VEGAS fait un choix entre toutes ces solutions pour tester l'une d'entre elles et évaluer alors l'un de ses voisins. Ce choix est guidé par une méthode de sélection issue

d'une stratégie de sélection d'opérateurs dans les algorithmes évolutionnaires. Un bandit manchot à plusieurs bras, correspondant aux solutions connues du plateau, est appliqué via la stratégie de l'UCB qui affecte un score à tous ces bras. Nous avons proposé d'utiliser une mesure d'évolvabilité pour calculer ce score pour que la meilleure solution pour conduire rapidement à une porte soit choisie. L'évolvabilité n'est qu'une mesure empirique et le calcul du score fait intervenir un bruit qui peut choisir, par le biais d'un paramètre C, plutôt une solution peu testée. Quand $C < 1$, le score favorise les solutions les plus évolvables : dans le cas contraire, $C > 1$, le score favorise des solutions peu testées. VEGAS est une recherche locale où le paramètre C règle le compromis entre l'exploitation ($C < 1$) et l'exploration ($C > 1$) des solutions du plateau.

Dans le chapitre 3, nous avons observé que les problèmes de flowshop et paysages-NKq présentent une structure neutre. VEGAS a été expérimenté sur ces deux problèmes. En premier, sur les paysages-NKq pour essayer de comprendre cette nouvelle dynamique proposée, puis sur le flowshop, qui est un problème plus proche du réel. Sur les paysages-NKq, nous avons réutilisé une mesure issue de la sélection de l'opérateur (aire sous la courbe) car elle donne de très bon résultat pour ce type de problème. Les performances de VEGAS sont souvent meilleures que celles du Netcrawler : guider la recherche pour explorer les plateaux semble assez favorable. De plus, VEGAS avec un compromis $C > 1$ est plus performant qu'avec un compromis $C < 1$. Ceci nous amène à conclure que la mesure d'évolvabilité choisie pour cette étude n'était pas la plus avantageuse.
VEGAS a été expérimenté sur les instances aléatoires de Taillard pour le flowshop en se servant d'une mesure d'évolvabilité fréquemment utilisée dans la littérature. Ces instances présentant un espace de recherche exponentiellement grand, nous avons décidé de limiter le nombre de solutions à tester (bras) et le nombre de solutions à visiter sur un même plateau respectivement réglé par les paramètres A et P. Pour ces deux paramètres, des valeurs arbitraires et d'autres issues de l'analyse de paysage (*cf.* section 3.2) ont été testées. Les expérimentations ont révélé que l'analyse de paysage permet de connaître un ordre de grandeur efficace pour les valeurs de ces paramètres.
Comme pour les paysages-NKq, VEGAS avec un compromis $C > 1$ est plus performant ce qui montre une fois encore que la mesure d'évolvabilité choisie n'est pas la bonne. De plus, les expérimentations montrent que le guidage aléatoire ne mène pas à de bons résultats et que VEGAS a des performances difficiles à comparer à celles du Netcrawler, surtout pour les instances 100-, 200- jobs.

VEGAS est une méthode relativement efficace sur les deux problèmes mais qui pourrait être améliorée en utilisant une mesure d'évolvabilité plus spécifique au problème. D'autres indicateurs d'analyse de paysage pourraient être développés pour aider à la définition de cette mesure.

Chapitre 5

Discussion

Sommaire

Les travaux de ce mémoire de thèse ont pour objectif de montrer l'intérêt d'analyser et d'utiliser la structure du paysage d'un problème d'optimisation combinatoire pour concevoir des recherches locales efficaces. Ces travaux ont été menés sur deux problèmes de logistique (le problème de flowshop de permutation et un problème de tournées de véhicules) et un problème académique (les paysages-NKq).

Après avoir décrit le contexte scientifique de la thèse et fourni un état de l'art des indicateurs existants pour étudier le paysage d'un problème, nous avons montré que les performances d'une métaheuristique s'expliquent par la structure de paysage associée. Puis, nous nous sommes plus particulièrement intéressés à la structure neutre que présentent certains problèmes. Ainsi, à partir de l'étude des paysages neutres, nous avons conçu deux recherches locales ayant des dynamiques différentes mais toutes deux tirant parti de la neutralité.

Dans ce dernier chapitre, nous synthétisons les principales contributions de cette thèse, puis une réflexion sur ces contributions nous amène à décrire une nouvelle recherche locale exploitant les qualités de celles déjà présentées dans ce mémoire pour les problèmes neutres. Enfin, nous discutons des différentes perspectives pour nos futurs travaux de recherche.

5.1 Contributions

Les contributions de ce mémoire sont organisées en cinq points.

5.1.1 Performances et analyse de paysage

Nous avons proposé une méthodologie pour analyser le paysage d'un problème d'optimisation et nous avons mené cette analyse sur deux problèmes issus de la logistique : un problème de tournées de véhicules (instances asymétriques avec flotte fixe et hétérogène avec contraintes de capacité) et le problème de flowshop de permutation (instances de Taillard). Pour chacun de ces deux problèmes, deux paysages, différenciés par leur opérateur de voisinage, et les performances de recherches locales suivant l'un ou l'autre des paysages ont été étudiées.

Pour le problème de tournées de véhicules, nous avons observé, quelque soit le paysage étudié, que les structures sont assez proches et que les performances des recherches locales sont statistiquement identiques. La rugosité de ces paysages explique bien la raison de l'utilisation conjointe de plusieurs opérateurs de voisinage lors de la résolution, à l'aide de métaheuristiques, de tels problèmes d'optimisation.

Pour le problème de flowshop, nous avons montré que les différences mises en évidence entre les deux structures de paysages se retrouvent lors de l'étude des performances de recherches locales suivant l'un ou l'autre de ces paysages. Nous en déduisons que la comparaison des paysages est utile pour choisir l'opérateur de voisinage d'une recherche locale.

Ainsi, notre première contribution prouve l'intérêt de comparer plusieurs paysages d'un même problème avec des indicateurs bien choisis afin de prédire les performances d'une métaheuristique pour correctement la paramétrer.

5.1.2 Neutralité d'un problème

Nous avons montré que les nombreuses solutions ayant la même valeur de fitness pour le problème de flowshop amènent à une structure neutre du paysage. Cette structure présente donc de nombreux plateaux qui freinent la progression de la plupart des recherches locales. Pour y remédier, la méthode de descente *Netcrawler* [Bar01] accepte de se déplacer vers une solution équivalente en qualité. Cette recherche locale profite alors des plateaux pour se déplacer et trouver une solution améliorante. D'autre part, en étudiant les méthodes efficaces pour le problème de flowshop, nous avons identifié une méthode dédiée [RS07]. Celle-ci est présentée comme aussi, voire plus, efficace que les métaheuristiques classiques de la littérature. Cependant, une telle méthode demande beaucoup de connaissances du problème et donc de longues études en amont. Nos expérimentations ont révélé que sur les instances de Taillard pour le flowshop, le Netcrawler a des performances comparables à celles de cette méthode dédiée.
Notre deuxième contribution montre, dans le cadre des problèmes neutres, qu'utiliser la neutralité explicitement conduit à de bonnes performances. La neutralité du problème ne doit pas être ignorée par la recherche locale.

5.1.3 NILS : utiliser le plateau des optima locaux

Nous avons choisi de tirer parti de l'existence des plateaux dans le paysage de certains problèmes d'optimisation (flowshop et paysages-NKq). L'étude de la structure neutre a révélé que les optima locaux sont sur des plateaux qui présentent pour la plupart des portes, solutions ayant au moins un voisin améliorant. Nous avons conçu NILS (Neutrality-based Iterated Local Search), une recherche locale itérée qui, dès qu'un optimum local est trouvé, autorise de se déplacer neutralement sur son plateau pour un nombre de pas maximal.
Les expérimentations sur les problèmes de flowshop et paysages-NKq ont montré que NILS obtient de très bonnes performances pour ces deux problèmes. En outre, les performances de NILS sont plus robustes et, en médiane, meilleures que celles du Netcrawler. Sur les instances structurées du flowshop, NILS permet même de trouver un nouveau *best-known*.
Ainsi, notre troisième contribution porte sur l'utilisation de la structure du paysage pour concevoir une première recherche locale efficace profitant explicitement de la neutralité des optima locaux.

5.1.4 VEGAS : utiliser plusieurs solutions d'un même plateau

Choisir la dernière solution trouvée comme solution courante est arbitraire. Aussi, nous avons choisi de considérer plusieurs solutions d'un même plateau plutôt qu'une solution unique lors du déplacement d'une recherche locale sur un plateau. Pour ce faire, nous avons proposé d'utiliser le concept de la mesure d'évolvabilité pour guider une recherche locale lors de son déplacement sur un plateau. Les solutions connues d'un même plateau représentent les bras d'un bandit manchot. Chaque bras est affecté d'un score, calculé à partir de son évolvabilité. Nous avons conçu VEGAS (Varying Evolvability-Guided Adaptive Search), une recherche locale qui se déplace vers la solution ayant le meilleur score pour évaluer un de ses voisins. VEGAS possède un paramètre qui règle le compromis entre exploitation d'une solution du plateau et l'exploration des solutions du plateau.

