Optimisation par métaheuristique adaptative distribuée en environnement de calcul parallèle

Christopher Jankee

OPTIMISATION PAR MÉTAHEURISTIQUE ADAPTATIVE DISTRIBUÉE EN ENVIRONNEMENT DE CALCUL PARALLÈLE

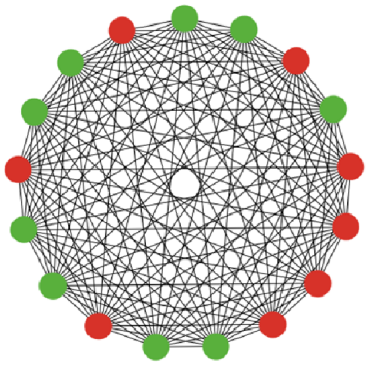
Thèse

Spécialité : Informatique

CHRISTOPHER JANKEE

Composition du jury

Président :	M. Philippe COLLARD	Professeur à université de Nice-Sophia-Antipolis
Rapporteurs :	M. Lhassane IDOUMGHAR	Professeur à université d'Haute Alsace
	M. Frédéric SAUBION	Professeur à université Angers
Directeur :	M. Cyril FONLUPT	Professeur à université du Littoral Côte d'Opale
Co-directeur :	M. Sébastien VEREL	Professeur à université du Littoral Côte d'Opale
	M.Bilel DERBEL	Maître de conférences à université de Lille



Université du Littoral Côte d'Opale

http://www-lisic.univ-littoral.fr/~jankee/

Thèse, printemps/été 2018

Soutenue le **31 août 2018** à Calais

PAGE WEB :
`http://www-lisic.univ-littoral.fr/~jankee/`

COURRIEL :
`christopher.jankee@univ-littoral.fr`

“The technical drawbacks to parameter tuning based on experimentation can be summarized as follows :

— the process of parameter tuning costs a lot of time, even if parameters are optimized one by one, regardless of their interactions ;

— for a given problem the selected parameter values are not necessarilyoptimal, even if the effort made in setting them was significant.”

“Les inconvénients des techniques de réglage des paramètres basés sur l’expérimentation peuvent être résumés comme suit :

— le processus de réglage des paramètres coûte beaucoup de temps, même si les paramètres sont optimisés un par un, indépendamment de leurs interactions ;

— pour un problème donné, les valeurs de paramètres sélectionnées ne sont pas nécessairement optimales, même si l’effort de mise en place est significatif.”

A E Eiben, Robert Hinterding and Zbigniew. Michalewicz - 2000 - Evolutionary computation 2 p 173 section parameter control. [BFMoo]

ABSTRACT

To solve discrete optimization problems of black box type, many stochastic algorithms such as evolutionary algorithms or metaheuristics exist and prove to be particularly effective according to the problem to be solved. Depending on the observed properties of the problem, choosing the most relevant algorithm is a difficult problem. In the original framework of parallel and distributed computing environments, we propose and analyze different adaptive optimization algorithm selection strategies. These selection strategies are based on reinforcement learning methods automatic, from the field of artificial intelligence, and on information sharing between computing nodes. We compare and analyze selection strategies in different situations. Two types of synchronous distributed computing environment are discussed : the island model and the master-slave model. On the set of nodes synchronously at each iteration, the adaptive selection strategy chooses an algorithm according to the state of the search for the solution.

In the first part, two problems OneMax and NK, one unimodal and the other multimodal, are used as benchmarks for this work. Then, to better understand and improve the design of adaptive selection strategies, we propose a modeling of the optimization problem and its local search operator. In this modeling, an important characteristic is the average gain of an operator according to the fitness of the candidate solution. The model is used in the synchronous framework of the master-slave model. A selection strategy is broken down into three main components : the aggregation of the rewards exchanged, the learning scheme and the distribution of the algorithms on the computing nodes. In the final part, we study three scenarios, and we give keys to understanding the relevant use of adaptive selection strategies over naïve strategies. In the framework of the master-slave model, we study the different ways of aggregating the rewards on the master node, the distribution of the optimization algorithms on the nodes of computation and the time of communication. This thesis ends with perspectives in the field of distributed adaptive stochastic optimization.

RÉSUMÉ

Pour résoudre des problèmes d'optimisation discret de type boîte noire, de nombreux algorithmes stochastiques tels que les algorithmes évolutionnaires ou les métaheuristiques existent et se révèlent particulièrement efficaces selon le problème à résoudre. En fonction des propriétés observées du problème, choisir l'algorithme le plus pertinent est un problème difficile. Dans le cadre original des environnements de calcul parallèle et distribué, nous proposons et analysons différentes stratégies adaptative de sélection d'algorithme d'optimisation. Ces stratégies de sélection reposent sur des méthodes d'apprentissage automatique par renforcement, issu du domaine de l'intelligence artificielle, et sur un partage d'information entre les nœuds de calcul. Nous comparons et analysons les stratégies de sélection dans différentes situations. Deux types d'environnement de calcul distribué synchrone sont abordés : le modèle en île et le modèle maître-esclave. Sur l'ensemble des nœuds de manière synchrone à chaque itération la stratégie de sélection adaptative choisit un algorithme selon l'état de la recherche de la solution.

Dans une première partie, deux problèmes OneMax et NK, l'un unimodal et l'autre multimodal, sont utilisés comme banc d'essai de ces travaux. Ensuite, pour mieux saisir et améliorer la conception des stratégies de sélection adaptatives, nous proposons une modélisation du problème d'optimisation et de son opérateur de recherche locale. Dans cette modélisation, une caractéristique importante est le gain moyen d'un opérateur en fonction de la fitness de la solution candidate. Le modèle est utilisé dans le cadre synchrone du modèle maître-esclave. Une stratégie de sélection se décompose en trois composantes principales : l'agrégation des récompenses échangées, la technique d'apprentissage et la répartition des algorithmes sur les nœuds de calcul. Dans une dernière partie, nous étudions trois scénarios et nous donnons des clés de compréhension sur l'utilisation pertinente des stratégies de sélection adaptative par rapport aux stratégies naïves. Dans le cadre du modèle maître-esclave, nous étudions les différentes façons d'agréger les récompenses sur le nœud maître, la répartition des algorithmes d'optimisation sur les nœuds de calcul et le temps de communication. Cette thèse se termine par des perspectives pour le domaine de l'optimisation stochastique adaptative distribuée.