Sur le problème des paysages-NKq, les performances de VEGAS sont meilleures que celles du Netcrawler. Ceci montre que pour ce problème, tenir compte de plusieurs solutions du même plateau est efficace. Sur le problème de flowshop, nous restons plus modérés. Néanmoins, un paramétrage efficace de VEGAS est obtenu grâce à l'étude des plateaux du paysage. Selon les instances, VEGAS est meilleur que le Netcrawler ou le contraire. Cependant, ces résultats montrent la difficulté à définir une bonne mesure d'évolvabilité.
Notre quatrième contribution conduit à considérer plusieurs solutions sur un même plateau plutôt qu'une unique, et de concevoir ainsi, une recherche locale se dirigeant vers une solution bien choisie du plateau.

5.1.5 Guider stratégiquement sur un plateau

Sur le problème neutre de flowshop, nous avons comparé VEGAS avec une méthode similaire qui fait son choix aléatoirement entre les solutions. Les résultats des expérimentations ont montré que, si la recherche locale considère plusieurs solutions équivalentes d'un même plateau, le choix de la solution dont l'un des voisins sera évalué ne peut pas être fait, généralement, aléatoirement.
Notre cinquième contribution insiste sur le fait que la stratégie de VEGAS ne peut être aléatoire et qu'elle doit tenir compte d'informations sur les caractéristiques de la structure du plateau.

5.2 Vers une nouvelle recherche locale...

Dans ce mémoire, nous avons présenté NILS et VEGAS, deux recherches locales tirant partie de la neutralité du paysage. Leurs dynamiques sont différentes. En effet, NILS se déplace neutralement dès qu'il trouve un optimum local alors que VEGAS considère potentiellement les plateaux de chaque solution rencontrée. Ces deux recherches locales ont été testées sur le problème de flowshop et des paysages-NKq. Sur ces deux problèmes, elles ont montré de plutôt bonnes performances comparées à la littérature. Nous proposons ici une analyse comparative de ces 2 méthodes. Le flowshop étant un problème réel de minimisation classique de la littérature que nous avons étudié tout au long de ce mémoire, nous nous focalisons dans cette partie sur ce problème. Ici, seules les instances de Taillard avec 50-, 100-, 200- jobs 10-, 20- machines seront étudiées car elles sont représentatives des instances réelles. Dans cette section, nous commençons par comparer les performances de NILS et VEGAS sur le problème du flowshop ce qui nous amène à réfléchir et présenter une ébauche d'une nouvelle méthode tirant partie des qualités de chacune d'elles.

5.2.1 NILS *vs.* VEGAS sur le problème de flowshop

NILS est une recherche locale itérée qui améliore la solution courante au moyen d'un *First Improvement Hill Climbing* puis qui, dès qu'un optimum local est trouvé, se déplace neutralement jusqu'à trouver une porte pour recommencer. Le nombre de déplacements neutres autorisés est réglé par le paramètre MNS. Dans le cas où aucune porte n'est trouvée après MNS pas neutres, la solution subit quelques transformations et la méthode de descente est de nouveau appliquée. Sur le problème de flowshop, plusieurs valeurs

de MNS ont été testées suivant la taille de l'instance. NILS$_*$ correspond à la meilleure configuration MNS de NILS selon l'instance considérée.
VEGAS est une recherche locale qui choisit la solution courante parmi les solutions connues du plateau courant, pour évaluer un de ses voisins. La particularité de VEGAS tient à sa stratégie de sélection de la solution courante, basée sur un score dont le paramètre C règle le compromis exploitation/exploration. Ce dernier favorise l'exploitation de la meilleure solution du plateau ou au contraire l'exploration des nouvelles solutions connues du plateau. À cause de la taille de l'espace de recherche pour les instances considérées du problème de flowshop, deux paramètres ont été ajoutés à VEGAS : P le nombre maximal de solutions recherchées sur un même plateau et A le nombre maximal de ces solutions candidates pour qu'un de leur voisin soit évalué. Quand le nombre de solutions découvertes sur un même plateau dépasse la valeur de P, VEGAS est redémarré d'une nouvelle solution de l'espace de recherche. Les expérimentations du chapitre 4 ont d'ailleurs montré que la valeur de ces paramètres pouvait être donnée par l'analyse de paysage. VEGAS$_*$ correspond à la meilleure configuration (C, P, A) de VEGAS selon l'instance considérée.
La figure 5.1 présente les performances de NILS$_*$ et VEGAS$_*$ sur six des instances de Taillard. La médiane de NILS pour les 30 performances obtenues sur chacune des instances est toujours meilleure. Seule pour l'instance 50- jobs 20- machines, VEGAS trouve une solution de meilleure valeur de fitness que NILS. Néanmoins, on peut conclure que NILS est meilleur que VEGAS pour le problème de flowshop tout au moins.

Le présent chapitre a pour but de prendre du recul sur les travaux présentés au cours de ce mémoire et de comprendre les défauts de VEGAS pour tenter de les corriger. L'étude structurelle des instances de Taillard peut nous aider dans cette réflexion. VEGAS a une dynamique proche du Netcrawler puisque la neutralité est présente à tout niveau de la recherche. Or, le degré de neutralité est en moyenne plus élevé sur les solutions de moins bonnes qualités, à des niveaux plus haut de la recherche. VEGAS perd donc du temps à parcourir les plateaux de mauvaises qualités alors que sa dynamique lui permet de ne pas être bloqué sur un plateau. NILS, quant à lui, ne tire partie de la neutralité qu'au niveau des optima locaux. VEGAS ne pourrait-il pas considérer uniquement les plateaux des optima locaux ?
De même, les expérimentations du chapitre 4 ont montré que la mesure d'évolvabilité utilisée (la moyenne des valeurs de fitness de voisins) pour calculer le score de chaque solution n'est pas adaptée à ce problème. Or, la stratégie de sélection d'une solution parmi les plateaux qui utilise ce score est une adaptation d'une méthode efficace pour la sélection de l'opérateur de voisinage dans les algorithmes évolutionnaires. L'analyse de paysage ne pourrait-elle pas aider à définir une bonne mesure d'évolvabilité qui rendrait VEGAS aussi efficace qu'attendu ?

5.2.2 I-VEGAS

Dans cette section, nous présentons I-VEGAS, une recherche locale née de nos réflexions sur les qualités et défauts que présentent NILS et VEGAS. I-VEGAS se pose comme une première piste de réflexion dans la conception d'une recherche locale issue de l'analyse structurelle d'une instance d'un problème.

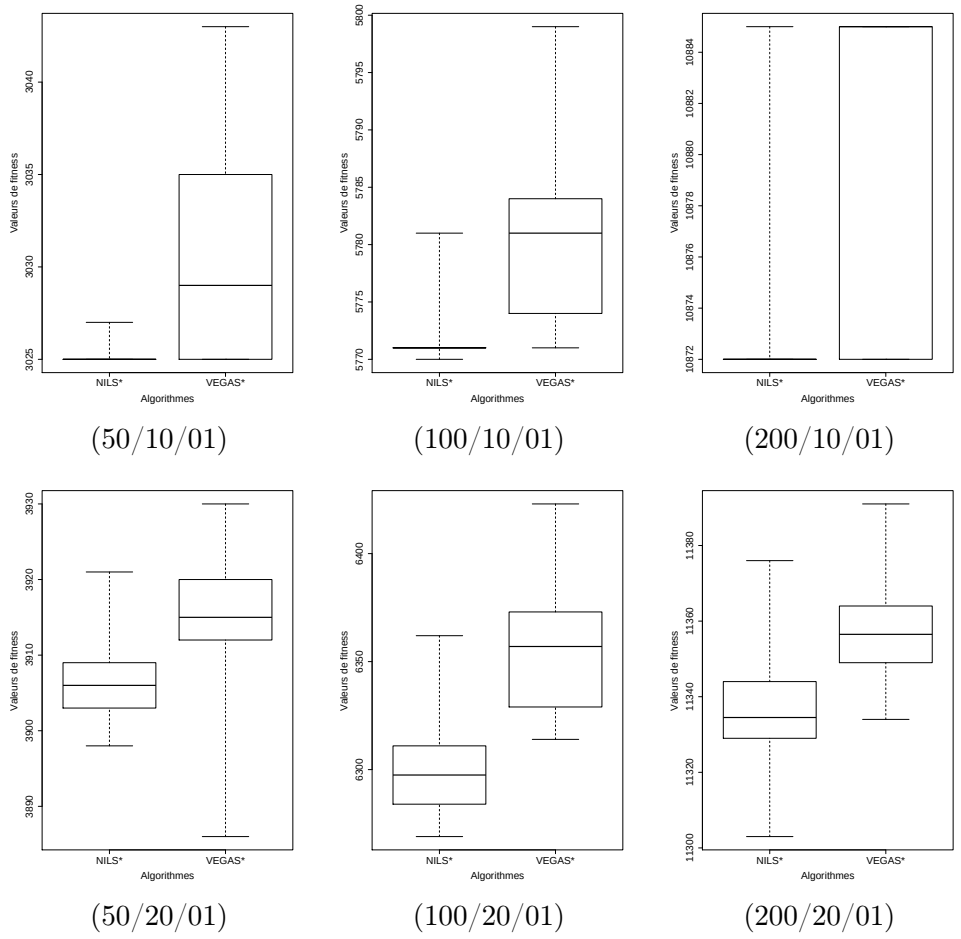


Figure 5.1 – Comparaison des performances de NILS$_*$ et VEGAS$_*$ pour les instances de Taillard, de gauche à droite, 50-, 100- 200- jobs 10- machines (haut) 20- machines (bas).