REMERCIEMENTS

Cette thèse a été une aventure formidable. J'ai eu l'occasion au cours de mon travail, des collaborations et des conférences de rencontrer des personnes passionnées. Je remercie tous ceux et celle qui m'ont entouré durant ces années à échanger sur des thèmes divers

Je tiens tout d'abord à remercie mon directeur de thèse Fonlup Cyril et mes co-directeurs Sébastien Verel, Bilel Derbel avec qui j'ai partagé de nombreux moments, à me transmettre leurs passions pour la recherche, à discuté les idées, et à les remettre en perspective. Cela m'a permis de faire évoluer ma vision sur le projet de recherche. Ils m'ont soutenu jusqu'au bout dans ce projet intense. Je voudrais en particulier remercier Sébastien pour sa passion, sa gentillesse, sa façon de reposer les problèmes. Il m'a vu évoluer depuis de nombreuses années. Il m'a véritablement transmis la passion pour la recherche académique, je me souviens de ces cours captivants de système complexe, il sait faire parler les résultats scientifiques et par là même transmettre, captiver. Je le remercie ainsi que Philippe Collard de m'avoir embarqué sur la route de la recherche à travers les projets étudiants et les stages de recherches.

Je remercie les membres du jury de ma soutane de thèse en particulier les rapporteurs de ma thèse. Ils m'ont donné une nouvelle lecture de mes travaux de recherche à travers leurs questionnements et donner de nouvelle piste de recherche.

Je tiens à remercier les financeurs de cette thèse, l'école doctorale EDSPI, le Laboratoire d'Informatique Signal et Image de la Côte d'Opale, et l'université du Littoral-Côte-d'Opale. Je remercie tous les collaborateurs du laboratoire ou chacun à leurs manières a su me transmettre quelque chose. Je remercie aussi les étudiants que j'ai rencontrés durant mes enseignements ou ceux ayant eu l'occasion d'encadrer. Ils ont toujours le chic de reposé les problèmes sous un nouvel angle de par leurs questionnements, leurs passions et leurs regards neufs.

J'ai eu l'opportunité de partir un mois au Japon, ce fut un évènement très fort de la thèse, je remercie les Japonais de leurs accueils chaleureux, et en particulier le laboratoire du professeur Tanaka à l'université de Shinshu de Nagano. Je remercie notamment Hernan Aguirre, Roland Armas et Hugo Monzón avec qui j'ai pu partagé très bon moment temps dans le travail que dans le quotient. Ce voyage de recherche est tout simplement inoubliable ! Je ne pourrai oublier mes quelques nuitées réalisées dans un hôtel capsule.

Je tiens à remercier ma famille pour leurs ouvertures d'esprit qui m'ont permis de grandir en accord avec moi-même. Je remercie mes frères pour

leurs gentillesses, et tous les moments que j'ai passés avec eux. Je remercie mes amis de longue date avec qui j'ai eu, toujours avec humour, des débats passionnés sur tellement de sujets. Je remercie les doctorants de tout horizon avec qui j'ai partagé beaucoup, durant les formations doctorales, les conférences, etc....

TABLE DES MATIÈRES

1

INTRODUCTION

Dans cette thèse, nous étudions le problème du réglage automatique des paramètres dans un environnement de calcul distribué. Le choix du meilleur paramétrage est un problème connu depuis le début des algorithmes évolutionnaires dans les années 1980.

Cette problématique étudiée dans un environnement de calcul séquentiel s'apparente à un problème du bandit manchot. Imaginer un casino où M machines à sous ont une probabilité de gain définie et indépendante. Tout joueur désire maximiser ces gains en fonction de contraintes notamment pour un budget B limité. Afin de maximiser les gains, une stratégie est définie pour tenter de sélectionner la machine à sous la plus performante. Par exemple, nous pouvons imaginer les trois stratégies de sélection simples suivantes :

1. choisir pour chaque jeton du budget B une machine à sous aléatoirement ;
2. choisir pour l'ensemble du budget B une machine à sous aléatoirement ;
3. tester N fois chaque machine à sous, puis choisir la machine à sous ayant rapporté le plus de gains.

La stratégie 1 consacre $1/M$ du budget à la machine à sous la plus performante car l'utilisation de chaque machine à sous est équiprobable. Dans la stratégie 2, il y a une chance sur M de consacrer l'ensemble du budget à la machine la plus performante. Et la stratégie 3 consacre $N \times M$ jetons à essayer de détecter la machine à sous la plus performante ; le reste du budget est consacré à la machine à sous apparemment la plus performante.

La stratégie 1 assure de consacrer à la machine à sous la plus performante $1/M$ du budget, contrairement aux stratégies 2 et 3 où le budget à consacrer à la plus performante peut être nulle. La stratégie 3 a l'avantage d'être en deux phases : (i) l'exploration, et (ii) l'exploitation. La phase (i) visite toutes les machines à sous et estime le gain possible. La phase (ii) utilise la machine à sous ayant la meilleure estimation de gain. La difficulté est d'établir un estimateur pertinent, donc de savoir détecter la machine à sous la plus performante. D'autres stratégies ont pu être proposées dans le cadre formel du problème de bandit manchot [SB98 ; ACF02].

La problématique de recherche du paramétrage optimal pour une optimisation efficace revient à une instance du problème de bandit manchot. Le budget du joueur s'apparente aux ressources de calcul disponible pour

résoudre le problème et les machines à sous aux paramètres possibles de l'algorithme d'optimisation. Le budget est limité alors que le nombre de paramètres possibles peut être infini. Nous étudions, par la suite, la manière de choisir un paramétrage durant l'optimisation de façon autonome et cela dans un contexte de résolution de problèmes d'optimisation et de calcul parallèle/distribué.

1.1 CONTEXTE GÉNÉRAL

En recherche opérationnelle et en intelligence artificielle, l'optimisation consiste à chercher une solution optimale à un problème selon un critère donné parmi un ensemble de solutions candidates, c'est-à-dire l'espace de recherche. Un problème peut être une modélisation d'une situation comme charger un porte-conteneurs, déterminer le meilleur temps de synchronisation entre les feux tricolores, placer les pièces électroniques sur une puce à imprimer en minimisant la longueur des pistes dans le domaine industriel, etc... Afin de résoudre un problème d'optimisation de façon exacte ou inexcate, il est conseillé d'utiliser un algorithme de résolution telle que la programmation par contraintes, les algorithmes de graphe, la programmation linéaire ou les métaheuristiques (les algorithmes évolutionnaires, recuit simulé, recherche tabou, Hill-Climber...) [Sia14]. Les métaheuristiques et les algorithmes évolutionnaires ne garantissent pas d'obtenir une solution exacte, seule une solution approchée est obtenue. La force de ces algorithmes est de pouvoir trouver une approximation d'une solution optimale pour des problèmes réputés difficiles en un temps acceptable. Ces algorithmes peuvent être utilisés aussi bien pour les problèmes à variable continue, que les problèmes d'optimisation à variable discrète. Dans les travaux de cette thèse, nous nous intéressons en particulier aux problèmes à variables discrètes à travers l'optimisation combinatoire. L'une des difficultés de ces méthodes d'optimisation est leur paramétrage. Afin d'avoir un algorithme performant, il est nécessaire de faire des choix de paramètres en fonction du problème traité. Dans le cas où les caractéristiques du problème sont connues, problèmes dits boîte blanche, ce choix est facilité ; par contre dans le cas où les caractéristiques du problème ne sont pas connues dits boîte noire, ce choix repose sur les connaissances de l'expert du problème.

Le théorème « No Free Lunch » – NFL – énoncé par David Wolpert et William Macready en 1997 [WM97] démontre qu'aucun algorithme ne surpasse un autre en moyenne sur l'ensemble de tous les problèmes d'optimisation combinatoire. Donc, il ne peut pas y avoir de méthode de résolution universelle, ce qui explique partiellement l'abondance d'algorithmes disponibles. Le paramétrage de l'algorithme de résolution est tout aussi important que le choix de l'algorithme. Il est possible d'imaginer que le choix automatique d'un algorithme efficace est la combinaison intelligente de composants

de bas niveau et le réglage correct de leurs paramètres. C'est une des questions les plus difficiles à laquelle la communauté d'optimisation par métaheuristique et algorithme évolutionnaire doit faire face. Par conséquent, cela conduit au défi difficile de concevoir un mécanisme de sélection efficace qui est capable de décider quels paramètres choisir pour un problème donné.

Depuis le début des algorithmes évolutionnaires, le réglage automatique des paramètres se pose [Gre86]. Choisir le paramétrage adéquat d'un algorithme pour résoudre un problème où dans ce contexte un paramètre est considéré au sens large du terme. Cela peut être le paramètre d'un algorithme, une métaheuristique ou un opérateur de mutation (voir section 2.4.3). Le problème du choix du paramètre, ou le problème de sélection d'algorithmes – Algorithme Selection Problem – [Ric76] est intéressant tant pour son importance pratique que pour les nouvelles possibilités de recherche qu'il ouvre et pour la conception de nouvelles techniques. Cela pose quatre questions de fond selon Kotthoff [Kot12] : sur quelle base choisir les algorithmes ? lequel choisir, quand et comment le choisir ? comment faciliter la sélection ? Trouver le paramètre le plus approprié permet de réduire le temps d'optimisation, notamment dans les cas où le calcul de la qualité de la solution – c'est-à-dire la fitness – est coûteux en temps de calcul par exemple une simulation de phénomène physique.

La sélection du paramétrage peut s'effectuer par deux approches principales [Eib+07] : (i) le réglage hors ligne et (ii) le réglage en ligne. Le réglage hors ligne détermine les paramètres ou l'algorithme avant l'optimisation et il sera utilisé tout au long du processus d'optimisation. Le réglage en ligne règle dynamiquement les paramètres durant le processus d'optimisation. Plus particulièrement, dans le réglage en ligne adaptatif, les paramètres sont choisis en fonction des performances passées. Le choix peut s'effectuer à partir d'une loi de distribution ou à partir d'un ensemble de paramètre nommé portefeuille ; dans ce document, nous considérons le portefeuille de paramètres. Les stratégies de sélection de paramètres se décomposent en trois composants : le calcul de la récompense, l'estimation de la qualité des paramètres et le choix effectif de l'opérateur en fonction de l'estimation des paramètres. La récompense permet d'estimer la pertinence d'une métaheuristique par rapport à une autre ; il existe naturellement plusieurs façons de calculer une récompense.

Le meilleur opérateur est basé sur les récompenses observées de l'ensemble des métaheuristiques. Il utilise une technique d'apprentissage statistique pour déterminer un classement pertinent ou le meilleur opérateur. Dans ce type de technique le dilemme exploration/exploitation est un des éléments essentiels, l'exploration – ou diversification – consiste en une phase exploratoire de l'espace des possibles, typiquement essayer une métaheuristique peu utilisée ; l'exploitation – ou intensification – l'utilisation d'un paramétrage ou les paramètres considérés comme les plus appropriés à la

situation. Dans ces travaux, nous nous concentrons sur l'étude des stratégies de sélection adaptative – c'est-à-dire en ligne – car ces techniques sont adaptées à la recherche des paramètres des algorithmes distribués. De plus, les paramètres pertinents appris par une stratégie de sélection adaptative peuvent être utilisés ultérieurement et peuvent servir de méthode de réglage hors ligne pour une prochaine optimisation.

Les environnements parallèles sont favorisés par l'augmentation des ressources de calcul (GPGPU, grille de calcul, etc...) la diminution des temps de communication et des coûts matériels faibles. L'optimisation de problème est demandeuse de ressource de calcul notamment les problèmes à combinatoire élevée, ou les problèmes dont le calcul de la qualité de la solution est important. Les algorithmes évolutionnaires peuvent se paralléliser naturellement : chaque nœud de calcul correspondant à une île peut accueillir un individu ou une sous-population. Chaque île peut communiquer avec une autre. Au lieu de concevoir des algorithmes parallèles spécifiques à un problème, il est intéressant de proposer un algorithme parallèle générique utilisant les algorithmes connus dans le cadre séquentiel – il en existe beaucoup – tels que les stratégies de sélections adaptatives.

La problématique étudiée dans ce travail de thèse est la conception et la compréhension du mécanisme de sélection adaptative des paramètres dans un environnement de calcul distribué. Il y a fondamentalement deux éléments importants dans ces travaux : le premier est la stratégie de sélection adaptative qui a été étudiée notamment par [Fia+09] dans un environnement de calcul séquentiel, et l'apport du calcul parallèle dans le mécanisme de stratégie de sélection. Ce dernier est la suite du travail initié par [DV11]. Les auteurs ont proposé un cadre général pour inscrire une stratégie de sélection dans un environnement distribué. La difficulté principale dans ce cadre est le type d'information à communiquer et comment l'intégrer dans une stratégie de sélection. Concevoir des stratégies de sélection qui intègrent de l'information sans que cela pénalise le calcul du meilleur opérateur sera abordé durant ces travaux. La seconde difficulté est d'avoir un bon compromis exploitation/exploration. Deux approches peuvent être étudiées : dans le déroulement séquentiel du choix du paramètre (i) et sur la répartition des paramètres sur l'ensemble des nœuds de calcul (ii).
(i) De façon séquentielle, à chaque itération un paramétrage est considéré sur l'ensemble des itérations. Trop explorer augmente le temps de résolution, et trop exploiter peut être tout aussi problématique, car avant d'exploiter, il faut choisir le paramétrage qui convient le mieux. (ii) La répartition des paramétrages disponibles sur l'ensemble des nœuds de calcul offre la possibilité d'explorer et d'exploiter au même moment. Cependant, ce choix est guidé du point de vue du nœud de calcul – local – ou du point de vue de l'ensemble des nœuds de calcul – global –. Un choix de paramétrage local n'est pas obligatoirement le meilleur choix pour l'ensemble des nœuds de calcul.