5.2.3 Description

NILS utilise le plateau de chaque optimum local trouvé pour tenter de trouver une porte puis une solution améliorante. NILS ne considère donc qu'une solution à la fois par plateau. Nous avons choisi d'utiliser le principe de VEGAS dans cette phase d'exploration du plateau. I-VEGAS est une recherche locale itérée qui améliore la solution courante par une méthode de descente jusqu'à trouver un optimum local. À partir de celui-ci, toute solution découverte du plateau est enregistrée en mémoire. La solution courante est alors choisie par la stratégie de sélection de VEGAS comme celle ayant le plus grand score pour qu'un de ses voisins soit évalué. Si un des voisins est de meilleure qualité que le plateau alors il devient la nouvelle solution courante et la méthode de descente est alors appliquée. Dans le cas où aucun voisin améliorant n'a été trouvé après que P solutions aient été visitées sur le même plateau, la solution courante subit quelques transformations et la méthode de descente est alors appliquée.

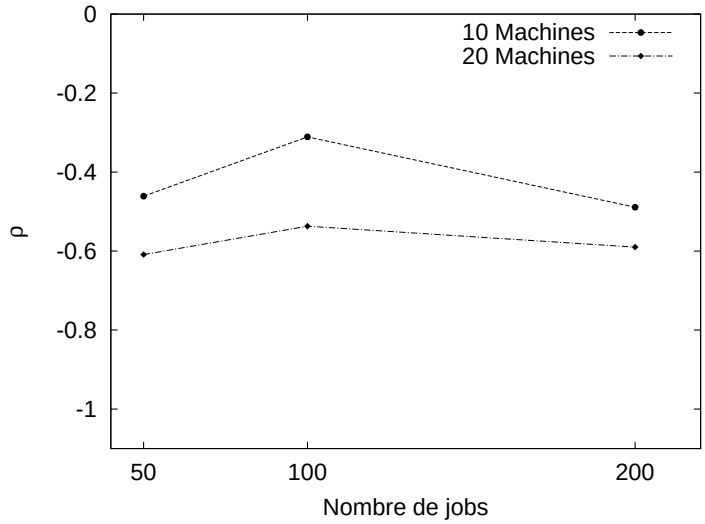


Figure 5.2 – Corrélation entre la moyenne des valeurs de fitness d'une solution et du nombre de solutions à visiter pour rencontrer une porte pour les instances 50-, 100-, 200-jobs et 10-, 20- machines.

La mesure d'évolvabilité est difficile à définir. Dans le chapitre 4, nous avons essayé d'utiliser la moyenne de valeurs de fitness des voisins d'une solution comme mesure de l'évolvabilité de celle-ci. Ainsi, dans un problème de minimisation, la solution la plus évolvable est celle pour laquelle cette moyenne est la plus faible. Or, les expérimentations nous ont montré que pour le problème de flowshop cette mesure n'est pas adaptée. Dans le chapitre 3, nous avions étudié la structure des plateaux des instances de Taillard pour le flowshop en utilisant des marches neutres qui collectent des informations au long de leur parcours. La moyenne des valeurs de fitness des voisins d'une solution est l'une de ces données, tout comme le nombre de solutions à visiter pour rencontrer une porte à partir de cette solution. Nous avons alors réutilisé ces données pour calculer la corrélation entre la moyenne des valeurs de fitness des voisins d'une solution du plateau et le nombre de solutions à visiter pour trouver une porte. La figure 5.2 présente les résultats de cette corrélation pour les six instances qui nous intéressent dans cette section. On voit ici, qu'une anti-corrélation semble se dessiner. Par conséquent, plus la moyenne est élevée et moins le nombre de solutions à visiter est faible *i.e.* plus une porte est proche. Ce résultat est contre intuitif et à l'opposé de notre utilisation initiale dans VEGAS. Cette analyse permet donc de garder la moyenne des valeurs de fitness des voisins d'une solution comme la mesure d'évolvabilité de VEGAS. Cependant, la solution la plus évolvable sera alors celle qui a la moyenne la plus forte. Ceci laisse à supposer qu'une solution ayant une moyenne élevée possède avec un probabilité plus élevée un voisin améliorant et moins de voisins neutres. Ces derniers annihilant l'effet de la mesure d'évolvabilité.

I-VEGAS a été testée sur les six instances de Taillard utilisées dans cette section. Les paramètres sont les mêmes que pour VEGAS. On a testé les valeurs de $C \in \{0.01; 1; 100\}$ et les valeurs les plus efficaces trouvées pour P et A (*cf.* section 4.4.3.2.3). Trois configurations de I-VEGAS ont donc été testées pour chaque instance. Les conditions expérimentales sont

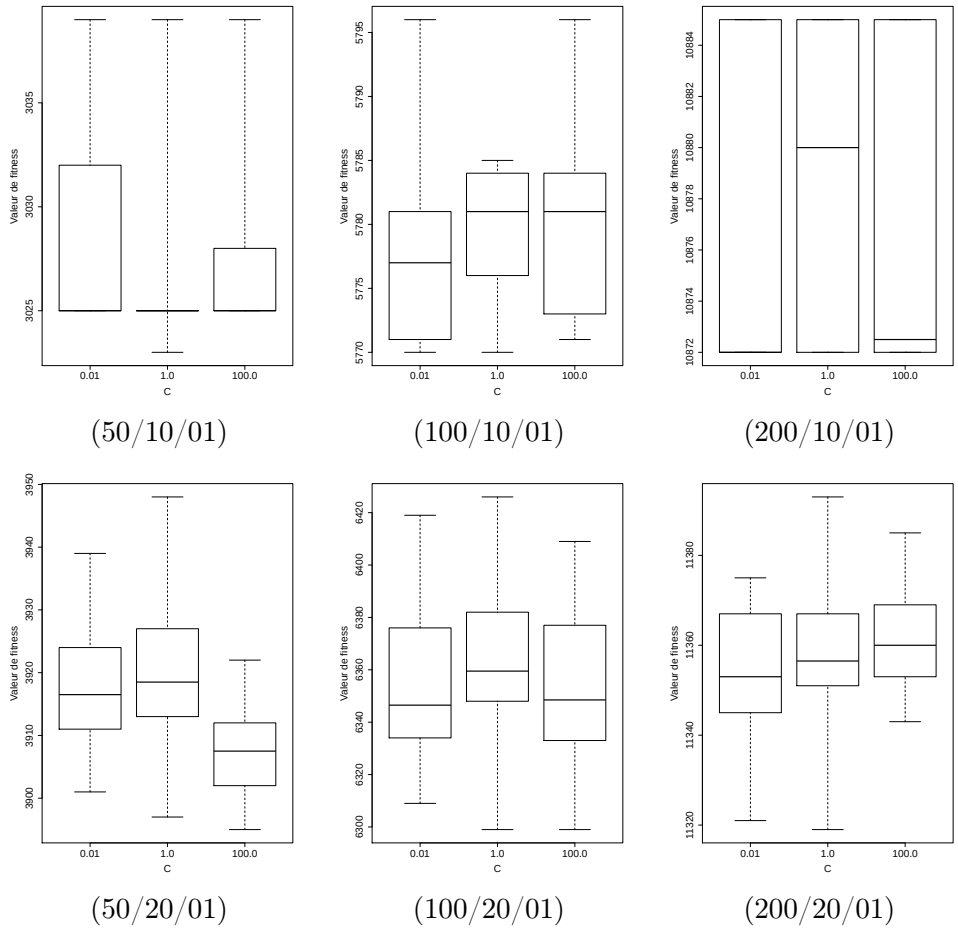


Figure 5.3 – Comparaison des performances de I-VEGAS selon les valeurs du paramètre C pour les instances de Taillard, de gauche à droite, 50-, 100- 200- jobs 10- machines (haut) 20- machines (bas).

toujours les mêmes pour que les comparaisons soient possibles. La condition d'arrêt est fixée à 2.10^7 évaluations maximales. 30 exécutions de chaque configuration de I-VEGAS ont été réalisées.