Nous avons voulu aussi mieux comprendre comment l'information à destination de la stratégie de sélection influence son comportement, notamment l'apprentissage de bonnes métaheuristiques ou le réapprentissage (c'est-à-dire s'adapter au changement de performance des paramètres au cours de l'optimisation).

Nous étudions les stratégies de sélection dans deux environnements parallèles différents typiquement rencontrés dans les algorithmes évolutionnaires : modèle en île et le modèle maître-esclave.
(i) Dans le modèle en île, chaque nœud de calcul est une entité autonome appelée île capable de résoudre le problème à l'aide d'un algorithme. L'objectif est d'avoir une stratégie de sélection capable de communiquer entre les îles afin que chaque nœud choisisse le bon algorithme de résolution. La stratégie de sélection doit collecter les informations voisines afin de déterminer les paramètres à appliquer ; la stratégie de sélection a une vision locale au nœud des informations sur les paramétrages à utiliser. Elle ne connaît pas les informations connues d'autres nœuds non voisins. (ii) Dans le modèle maître-esclave, chaque nœud esclave exécute un algorithme en fonction des instructions envoyées par le nœud maître. La stratégie de sélection s'exécute sur le nœud maître collectant l'ensemble des informations produites par les nœuds esclaves, elle a une vision globale. La conception d'une stratégie de sélection trouvant les paramètres adéquats à la situation permet d'optimiser un problème sans a priori sur l'optimisation. Seul le choix du portefeuille de paramètres est guidé par la classe du problème. Chacune des deux architectures a son intérêt, le modèle en île par sa nature distribuée et tolérante aux pannes. Par contre son implémentation s'avère plus délicate surtout au niveau du système de communication entre les îles. L'architecture maître-esclave bien que son implémentation soit plus facile, les communications entre le nœud maître et les nœuds esclaves sont limitées par le débit de communication. Il peut y avoir un goulot d'étranglement quand le nœud maître envoie en même temps à l'ensemble des nœuds esclaves un message, et que les nœuds esclaves sont en trop grand nombre. Dans les architectures maître-esclave nous avons naturellement tenté de réduire les communications. Ces architectures sont appropriées lorsqu'il y a beaucoup de calcul à réaliser, typiquement dans le cas où le problème fait appel à une fonction de fitness étant une simulation informatique.

Dans la suite, nous présentons les éléments de contribution et les problématiques sur l'étude de la stratégie de sélection, une étude des stratégies de sélection pour un environnement distribué de modèle en île, une étude du comportement des stratégies de sélection à travers le modèle « fitness cloud » et une étude des stratégies de sélection dans un modèle de calcul maître-esclave.

1.2 SYNTHÈSE DES TRAVAUX

1.2.1 *Modèle en île à récompense collective*

Les modèles en île sont inspirés des algorithmes évolutionnaires ; la population est répartie sur différentes îles, un nœud de calcul correspond à une île. Chaque île accueille une sous-population d'un ou plusieurs individus. Chacune communique des informations en fonction d'une topologie : cercle, complet, etc. Deux types de modèles en île peuvent être considérés : (i) les modèles en île hétérogène, et (ii) les modèles en île homogène.
(i) Dans le cas des modèles en île hétérogène, les paramètres (ou les métaheuristiques) peuvent être différents de l'île voisine pour réaliser l'optimisation. (ii) les modèles en île homogène considèrent pour chaque île le même paramétrage (ou les métaheuristiques). Dans ces travaux, nous nous focalisons sur les modèles en île hétérogène. Ces travaux font suite aux propositions de Derbel *et al.* [DV11] proposant un cadre pour l'optimisation distribuée nommée « Distributed Adaptive Metaheuristic Selection » – DAMS –. L'idée intuitive est d'avoir trois niveaux : l'échange de la meilleure solution entre les îles, l'échange de l'information des stratégies de sélection afin de prendre une décision la plus éclairée possible, l'exécution de la métaheuristique et le calcul de la fitness sur chaque île. Ce cadre distribué est générique dans le sens où il accepte différentes stratégies possibles.

À travers cette étude, notre objectif est de répondre aux questions :

— Comment élaborer une stratégie de sélection au sein d'un environnement distribué de modèle en île ?
— Quelle information partagée ?
— Quelle stratégie est la plus pertinente ? Quels sont les paramètres de ces stratégies de sélection ? Sont-ils robustes ?

Pour répondre à ces questions, nous nous sommes inspirés du cadre défini par DAMS. Il considère trois paramètres de conception importants : le portefeuille de métaheuristiques, la stratégie de sélection et le problème à résoudre. Le portefeuille de métaheuristiques dépend du problème d'optimisation. Dans nos travaux, nous nous intéressons au problème OneMax et au problème NK, deux problèmes bien connus dans la littérature. Le problème OneMax, a été utilisé pour étudier les stratégies de sélection dans un cadre séquentiel [Fia+09 ; Fia10 ; Fia+10a ; Fia+10b]. Nous avons adapté des stratégies de sélection existantes dans le cadre séquentiel au cadre distribué. Ces techniques sont issues de l'apprentissage automatique.

L'algorithme DAMS fonctionne de la manière suivante. Chaque île fonctionne de façon synchrone, c'est-à-dire que chaque île envoie en même temps l'information à ses voisins et chaque île attend en même temps les informations de ses voisins. Un portefeuille de paramètres – un portefeuille de métaheuristiques – défini en fonction du problème à résoudre est identique

pour chaque île. Chaque île exécute la même stratégie de sélection. Lors de la phase d'initialisation, la solution et les paramètres de la stratégie de sélection sont définis. Ensuite deux éléments sont communiqués : la solution et les récompenses associées aux paramètres sont envoyées au voisin. La stratégie utilise les récompenses pour déterminer un paramétrage – métaheuristique – pertinent à utiliser pour optimiser le problème. L'exécution de l'algorithme associé au paramétrage modifie la solution, cela permet de calculer la récompense qui est la différence de fitness (qualité de la solution) entre deux itérations.

Les éléments importants de l'algorithme sont :

— l'initialisation des récompenses pour la stratégie de sélection. Comment la stratégie de sélection choisit une métaheuristique alors qu'elle n'a pas de récompense associée par le passé ?
— comment s'effectue l'échange des récompenses entre les voisins ?

À travers l'algorithme DAMS, nous avons étudié et comparé différentes stratégies de sélection en environnement distribué et en environnement séquentiel pour le problème OneMax et le problème NK. Les stratégies de sélection conçues pour un environnement séquentiel sont nommées indépendantes et les stratégies de sélection conçues pour un environnement distribué sont nommé collectives. Nous avons exploré des variantes calculant différemment la stratégie de sélection. Les stratégies de sélection adaptative ont deux stratégies de sélection de base : la stratégie aléatoire et la stratégie constante.

Quatre techniques différentes sont étudiées dans le chapitre 3. La technique Select-Best-and-Mutate – SBM – sélectionne le paramétrage ayant la meilleure récompense suivant une probabilité p, sinon un paramétrage est choisi aléatoirement dans le portefeuille. La technique Upper Confidence Bounds – UCB – estime la borne supérieure de la moyenne empirique pour chaque opérateur et sélectionne celui ayant la meilleure estimation. Adaptative Pursuit – AP – sélectionne proportionnellement un opérateur à exécuter en fonction de probabilité ; il met à jour les opérateurs correspondants en fonction de l'estimation de la récompense. La technique ϵ-Greedy qui moyenne l'ensemble des récompenses pour chaque paramétrage choisit le paramétrage ayant la meilleure récompense. Deux variantes pour chaque stratégie de sélection sont considérées : individuelle et collective. Une stratégie de sélection individuelle utilise seulement les récompenses calculées sur le nœud local, une stratégie de sélection collective utilise les récompenses du nœud local et des nœuds voisins. Pour plus de détails voir la section 2.4.

Ces travaux présentés au chapitre 3 ont fait l'objet de publication dans [Jan+15b] et [Jan+15a]. Nous présentons plusieurs stratégies de sélection dans un cadre distribué et étudions leur performance.

1.2.2 *Benchmark : modèle fitness cloud*

Mieux saisir le comportement de stratégie de sélection adaptative nous permet de saisir les éléments de conception importants, et ainsi aider à la conception de stratégies de sélection. Un des éléments importants dans l'établissement d'une stratégie de sélection est le calcul de la récompense. Nous proposons un modèle qui capture le problème et la qualité du paramétrage en fonction de la qualité de la solution. Les objectifs de ce modèle sont de comprendre le comportement de l'apprentissage des stratégies de sélection par rapport à l'amélioration attendue de chaque métaheuristique, de déterminer dans quel contexte une stratégie de sélection est pertinente ou au contraire non pertinente. Sur la base de ces observations, notre but est de proposer des améliorations d'une stratégie de sélection et concevoir de nouvelles stratégies plus pertinentes.

Des modèles pour étudier les stratégies de sélection ont été proposés dans la littérature [DaC+08] [Thi07]. Nous proposons une modélisation du problème et de son opérateur dont la performance d'un paramétrage est donnée en fonction de la fitness. Définir la qualité du paramétrage en fonction de la fitness permet d'être au plus proche des problèmes, de pouvoir modéliser des caractéristiques importantes comme la dégradation de la qualité de paramétrage en fonction de l'accroissement de la fitness. Une dégradation est par exemple observée dans le OneMax et le NK au fur et à mesure de l'évolution. Un autre élément de modélisation important est le changement de performance entre deux paramétrages. Le fitness cloud – nuage adaptatif – développé par [VCC03] établit l'espérance de l'amélioration d'un opérateur en fonction de la fitness précédente en le mesurant sur un problème. L'espérance de l'amélioration en fonction de la fitness est donc une abstraction d'un problème. Nous pouvons établir un modèle de problème à l'aide d'une loi de l'espérance de l'amélioration que nous avons nommé le modèle « fitness cloud ». Le FCM nous permet d'exhiber des comportements de la stratégie de sélection en fonction de situation choisie modélisée et de les évaluer avec des problèmes tels que le problème OneMax ou le problème NK.

À travers notre étude, à l'aide de différents scénarios, nous répondons aux questions suivantes :

— Comment les caractéristiques du problème influencent-elles les stratégies de sélection ?
— Quelle est la composition optimale d'un bon portefeuille d'algorithmes ?
— Dans quelle situation, une stratégie de sélection est pertinente par rapport aux stratégies naïves, c'est-à-dire de sélection aléatoire ou constante ?

Le FCM permet de capturer différentes situations au travers de scénarios. Nous en proposons plusieurs qui capturent des propriétés inspirées des deux problèmes académiques étudiés OneMax et NK dans un contexte séquentiel. Ces propriétés sont la dégradation de l'amélioration attendue, le rapport

entre les améliorations attendues des opérateurs en fonction de l'état de la recherche et le changement de l'ordre de la performance des opérateurs (le meilleur opérateur change au cours du temps).

Nous avons étudié au chapitre 4 et publié dans [Jan+16] quelques situations à travers une modélisation du problème et de son opérateur dans un environnement séquentiel. Nous avons pu établir en fonction des paramètres du scénario dans quel cas une stratégie de sélection constante est plus intéressante par rapport une stratégie de sélection basée sur une méthode d'apprentissage du meilleur opérateur.

1.2.3 *Modèle de calcul maître-esclave*

L'étude des stratégies de sélection avec le fitness cloud modèle est réalisée dans un environnement séquentiel. Nous étendons ce modèle au cadre parallèle. L'ajout du calcul parallèle ajoute une nouvelle dimension que la stratégie de sélection doit traiter. À la différence des modèles en île, nous avons choisi le modèle maître-esclave car ce dernier est plus aisé d'implémentation notamment dans le traitement des communications. De plus, cela nous permet d'étudier un environnement de calcul très utilisé.