5.2.4 Résultats préliminaires

Tout d'abord, étudions les performances de I-VEGAS en fonction des valeurs de C. $C = 0.01$ (I-VEGAS-exploitation) favorise les solutions les plus évolvables du plateau, contrairement à $C = 100$ (I-VEGAS-exploration) qui favorise les solutions les moins testées du plateau. La figure 5.3 présente les performances des trois configurations de I-VEGAS pour les valeurs de C différentes. Dans tous les cas, quand $C = 1$, le choix de la solution n'est pas soumis à une réelle stratégie ce qui ne donne pas globalement de bons résultats. Pour toutes les instances, exceptée l'instance 50- jobs 20- machines, I-VEGAS exploita-

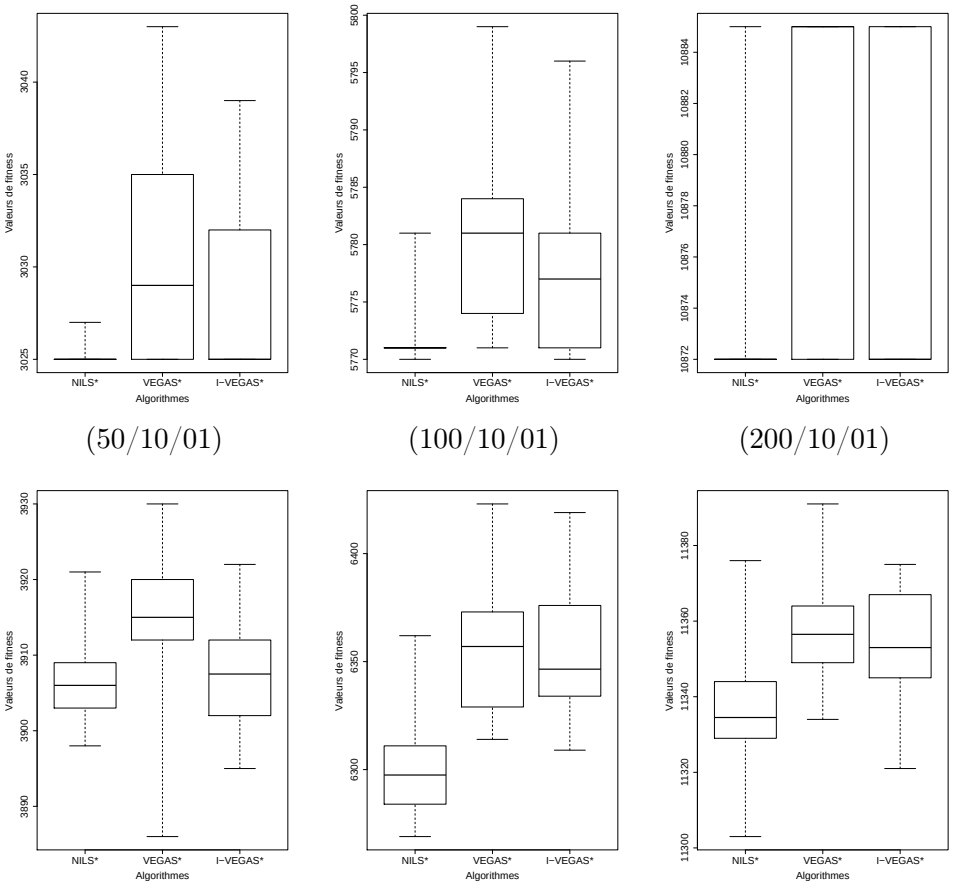


Figure 5.4 – Comparaison des performances de NILS$_*$, VEGAS$_*$ et I-VEGAS$_*$ pour les instances de Taillard, de gauche à droite, 50-, 100- 200- jobs 10- machines (haut) 20- machines (bas).

tion donne de meilleures performances médianes que I-VEGAS exploration. Ceci n'était pas le cas pour VEGAS où la mesure d'évolvabilité était mal définie par rapport à ces instances du problème de flowshop car les configurations les plus performantes de VEGAS favorisaient l'exploration des nouvelles solutions trouvées sur le plateau. Ces résultats sur les performances de I-VEGAS confirment que la mesure d'évolvabilité, encouragée par l'analyse de paysage, est mieux choisie.

Maintenant, comparons I-VEGAS avec NILS et VEGAS. La figure 5.4 compare les performances obtenues par respectivement la meilleure configuration de NILS, VEGAS et I-VEGAS pour les six instance de Taillard considérée dans cette section. Clairement I-VEGAS améliore les performances de VEGAS. Néanmoins, les performances de I-VEGAS restent en deçà de celles de NILS. Premièrement, tirer partie uniquement des plateaux des optima locaux est donc une bonne idée pour ce type de problème neutre. Deuxièmement, la stratégie de sélection est peut-être aussi à remettre en question. I-VEGAS présente

un début de réflexion sur la manière dont l'étude de paysage peut aider à concevoir une recherche locale qui se veut adaptée au problème et à sa structure.

5.3 Perspectives

Les travaux décrits dans ce mémoire ont amené à des résultats intéressants sur les liens existants entre la structure de paysage et les recherches locales efficaces. Néanmoins, ils laissent de nombreuses questions et pistes à explorer.

5.3.1 Autres problèmes neutres

Le problème de flowshop est très étudié et des méthodes spécifiques ont été développées pour le résoudre. À notre connaissance, la neutralité de sa structure n'avait jamais été montrée de façon si détaillée, ni utilisée pour le résoudre efficacement. Nous pensons qu'il doit exister parmi les problèmes de logistique ou plus largement d'optimisation combinatoire des problèmes difficiles non encore identifiés comme neutres. Un tel problème pourrait être alors résolu en utilisant une recherche locale générique (par exemple, NILS ou VEGAS) paramétrée grâce à des informations tirées de la structure de son paysage. Par exemple, les problèmes de type coloration de graphe présentent des plateaux encore assez peu exploités pour les résoudre [Mad02].

5.3.2 Analyse de Paysage *on-line*

L'analyse de la structure de paysage d'un problème reste une étape préliminaire à la compréhension puis à la conception d'une recherche locale adaptée à l'instance à résoudre elle-même. Néanmoins, cette étude structurelle prend un temps non négligeable avant de pouvoir entamer les expérimentations. Nous proposons d'introduire l'analyse du paysage dans la dynamique de la recherche locale. Ainsi, une première étape serait d'avoir des outils pour analyser rapidement le type de la structure pour démarrer avec une recherche locale basique dont les composants évolueraient grâce aux informations collectées au cours du processus d'optimisation. Nous pourrions commencer à réfléchir à la mise en œuvre d'une telle idée sur le problème de flowshop en utilisant NILS ou VEGAS.

5.3.3 Analyse de paysage pour définir l'évolvabilité

Dans la section précédente, nous avons vu comment une mesure d'évolvabilité classique de la littérature pouvait être utilisée plus efficacement grâce aux résultats de l'analyse du paysage. Or, le choix de cette mesure est difficile, surtout sur les problèmes de grandes tailles. Nous pensons que la structure de paysage du problème peut aider à définir une mesure d'évolvabilité propre à l'instance d'un problème. Ainsi, nous envisageons de chercher des indicateurs de paysage qui permettraient de caractériser l'évolvabilité de chaque solution d'un même plateau pour décrire localement leurs différences.

5.3.4 Stratégies adaptatives et guidage sur les plateaux

Nous avons adapté directement une stratégie efficace pour la sélection d'opérateurs pour les algorithmes évolutionnaires à notre problème de sélection de solutions sur un plateau. Cette stratégie dépend de la façon de calculer le score. Celle-ci n'est peut-être pas la plus adaptée à notre situation de choix d'une solution sur un plateau pour conduire rapidement à une porte et donc à un voisin améliorant. Une étude approfondie d'autres méthodes statistiques nous semble très intéressante pour obtenir une stratégie de sélection adaptée à notre situation de plateaux.

Annexe A

Échange-distance

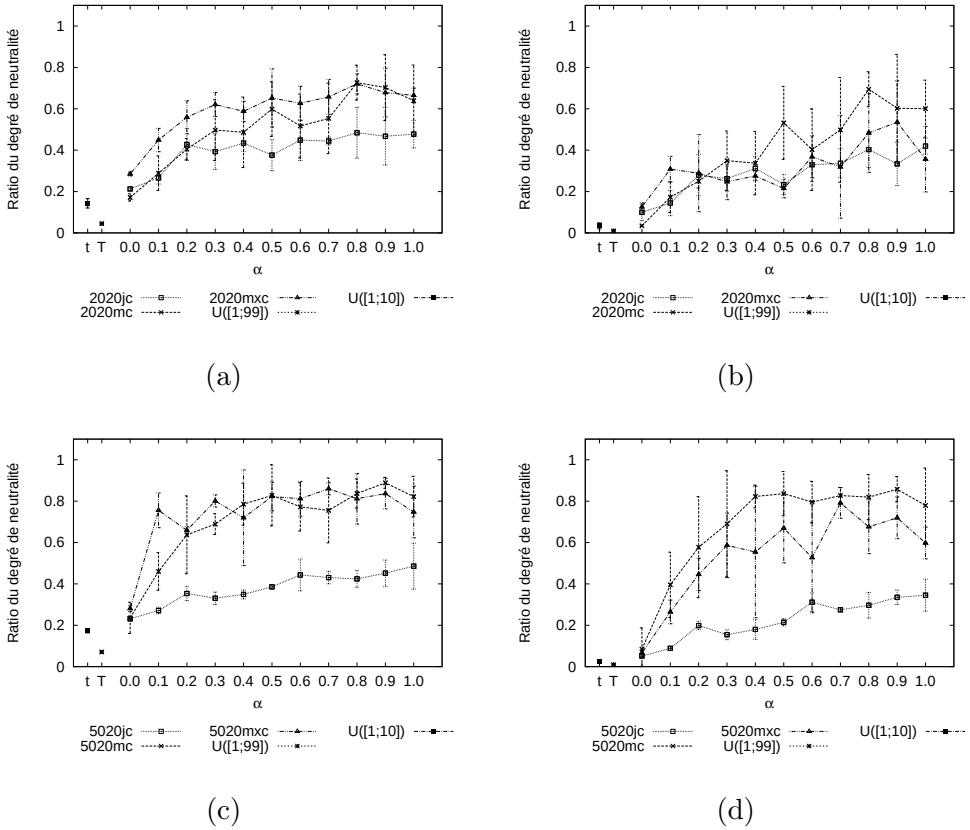
Pour un problème ayant 7 clients et une flotte de trois véhicules, l'exemple suivant illustre les étapes de l'algorithme de calcul de distance. Pour transformer $s_1 = (0\ 1\ 3\ 2\ 7\ 0\ 6\ 0\ 4\ 5\ 0)$ en $s_2 = (0\ 1\ 3\ 2\ 0\ 4\ 7\ 5\ 0\ 6\ 0)$, quatre *échange* sont nécessaires.