Dans une architecture maître-esclave, le nœud maître prend l'ensemble des décisions et les nœuds esclaves réalisent les tâches en parallèle demandées par le maître. Il a l'avantage par rapport au modèle en île d'avoir une implémentation plus aisée notamment pour la communication entre les nœuds. Par contre, une des limites est la charge réseau entre le nœud maître et les nœuds esclaves. Dans une stratégie de sélection séquentielle, à chaque itération une récompense est donnée à la stratégie de sélection ; dans une stratégie de sélection maître-esclave, ce n'est pas une seule récompense qui est calculée par itération mais une pour chaque nœud. Par conséquent, le nœud maître a une vision globale sur le système contrairement au modèle en île. Pour résoudre un problème avec une stratégie de sélection adaptative avec le modèle maître esclave, le maître accomplit deux éléments importants : (i) l'agrégation des récompenses calculées sur les nœuds esclaves, et (ii) l'exécution de la stratégie de sélection pour déterminer les métaheuristiques à exécuter sur les nœuds esclaves. Concernant les communications, deux types d'envois de messages peuvent être considérés : les communications synchrones ou asynchrones. Dans la communication synchrone à chaque itération, tous les nœuds esclaves reçoivent leurs tâches à réaliser et le maître attend la réception de l'ensemble émis par des messages des nœuds esclaves après exécution de leur tâche. Le maître pourra procéder à un nouvel envoi de tâches aux nœuds esclaves que lorsqu'ils sont tous réceptifs. La limite dans ce type de communication est que si le temps calcul sur le nœud varie, du temps de calcul sera perdu par les nœuds qui attendent les nœuds encore au travail. La communication asynchrone est une solution à cette limite

puisque le maître envoie le travail à un nœud quand celui-ci a fini. Cependant, chaque nœud retournant de l'information au maître à n'importe quel moment, la question se pose sur l'obsolescence de l'information lors de la prise de décision de la stratégie de sélection. Dans ces travaux, nous nous intéressons à la communication synchrone.

Dans ce travail, nous nous inscrivons dans une architecture synchrone, nous présentons une technique de traitement par lots sur les nœuds esclaves pour réduire le nombre de communications entre le maître et les esclaves. À travers cette étude, nous cherchons à répondre aux questions suivantes :

— Comment construire une stratégie de sélection performante au sein d'un environnement parallèle maître-esclave en fonction des caractéristiques du problème et du nombre de nœuds de calcul ?
— La façon de considérer la récompense influence l'estimation du paramétrage. Comment calculer une récompense pour chaque paramétrage du portefeuille de façon pertinente ?
— Comment réduire le temps de communication ?
— Avec le portefeuille de paramètres, il est possible d'explorer le paramètre le plus approprié et de l'allouer proportionnellement au temps. Quelle est la répartition optimale des opérateurs sur les nœuds de calcul ?

Dans cette architecture, la stratégie de sélection a une vision de l'ensemble des nœuds, elle calcule le meilleur opérateur pour chaque nœud sur la base des récompenses. Deux types de stratégie de sélection sont possibles : homogène et hétérogène. Les stratégies de sélection homogène déterminent un opérateur, le meilleur si possible, qui sera affecté à l'ensemble des nœuds esclave. La stratégie de sélection hétérogène détermine un opérateur pour chaque nœud esclave, cela signifie que deux nœuds différents peuvent avoir un opérateur différent.

Nous avons étudié au chapitre 5 et publié dans [Jan+17a], [Jan+17b] des éléments de conception du modèle de distribution maître-esclave tel que la comparaison homogène et hétérogène et le traitement par lot des paramétrages. L'architecture maître-esclave a l'avantage de centraliser l'ensemble des informations connues de résolution au niveau du nœud maître. Ceci dans l'objectif d'affecter au mieux les ressources de calcul des esclaves. À l'aide du modèle établi précédent, nous avons étudié les modes de calcul de la récompense et de la répartition des paramétrages entre des nœuds esclaves.

2

PARAMÉTRAGE AUTOMATIQUE POUR L'OPTIMISATION

Dans ce chapitre, nous faisons un tour d'horizon des concepts à la base de ces travaux. Le choix des paramètres est une des difficultés que l'informaticien doit prendre en compte dans la conception d'algorithme d'optimisation dans l'objectif de résoudre des problèmes. D'ailleurs, dès 1986, Grefenstette[Gre86] propose de trouver les paramètres optimaux des algorithmes génétiques par des algorithmes génétiques.

Dans la section 2.1, nous définissons un problème d'optimisation et les outils nécessaires pour étudier les caractéristiques d'un problème d'optimisation. Nous posons notamment la définition des problèmes boîte noire. La conception dans ces travaux des stratégies de sélection sera orientée a priori pour les problèmes boîte noire. Ces derniers sont largement répandus pour les problèmes industriels. La section 2.2 présente deux problèmes d'optimisation utilisés comme benchmark bien connus de la littérature scientifique : le problème OneMax et le paysage NK. Le premier est un problème monomodal c'est-à-dire avec un seul optimum local qui est global ; contrairement au second qui lui est multi-modal, c'est-à-dire ayant plusieurs optimaux locaux. La section 2.3 aborde les différents algorithmes d'optimisation stochastiques existants. La section 2.4 traite des deux manières – avant et durant la résolution d'un problème – de réaliser le paramétrage automatique des algorithmes d'optimisation. La section 2.5 approfondit l'étude du paramétrage automatique durant la résolution d'un problème de façon séquentielle. La section 2.6 discute du paramétrage automatique dans un cadre distribué.

2.1 PROBLÈMES D'OPTIMISATION « BOÎTE NOIRE »

Généralement, un problème d'optimisation est un problème auquel nous nous attachons à trouver une solution, si possible la meilleure solution selon un ou plusieurs critères considérés.

Trois catégories de problèmes d'optimisation peuvent être étudiées : les problèmes dits boîtes blanches, boîtes grises et boîtes noires. Concernant les problèmes boîtes blanches, la connaissance de la définition des problèmes est complète. Pour ce type de problème, la définition algébrique est totalement connue par l'utilisateur de la méthode d'optimisation, par exemple le problème du voyageur du commerce. À l'opposé, pour les problèmes boîtes noires aucune structure n'est connue a priori. Pour les boîtes grises, seule une connaissance partielle est disponible. L'ensemble des travaux de cette thèse est placé sous l'angle des problèmes boîtes noires.

2.1.1 *Définition*

En optimisation, un problème mono-objectif est un couple (X, f) avec X un ensemble de solutions possibles au problème, et f est une fonction qui associe à chaque élément de x. L'ensemble des solutions X acceptables pour la fonction est appelé espace de recherche, et la fonction f est appelée la fonction fitness, fonction d'évaluation ou encore fonction objectif.

La recherche d'une ou plusieurs solutions optimales x^*, s'effectue par minimisation ou par maximisation. Plus formellement, dans le cas de la maximisation :

$$x^* = \underset{x \in X}{\operatorname{argmax}}\ f(x) \tag{1}$$

Le coût en temps de calcul de la fonction de fitness varie suivant la nature de la fonction.

Par exemple, le calcul d'opérations algébriques possède un coût négligeable de l'ordre de quelques microsecondes. À l'opposé, une simulation informatique de mobilité urbaine contenant vingt-mille agents [Arm+16] dure plusieurs minutes pour réaliser une évaluation, ou encore la simulation d'une partie du réacteur nucléaire peut durer quelques dizaines de minutes en moyenne [Mun+16]. La recherche de la meilleure solution sera donc impactée par le coût de calcul de la fitness. Il est pertinent pour les problèmes ayant une évaluation coûteuse en calcul de profiter des environnements de calcul parallèle. Ces environnements permettent d'utiliser dans un même temps plus de ressources et de puissance informatique.

Les problèmes boîtes noires n'ont aucune information connue a priori sur les propriétés de la fonction à optimiser [1]. La résolution du problème ne peut être guidée par les propriétés de la fonction à optimiser. Choisir l'algorithme relève donc plus de l'intuition de l'expert, de l'analyse expérimentale, que de résultats analytiques.

Le choix du paramètre peu reposer sur l'analyse du problème boîte noire. Le problème flow-shop [Mar+11], par exemple, est un problème ayant beaucoup de solutions de même qualité c'est-à-dire beaucoup de neutralité. Il aura besoin d'un algorithme explorant l'espace de recherche au lieu d'exploiter une petite portion. La stratégie de sélection de paramètres pourra déterminer durant l'optimisation le paramétrage permettant un bon équilibre entre l'exploitation et l'exploration. La stratégie de sélection analyse de façon empirique le comportement des paramètres et détermine le paramètre le plus approprié.

1. Hormis quelques propriétés simples par exemple la dimension.

2.1.2 *Paysage de fitness et fitness cloud*

PAYSAGE DE FITNESS La notion de paysage de fitness – ou paysage adaptatif – est issue de la biologique évolutive, introduite par Wright [Wri32]. Il décrit la fitness en fonction du génotype ou du phénotype permettant de caractériser les déplacements d'une population d'individus. Les paysages de fitness ont aussi été introduits en optimisation dans le domaine des algorithmes évolutionnaires et des recherches locales pour modéliser la dynamique de ces algorithmes. En plus de donner une image intuitive sous forme de pics (optimal locaux et optimal globaux), vallées ou plateaux (neutralité), il permet également d'établir des mesures caractérisant la difficulté d'optimisation comme la neutralité, la rugosité, la multimodalité, la monomodalité, etc... Par exemple, la neutralité mesure la qualité des solutions voisines ayant la même fitness ou proche. La rugosité mesure la quantité de discontinuités ; son opposé est la quantité de plateaux. Il décrit donc l'organisation des solutions. Pour une représentation visuelle voir la figure 1 extraite de [Ver16].

Plus formellement, un paysage de fitness est un triplet (X, N, f) où X est l'ensemble des solutions potentielles au problème d'optimisation appelé espace de recherche, $N : X \to 2^X$ est une relation de voisinage entre les solutions qui associe à chaque solution un ensemble de solutions dites voisines et $f : X \to \mathbb{R}$ est la fonction d'évaluation à optimiser, aussi appelée fonction de fitness. Une solution voisine est accessible avec un opérateur de mutation.

Un optimum global est la meilleure solution possible trouvable de l'espace de recherche selon la fonction d'évaluation. Il peut y avoir plusieurs optimaux globaux. Un optimal local est la meilleure solution localement, relativement au voisinage, représenté par un sommet. Il peut en exister plusieurs. Dans les algorithmes d'optimisation, l'exploitation tend à converger vers un optimal local, l'algorithme est piégé ; et l'exploration permet éventuellement d'échapper à un optimal local afin de trouver une meilleure autre solution dans l'espace de recherche. L'analyse du paysage de fitness, qui est propre au problème et à l'opérateur, apporte des informations importantes sur le choix le plus pertinent d'une classe de métaheuristiques par rapport à une autre et permet d'en concevoir une plus adaptée. Il permet de capturer la structure du paysage. Par exemple, si un problème a beaucoup de neutralité, une métaheuristique explorant beaucoup sera plus pertinente que pour un problème ayant des sommets marqués ou exploiter sera plus intéressant. Il donne aussi une idée sur le degré de difficulté de la résolution du problème.

FITNESS CLOUD Le fitness cloud model a été introduit par Verel, Collard et Clergue [VCC03 ; CVC04] et indépendamment par Barnett [Bar03]. Ce modèle représente la fitness après application d'un opérateur local en fonction de la fitness précédente. Le fitness cloud de l'opérateur de recherche local (op) est défini [Ver+07] comme la densité de probabilité bivariée condition-

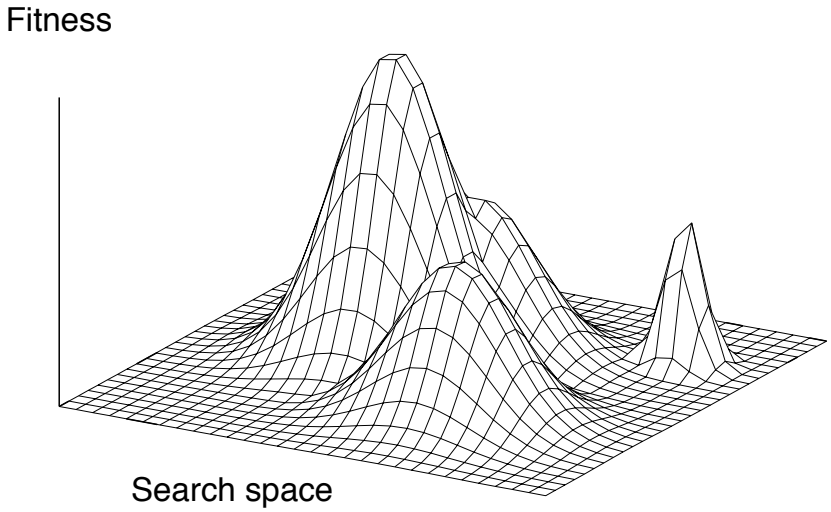


FIGURE 1 – Une représentation visuelle d'un paysage de fitness multimodal issue de [Ver16].

nelle $P_{op}(Y = \widetilde{\varphi}|X = \varphi)$ pour atteindre la solution de fitness $\widetilde{\varphi}$ depuis la solution de fitness φ en appliquant à l'opérateur op. La représentation graphique du Fitness cloud Model est donnée par l'ensemble $FC = \{(\varphi, \widetilde{\varphi})|P_{op}(\varphi, \widetilde{\varphi}) \neq 0\}$. Lorsque le nuage exhibe une forme globalement linéaire, la pente de la régression linéaire – ou les pentes d'une régression linéaire par morceaux – est un indicateur sur la difficulté de résolution du problème avec l'opérateur considéré [Van+06].