- $i = 1$ à 3
  $s_{1i} = s_{2i}$: les valeurs sont les mêmes dans les deux solutions.
  $distance = 0$, $list = ()$.
- $i = 4$
  $s_{1,4} \neq s_{2,4}$: 0 est attendu.
  $s_{1,4} = 7$ et $s_{2,6} = 7$.
  Or $s_{1,6} = 6 \neq 0 = s_{2,4}$
  $list$ est vide : la position 4 est enregistrée dans $list$.
  $distance = 0$, $list = (4)$.
- $i = 5$
  $s_{1,5} \neq s_{2,5}$: 4 est attendu, non pointé dans $list$.
  $s_{1,8} = 4$.
  EX$(5, 8)$ dans s_1 donne $s'_1 = (0\ 1\ 3\ 2\ 7\ 4\ 6\ 0\ 0\ 5\ 0)$.
  $distance = 1$, $list = (4)$.
- $i = 6$
  $s_{1,6} \neq s_{2,6}$: 7 est attendu, pointé dans $list$. $s_{1,4} = 7$.
  $s_{1,6} \neq 0$: $list$ reste inchangé.
  EX$(4, 6)$ dans s'_1 donne $s''_1 = (0\ 1\ 3\ 2\ 6\ 4\ 7\ 0\ 0\ 5\ 0)$.
  $distance = 2$, $list = (4)$.
- $i = 7$
  $s_{1,7} \neq s_{2,7}$: 5 est attendu, non pointé dans $list$.
  $s_{1,9} = 5$.
  EX$(7, 9)$ dans s''_1 donne $s'''_1 = (0\ 1\ 3\ 2\ 6\ 4\ 7\ 5\ 0\ 0\ 0)$.
  $distance = 3$, $list = (4)$.
- $i = 8$
  $s_{1,8} = s_{2,8}$: les valeurs sont les mêmes dans les deux solutions.
  $distance = 3$, $list = (4)$.
- $i = 9$
  $s_{1,9} \neq s_{2,9}$: 6 est attendu, pointé dans $list$. $s_{1,4} = 6$.
  $s_{1,9} = 0 = s_{2,4}$: la position 4 est enlevée de $list$.
  EX$(4, 9)$ dans s'''_1 donne $s''''_1 = s_2 = (0\ 1\ 3\ 2\ 0\ 4\ 7\ 5\ 0\ 6\ 0)$.
  $distance = 4$, $list = ()$.

Annexe B

Paysage neutre du problème de Flowshop

Instances structurées



(a) (b)

(c) (d)

Figure B.1 – Ratio du degré de neutralité moyen des instances structurées du flowshop avec 20 machines pour des solutions choisies aléatoirement (a-c) et pour les optima locaux (b-d). Les résultats sont présentés pour **20 jobs (a-b), 50 jobs (c-d)**. Les valeurs de α sont en abscisse ainsi que l'instance de Taillard $\mathcal{U}([1;99])$ (T) et l'instance "type Taillard" $\mathcal{U}([1;10])$ (t).

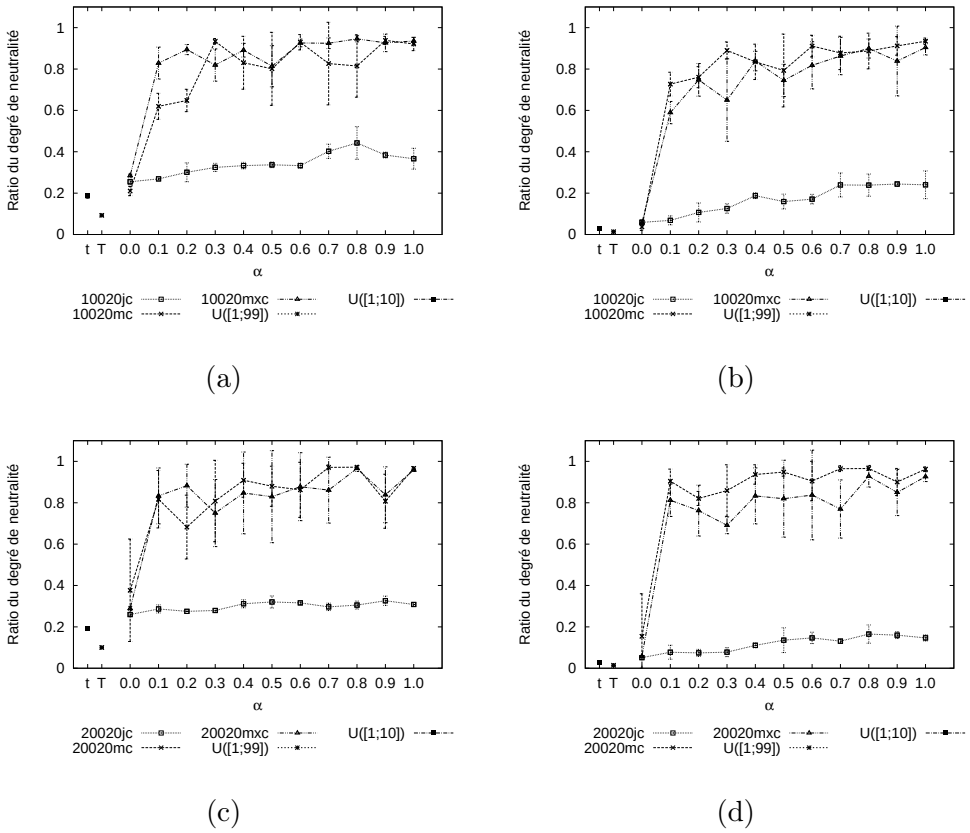


Figure B.2 – Ratio du degré de neutralité moyen des instances structurées du flowshop avec 20 machines pour des solutions choisies aléatoirement (a-c) et pour les optima locaux (b-d). Les résultats sont présentés pour **100 jobs (a-b), 200 jobs (c-d)**. Les valeurs de α sont en abscisse ainsi que l'instance de Taillard $\mathcal{U}([1;99])$ (T) et l'instance "type Taillard" $\mathcal{U}([1;10])$ (t).

Annexe C

Paysage neutre des paysages-NKq

Degrés de neutralité

Tableau C.1 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage de **solutions choisies aléatoirement** pour les paysages-NKq, $N = 128$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	41 - 32%	33 - 26%	28 - 22%	25 - 19%
3	26 - 20%	21 - 17%	17 - 14%	15 - 12%
4	19 - 15%	15 - 12%	13 - 10%	11 - 9%

Tableau C.2 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage des **optima locaux** pour les paysages-NKq, $N = 128$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	31 - 25%	19 - 14%	6 - 9%	4 - 6%
3	19 - 15%	11 - 7%	3 - 5%	2 - 4%
4	12 - 9%	8 - 5%	3 - 4%	2 - 2%

Tableau C.3 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage de **solutions choisies aléatoirement** pour les paysages-NKq, $N = 256$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	89 - 35%	67 - 26%	56 - 22%	49 - 19%
3	54 - 21%	41 - 16%	35 - 14%	30 - 12%
4	43 - 17%	31 - 12%	26 - 10%	22 - 9%

Tableau C.4 – Degré de neutralité moyen et son ratio par rapport à la taille du voisinage des **optima locaux** pour les paysages-NKq, $N = 256$ (les valeurs sont arrondies à l'entier).

K	2	4	6	8
q =2	70 - 28%	36 - 14%	22 - 8%	14 - 5%
3	39 - 15%	20 - 8%	12 - 5%	7 - 3%
4	33 - 13%	14 - 6%	8 - 3%	5 - 2%

Autocorrélation du degré de neutralité

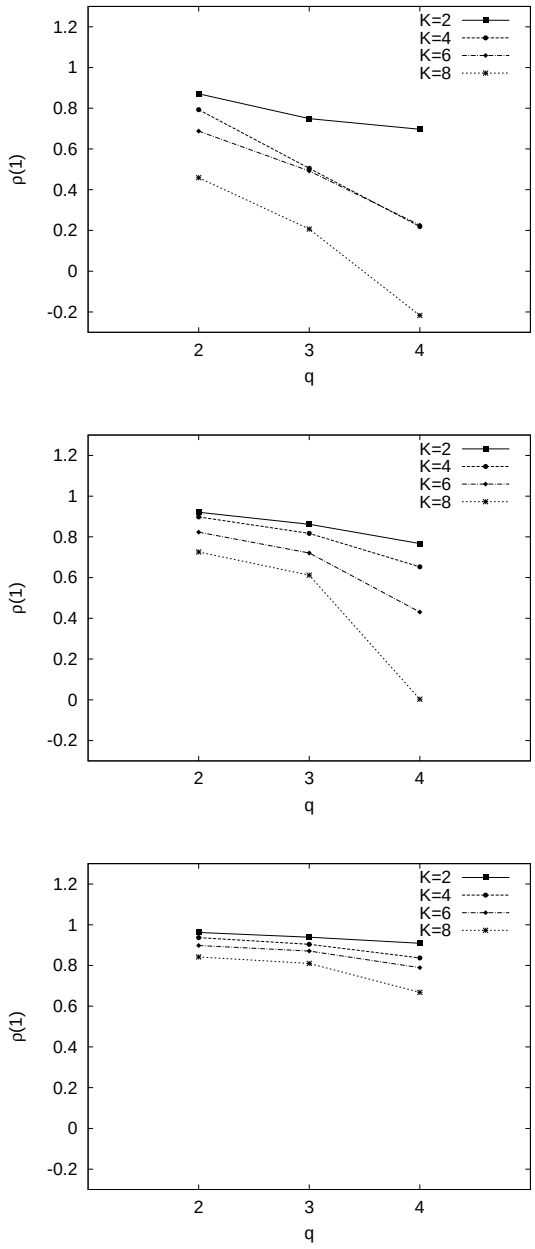
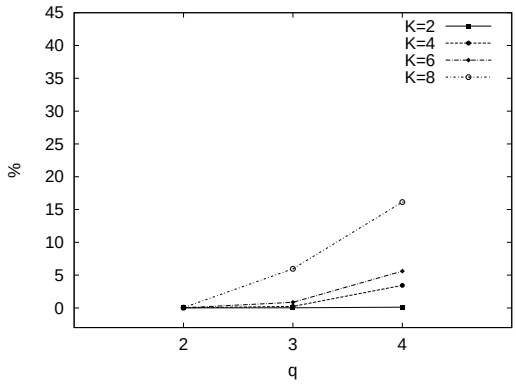
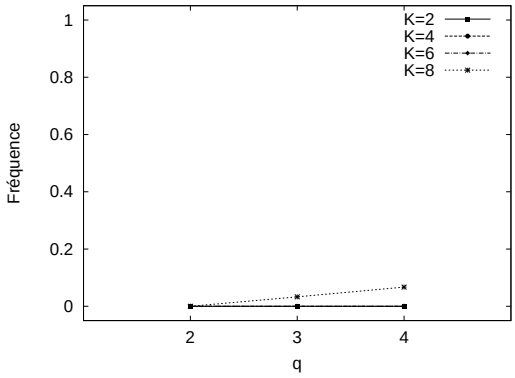


Figure C.1 – Première valeur de l'autocorrélation $\rho(1)$ du degré de neutralité entre voisins d'un plateau en fonction du nombre de jobs pour de haut en bas, $N = 64$, $N = 128$ et $N = 256$.

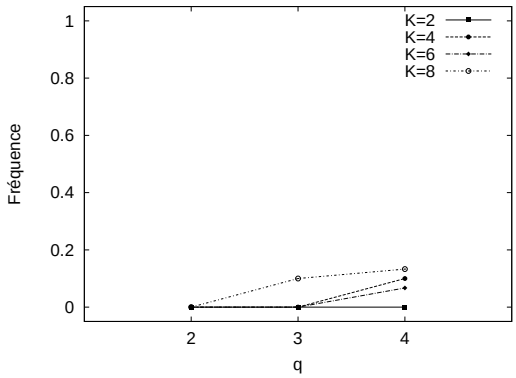
Échantillonnage et typologies des plateaux



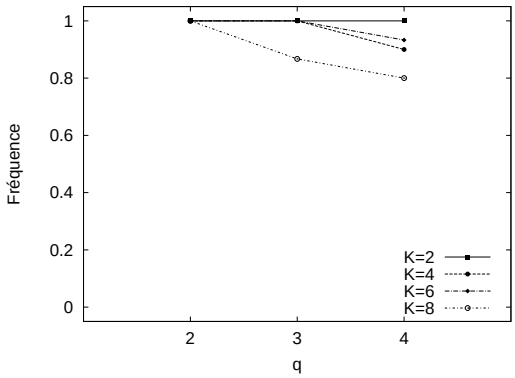
(a) Pourcentage moyen de solutions visitées au moins deux fois par les marches neutres.



(b) Fréquence moyenne des plateaux dégénérés de type T1.

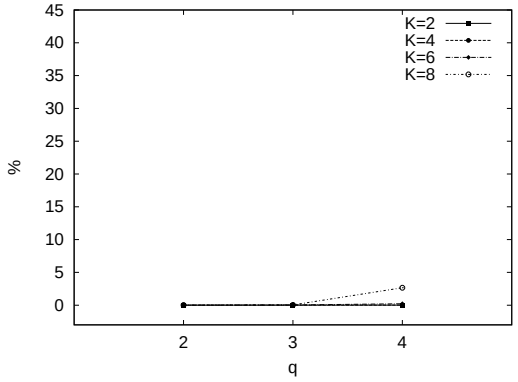


(c) Fréquence moyenne des plateaux de type T2.

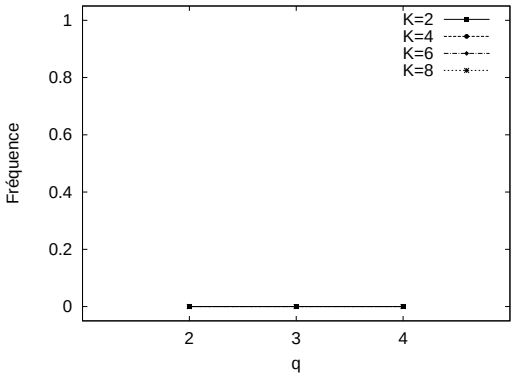


(d) Fréquence moyenne des plateaux de type T3.

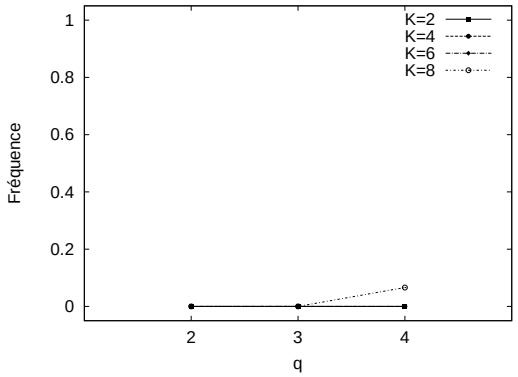
Figure C.2 – Pourcentage moyen de solutions visitées au moins deux fois et fréquences moyennes des différents type de plateaux (b,c,d), pour les paysages-NKq avec $N = 128$.



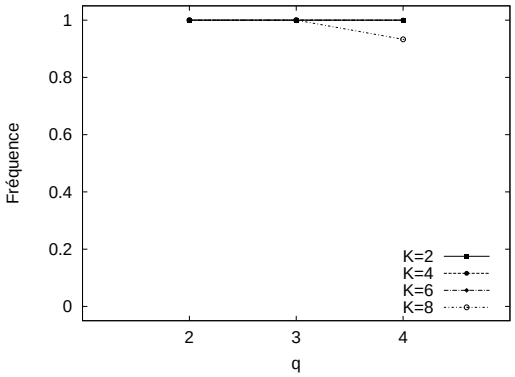
(a) Pourcentage moyen de solutions visitées au moins deux fois par les marches neutres.



(b) Fréquence moyenne des plateaux dégénérés de type T1.



(c) Fréquence moyenne des plateaux de type T2.



(d) Fréquence moyenne des plateaux de type T3.

Figure C.3 – Pourcentage moyen de solutions visitées au moins deux fois et fréquences moyennes des différents type de plateaux (b,c,d), pour les paysages-NKq avec $N = 256$.

Portes sur les plateaux et compromis coût / qualité

Tableau C.5 – T3 donne le nombre de plateaux de type T3. P_{max} donne la longueur des marches neutres. nbS donne le nombre de solutions nécessaires à visiter pour rencontrer une porte en se déplaçant sur le plateau. LgM donne la longueur de marche moyenne pour trouver les optima locaux. Toutes ces valeurs sont données pour des instances des paysages-NKq, $N = 128$.

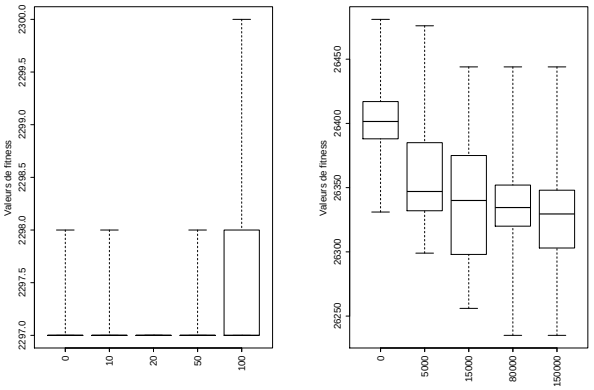
Instances		T3	nbS				LgM			
		nb	Min	Med	Moy	Max	Min	Med	Moy	Max
$q = 2$	$K = 2$	30	2	**3**	4.3	12	14	**33**	33.3	54
	$K = 4$	30	2	**3**	3.9	20	14	**32**	32.5	56
	$K = 6$	30	2	**3**	4	14	12	**30**	30.5	52
	$K = 8$	30	2	**4**	5	37	10	**29**	29.3	51
$q = 3$	$K = 2$	30	2	**3**	4.3	11	19	**43**	43.2	70
	$K = 4$	30	2	**3**	7.3	66	17	**42**	41.9	70
	$K = 6$	30	2	**3**	3.5	18	15	**39**	39.2	69
	$K = 8$	26	2	**2**	2.9	10	11	**37**	37.3	69
$q = 4$	$K = 2$	30	2	**3**	4.5	30	24	**48**	48	79
	$K = 4$	27	2	**3**	4	15	18	**47**	46.8	87
	$K = 6$	28	2	**2**	9.1	104	18	**44**	43.8	78
	$K = 8$	24	2	**2**	2.9	10	12	**41**	41.6	77

Tableau C.6 – T3 donne le nombre de plateaux de type T3. P_{max} donne la longueur des marches neutres. nbS donne le nombre de solutions nécessaires à visiter pour rencontrer une porte en se déplaçant sur le plateau. LgM donne la longueur de marche moyenne pour trouver les optima locaux. Toutes ces valeurs sont données pour des instances des paysages-NKq, $N = 256$.

Instances		T3	nbS				LgM			
		nb	Min	Med	Moy	Max	Min	Med	Moy	Max
$q = 2$	$K = 2$	30	2	**4**	6.1	25	35	**62**	62.4	90
	$K = 4$	30	2	**3**	3.8	8	35	**64**	63.8	94
	$K = 6$	30	2	**2**	3	7	32	**62**	61.7	93
	$K = 8$	24	2	**3**	5	33	31	**60**	59.7	95
$q = 3$	$K = 2$	26	2	**3**	3.6	10	51	**82**	82.3	119
	$K = 4$	27	2	**3**	4.3	13	47	**82**	82.3	127
	$K = 6$	29	2	**3**	3.3	9	43	**80**	80.4	120
	$K = 8$	23	2	**2**	2.4	5	35	**77**	77.2	122
$q = 4$	$K = 2$	24	2	**5.5**	6	23	53	**90**	90.1	134
	$K = 4$	25	2	**3**	3.3	8	54	**93**	93	140
	$K = 6$	24	2	**3**	4	16	44	**90**	90.6	141
	$K = 8$	17	2	**2**	3.1	9	40	**87**	87	139

Annexe D

NILS sur le flowshop



(20/20/01) (500/20/01)

Figure D.1 – Boîtes à moustaches des 30 valeurs de fitness trouvées pour différentes valeurs de MNS après 2.10^7 évaluations pour les instances 20/20/01 et 500/20/01. Pour les instances 20/ 5/01, 20/10/01, 50/ 5/01 et 100/ 5/01, toutes les configurations de NILS trouvent la meilleure valeur connue de la littérature, c'est pourquoi les boîtes à moustaches correspondantes ne sont pas représentées.

Bibliographie

[ACBF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47 :235–256, 2002.

[Alt94] L. Altenberg. The evolution of evolvability in genetic programming. In *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, 1994.

[AZ98] E. Angel and V. Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science*, 191 :229–243, 1998.

[Bac99] V. Bachelet. *Métaheuristiques parallèles hybrides : application au problème d'affectation quadratique*. PhD thesis, Université des Sciences et Technologies de Lille, France, 1999.

[Bar01] L. Barnett. Netcrawling - optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation*, CEC 2001, pages 30–37. IEEE Press, 2001.

[Bea83] J.E. Beasley. Route-first cluster-second methods for vehicle routing. *Omega*, 11 :403–408, 1983.

[Boe95] K. D. Boese. Cost versus distance in the traveling salesman problem. Technical report, UCLA Computer Science Dept., Los Angeles, USA, 1995.

[BPRV03] U. Bastolla, M. Porto, H. E. Roman, and M. Vendruscolo. Statistical properties of neutral evolution. *Journal Molecular Evolution*, 57(S) :103–119, 2003.

[BSUS72] W.A. Beyer, M.L. Stein, S.M. Ulam, and T. Smith. Metric in biology, an introduction. LA 4937, University of California, Los Alamos, 1972.

[CDS70] H. Campbell, R. Dudek, and M. Smith. A heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16(10) :630–637, 1970.

[CMT04] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo : A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10 :357–380, 2004.

[Cri88] D. Critchlow. Ulam's metric. *In Encyclopedia of Statistical Sciences*, 9 :379–380, 1988.

[Dan77] D. Dannenbring. An evaluation of flow shop sequencing heuristics. *Management Science*, 23 :1174–1182, 1977.

[dBS01] M. L. den Besten and T. Stützle. Neighborhoods Revisited : An Experimental Investigation into the Effectiveness of Variable Neighborhood Descent for Scheduling. In *Proceedings of the 4th Metaheuristics International Conference*, MIC 2001, pages 545–549, 2001.

[DFSS08] L. Da Costa, Á. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2008, pages 913–920. ACM Press, 2008.

[DMC96] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 26(1) :29–41, 1996.

[Dra10] B. Draskoczy. Fitness distance correlation and search space analysis for permutation based problems. In *Proceedings of the 11th European Conference of Evolutionary Computation in Combinatorial Optimization*, volume 6022 of *EvoCOP 2010*, pages 47–58. LNCS, Springer, 2010.

[EMSS07] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. 2007.

[FRP97] C. Fonlupt, D. Robilliard, and P. Preux. Fitness landscape and the behavior of heuristics. In *Proceedings of Evolution Artificielle 1997*, 1997.

[FSS10] Á. Fialho, M. Schoenauer, and M. Sebag. Toward comparison-based adaptive operator selection. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2010, pages 767–774. ACM Press, 2010.

[FTV94] M. Fischetti, P. Toth, and D. Vigo. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Annals of Operations Research*, 42(5) :846–859, 1994.

[GHL94] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40 :1276–1290, 1994.

[GJ90] M. R. Garey and D. S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[GL98] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1998.

[GLMT99] M. Gendreau, G. Laporte, C. Musaraganyi, and É. D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26 :1153–1173, 1999.

[Glo77] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1) :156–166, 1977.

[Gre87] J. J. Grefenstette. *Incorporating problem specific knowledge into genetic algorithms*, chapter Genetic Algorithms and Simulated Annealing. Morgan Kaufmann Publishers Inc., 1987.

[GW04] J. Grabowski and M. Wodecki. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research*, 31 :1891–1909, October 2004.

[GWH+02] N. Geard, J. Wiles, J. Hallinan, B. Tonkes, and B. Skellett. A comparison of neutral landscapes - NK, NKp and NKq. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC 2002, pages 205–210. IEEE Press, 2002.

[Hol92] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[IMT95] H. Ishibuchi, S. Misaki, and H. Tanaka. Modified simulated annealing algorithms for the flow shop sequencing problem. *European Journal of Operational Research*, 81(2) :388–398, 1995.

[JF95] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[JSW09] F. Jin, S. Song, and C. Wu. *Structural Property and Meta-heuristic for the Flow Shop Scheduling Problem*, volume 230 of *Studies in Computational Intelligence*, pages 1–20. Springer Berlin / Heidelberg, 2009.

[Kau93] S. Kauffman. *The Origins of Order : Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[KGH00] J. H. Kao, P. Galinier, and M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'Intelligence Artificielle*, 13(2) :283–324, 2000.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[Kou98] C. Koulamas. A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research*, 105(1) :66–71, 1998.

[KS75] W.H. Kohler and K. Steiglitz. Exact, approximate, and guaranteed accuracy algorithms for the flowshop problem n/2/F/F. *Journal ACM*, 22(1) :106–114, 1975.

[Kub07] M. Kubiak. *Distance metrics and fitness-distance analysis for the capacitated vehicle routing problem*, volume 39 of *Operations Research/Computer Science Interfaces Series*, chapter 18, pages 345–364. Springer US, 2007.

[Mad02] B. Madeline. *Algorithmes évolutionnaires et résolution de problèmes de satisfaction de contraintes en domaines finis*. PhD thesis, Université de Nice-Sophia Antipolis, France, 2002.

[MDJ+11a] M.-É. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. NILS : a neutrality-based iterated local search and its application to flowshop scheduling. In *Proceedings of the 12th European Conference of Evolutionary Computation in Combinatorial Optimization*, EvoCOP 2011, pages 191– 202. LNCS, Springer, 2011.

[MDJ+11b] M.-É. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. On the neutrality of flowshop scheduling fitness landscapes. In *Proceedings of the 5th Learning and Intelligent OptimizatioN Conference*, LION 2011, pages 238–252. LNCS, Springer, 2011.

[MDJ+11c] M.-É. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. The road to VEGAS : Guiding the search over neutral networks. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2011, pages 1979–1986. ACM Press, 2011.

[Mer00] P. Merz. *Memetic Algorithms for Combinatorial Optimization Problems : Fitness Landscapes and Effective Search Strategies*. PhD thesis, Universitat Gesamthochschule, Siegen, Deutschland, Deutchland, 2000.

[Mer04] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12 :303–325, 2004.

[MHJD10] M.-É. Marmion, J. Humeau, L. Jourdan, and C. Dhaenens. Comparison of neighborhood for the HFF-AVRP. In *Proceedings of the 8th ACS/IEEE International Conference on Computer Systems and Applications, NATCOMP workshop*, AICCSA 2010, pages 1–7. IEEE Press, 2010.

[MIT96] T. Murata, H. Ishibuchi, and H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers and Industrial Engineering*, 30 :1061–1071, 1996.

[MJD10a] M.-É. Marmion, L. Jourdan, and C. Dhaenens. A fitness landscape analysis for the permutation flowshop scheduling problem. In *International Conference on Metaheuristics and Nature Inspired Computing*, META 2010, 2010.

[MJD10b] M.-É. Marmion, L. Jourdan, and C. Dhaenens. Une nouvelle mesure de distance pour l'ACVRP. In *11e congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, ROADEF 2010, 2010.

[MJD11] M.-É. Marmion, L. Jourdan, and C. Dhaenens. Fitness landscape analysis and metaheuristics efficiency. *Journal of Mathematical Modelling and Algorithms*, 2011.

[NE98] M. Newman and R. Engelhardt. Effect of neutral selection on the evolution of molecular species. *Proceedings of the Royal Society London B.*, 256 :1333–1338, 1998.

[NEH83] M. Nawaz, E. E. Enscore, and I. Ham. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *Management Science*, 11 :91–95, 1983.

[NS96] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91 :160–175, 1996.

[OP89] I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6) :551–557, 1989.

[OS90] F. A. Ogbu and D. K. Smith. The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem. *Computers & Operations Research*, 17 :243–253, 1990.

[Pap76] C. H. Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton University, US, 1976.

[Pel10] M. Pelikan. NK landscapes, problem difficulty, and hybrid evolutionary algorithms. In Martin Pelikan and Jürgen Branke, editors, *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO 2010, pages 665–672. ACM Press, 2010.

[Pri09] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22 :916–928, September 2009.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[Ree93] C. R. Reeves. Improving the efficiency of tabu search for machine sequencing problems. *Journal of the Operational Research Society*, 44(4) :375–382, 1993.

[Ree95] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, 22 :5–13, 1995.

[Ree97] C. R. Reeves. Landscapes, operators and heuristic search. *Annals of Operations Research*, 86 :473–490, 1997.

[Rib94] C. C. Ribeiro. Applications of combinatorial optimization. *Annals of Operations Research*, 50, 1994.

[RM05] R. Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165 :479–494, 2005.

[RMA06] R. Ruiz, C. Maroto, and J. Alcaraz. Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, 34(5) :461–476, 2006.

[RMB98] R. Z. Ríos-Mercado and J. F. Bard. Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Computers & Operations Research*, 25(5) :351–366, 1998.

[RMB99] R.Z. Ríos-Mercado and J.F. Bard. A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. *IIE Transactions*, 31 :721–731, 1999.

[RS07] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3) :2033–2049, 2007.

[RY98] C. Reeves and T. Yamada. Genetic algorithms, path relinking and the flowshop sequencing problem. *Evolutionary Computation*, 6 :45–60, 1998.

[SÖ5] K. Sörensen. Path relinking for the vehicle routing problem using the edit distance. In *The 6th metaheuristics international conference*, pages 839–846. MIC2005, Vienna, Austria, 2005.

[SB92] S. Stöppler and C. Bierwirth. The application of a parallel genetic algorithm to the $n/m/P/C_max$ flowshop problem. In *New Directions for Operations Research in Manufacturing*, pages 161–175. Springer-Verlag, 1992.

[SHLO02] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10 :1–34, 2002.

[SS06] K. Sörensen and M. Sevaux. MA|PM : memetic algorithms with population management. *Computers & Operations Research*, 33 :1214–1225, 2006.

[SS07] T. Schiavinotto and T. Stützle. A review of metrics on permutations for search landscape analysis. *Computers & Operations Research*, 34 :3143–3153, 2007.

[Sta96] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20 :1–45, 1996.

[Stü97] T. Stützle. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing*, EUFIT'98, pages 1560–1564. Verlag, 1997.

[Stü98] T. Stützle. Applying iterated local search to the permutation flow shop problem. Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt, Deutschland, 1998.

[Tai90] E. Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1) :65–74, July 1990.

[Tai93] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64 :278–285, 1993.

[Tai99] E. Taillard. A heuristic column generation method for the heterogeneous VRP. *RAIRO, Operations Research*, 33(1) :1–14, 1999.

[Thi05] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, GECCO 2005, pages 1539–1546. ACM Press, 2005.

[TKV04] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, 152(1) :148–158, 2004.

[VCC04] S. Verel, P. Collard, and M. Clergue. Scuba Search : when selection meets innovation. In *Proceedings of the 2004 Congress on Evolutionary Computation*, CEC'04, pages 924 – 931. IEEE Press, 2004.

[VCC06] S. Verel, P. Collard, and M. Clergue. Measuring the Evolvability Landscape to study Neutrality. In *Poster at conference onGenetic and Evolutionary Computation*, GECCO 2006, pages 613–614. ACM Press, 2006.

[Ver05] S. Verel. *Étude et exploitation des réseaux de neutralité dans les paysages adaptatifs pour l'optimisation difficile*. PhD thesis, Université de Nice-Sophia Antipolis, France, 2005.

[VOT08] S. Verel, G. Ochoa, and M. Tomassini. The connectivity of NK landscapes' basins : a network analysis. In *Proceedings of Artificial Life Conference Alife XI*, pages 648–655, 2008.

[Wag05] A. Wagner. Robustness, evolvability, and neutrality. *FEBS Letters*, 579 :1772–1778, 2005.

[WBWH02] J.-P. Watson, L. Barbulescu, L. D. Whitley, and A. E. Howe. Contrasting structured and random permutation flow-shop scheduling problems : Search-space topology and algorithm performance. *Informs Journal on Computing*, 14 :98–123, 2002.

[Wei90] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63 :325–336, 1990.

[WH89] M. Widmer and A. Hertz. A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41(2) :186–193, 1989.

[Wri32] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the Sixth International Congress on Genetics*, volume 1, 1932.

Résumé
Les problèmes d'optimisation combinatoire sont généralement NP-difficiles et les méthodes exactes demeurent inefficaces pour les résoudre rapidement. Les métaheuristiques sont des méthodes génériques de résolution connues et utilisées pour leur efficacité. Elles possèdent souvent plusieurs paramètres qui s'avèrent fastidieux à régler pour obtenir de bonnes performances. Il est alors intéressant de chercher à rendre plus évident, voire à automatiser, ce réglage des paramètres. Le paysage d'un problème d'optimisation combinatoire est une structure, basée sur la notion de voisinage, permettant de caractériser le problème puis de suivre la dynamique d'une méthode d'optimisation pour comprendre son efficacité.
Les travaux de cette thèse portent sur l'analyse de paysage de problèmes d'optimisation combinatoire et le lien étroit avec certaines classes de métaheuristiques, basées sur une exploration du voisinage des solutions. Ainsi, nous montrons l'influence de la structure de paysage sur la dynamique d'une métaheuristique, pour deux problèmes issus de la logistique. Ensuite, nous analysons les caractéristiques du paysage qui permettent de concevoir et/ou paramétrer des métaheuristiques, principalement des recherches locales, efficaces.
La neutralité est, en particulier, une caractéristique structurelle importante des paysages. De tels paysages présentent de nombreux plateaux bloquant la progression d'une recherche locale. Après une analyse fine des plateaux, nous prouvons que cette structure neutre ne doit pas être ignorée. Puis, nous utilisons plusieurs informations liées à la neutralité, et plus particulièrement aux plateaux bloquants, pour concevoir une première recherche locale simple à mettre en œuvre et efficace. Enfin, pour approfondir nos travaux sur les structures neutres, nous avons choisi d'exploiter la neutralité à tous les niveaux du paysage pour concevoir une nouvelle recherche locale basée sur la capacité des solutions d'un même plateau à produire une amélioration. Une stratégie de guidage vers cette solution est alors proposée.
La thèse se termine par l'analyse comparative des deux méthodes d'optimisation proposées pour les problèmes neutres afin d'en exploiter de nouvelles caractéristiques, et ainsi, renforcer le lien entre l'analyse de paysage et la conception de méthodes efficaces.

Mots clés Recherche opérationnelle ; Optimisation combinatoire ; Programmation heuristique ; Métaheuristiques ; Analyse de paysage ; Neutralité ; Ordonnancement ; Problèmes de tournées

---

Abstract
Many problems from combinatorial optimization are NP-hard, so that exact methods remain inefficient to solve them efficiently. However, metaheuristics are approximation methods known and used for their efficiency. But they often require a lot of parameters, which are very difficult to set in order to provide good performance. As a consequence, a challenging question is to perform such parameter tuning easier, or adaptive. The fitness landscape of given combinatorial optimization problem, based on a search space, a fitness function and a neighborhood relation, allow to characterize the problem structure and make the understanding of the dynamics of search approches possible.
This thesis deals with fitness landscape analysis, together with the link with some neighborhood-based metaheuristic classes. We show the influence of the landscape structure on the dynamics of metaheuristics, for two challenging problems from the field of logistics. We analyze the landscape characteristics which help to design efficient local search metaheuristics and/or to set their parameters.
Neutrality is one of the main structural characteristic of a landscape. Such landscapes have numerous plateaus, which often inhibits the progress of local search algorithms. After a deep analysis of these plateaus, we prove that this neutral structure cannot be ignored. Then, we use several information linked with neutrality, and particularly with blocking plateaus, in order to design a first local search approach, which appear to efficient and easy to implement. At last, in order to extend our work on the neutral structure, we chose to exploit the neutrality involved in the whole landscape. We propose a new local search algorithm, based on the ability of solutions of a plateau to produce improvement by means of a guiding strategy.
The thesis ends with an experimental analysis of the two local search methods presented for neutral problems in order to exploit new characteristics, and then to strengthen the link between fitness landscape analysis and efficient algorithm design.

Keywords Operational Research ; Combinatorial Optimization, Heuristic Programming, Metaheuristics ; Landscape Analysis ; Neutrality ; Scheduling ; Routing