2.2 BENCHMARK : PROBLÈME D'OPTIMISATION

Nous nous sommes attachés à étudier les stratégies de sélection à travers un banc d'essai bien connu de la littérature : le problème OneMax et le paysage NK. Le OneMax a été utilisé pour l'étude des stratégies de sélection dans le cadre séquentiel par [Fia+09] et dans un cadre distribué via des modèles en îles par [DV11]. Le problème OneMax a l'avantage d'être associé à des opérateurs bien décrits analytiquement (voir les travaux de [Chi+15; DD14; GL11]). Le NK est un problème NP-complet avec un paramètre permettant de régler la difficulté du problème. Tout au long de ce document, nous utilisons une métaheuristique $(1 + \lambda)$-ES ou un parent produit un en-

semble λ de descendants pour résoudre les problèmes d'optimisation. Les opérateurs de mutation seront choisis en fonction du problème.

2.2.1 *Problème OneMax*

L'espace de recherche du problème unimodal OneMax est constitué des chaînes binaires $\{0,1\}^N$ de longueur N. L'optimal est obtenu quand la chaîne binaire ne contient que des uns. La fonction de fitness se définit ainsi :

$$f(x) = \sum_{i=1}^{N} x_i \tag{2}$$

La fitness est égale au nombre de 1 dans la chaîne binaire : $|x|_1$. Généralement, lors de l'utilisation de ce problème pour tester un algorithme d'optimisation, l'ensemble des bits est initialisé à zéro, ou aléatoirement. Deux types d'opérateurs de mutation sont définis dans le cadre de ce problème. L'opérateur bit-flip mute chaque case de la chaîne binaire avec une probabilité de c/N (c constant), avec N la longueur de la chaîne binaire. L'opérateur kbit mute exactement k cases strictement différentes (k constant).

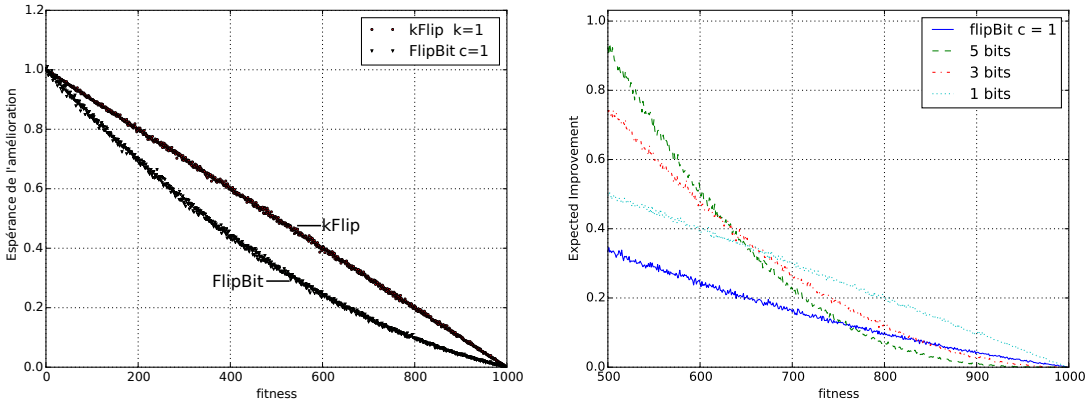
Pour ce problème, nous utilisons un portefeuille de quatre opérateurs de mutation : trois opérateurs kbit $k = \{1, 3, 5\}$ et l'opérateur bit-flip avec $c = 1$. Ce choix est dans la continuité des travaux précédents réalisés dans un cadre séquentiel par [Fia+09] et par [GLS16 ; Goë14 ; GL11] pour les modèles en île avec migration de la population.

Afin de fixer les idées sur le comportement des opérateurs, la figure 2 montre l'espérance de l'amélioration pour différent opérateur en fonction de la fitness. Ces résultats bien connus sont issus d'expériences, ils peuvent être aussi obtenus de façon analytique dans [GL11]. En (a), une comparaison expérimentale des deux opérateurs kflip avec $k = 1$ et bit-flip avec $c = 1$, l'espérance de l'amélioration n'est pas la même sauf quand la fitness est égale à zéro et à N. Cette différence tient du fait que le bit-flip ne mute pas obligatoirement un bit à chaque fois qu'il est appliqué contrairement au kflip. Cette différence bien que faible, en faveur du kflip, influence le temps de résolution. En (b), l'amélioration de chaque opérateur, ces opérateurs composent le portefeuille dans la majorité des expériences, nous pouvons voir que dans l'ordre que l'opérateur 5 bits, 3 bits, et 1 bit sont meilleurs successivement.

2.2.2 *Problème des paysages NK*

Le problème NK, inventé par Kauffman [Kau93], issu de la biologie est une classe de problèmes plus sophistiquée. La famille des paysages NK constitue un modèle de problèmes multimodaux et NP-complets [2].

2. Lorsque les liens épistatiques sont aléatoires



(a) (b)

FIGURE 2 – L'espérance de l'amélioration des opérateurs de mutation sur le problème OneMax $N = 1000$. En (a) comparaison du kflip $k = 1$ et bit-flip $c = 1$ avec $\lambda = 1$, (b) comparaison des performances du portefeuille qui est utilisé.

L'espace de recherche correspond à une chaîne binaire de longueur N : $\{0,1\}^N$. N fait référence à la dimension du problème et K au nombre de bits qui influencent une position particulière de la chaîne de bits, c'est-à-dire les interactions épistatiques. La table de récompense des liens épistatiques est générée aléatoirement uniformément avec des nombres réels compris entre 0 et 1.

La fonction objectif $f : \{0,1\}^N \to [0,1)$ à maximiser est définie comme suit :

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots, x_{i_K}) \tag{3}$$

où $f_i : \{0,1\}^{K+1} \to [0,1)$ définit la fonction de composants associée à chaque bit x_i.

En augmentant le nombre d'interactions épistatiques K de 0 à $(N-1)$, les paysages NK peuvent être progressivement ajustés de lisse à rugueux. Quand $K = 0$, le problème est unimodal et semblable au problème OneMax, par contre quand $K \geqslant 1$ le problème est multimodal et non-linéaire. Dans ce travail, nous définissons la position de ces interactions au hasard. Les valeurs des composants sont uniformément réparties dans la plage $[0,1)$.

Les opérateurs de mutation sont des bit-flip avec le paramètre c qui permet de régler le taux de mutation de chaque bit de la chaîne binaire.

2.2.3 *Modèle de problème d'optimisation*

Des modélisations de problème d'optimisation ont été proposées pour des environnements non-stationnaires, c'est-à-dire où la performance de l'opérateur change au cours de l'optimisation.

Thierens *et al.* [Thi05 ; Thi07] proposent une modélisation dont la performance d'un opérateur change en fonction de période. Une période correspond à un nombre d'appels de la fonction à optimiser par l'algorithme. Le scénario appelé « uniforme » est défini comme suit : dix périodes et 5 opérateurs sont définis. La durée d'une période, fixée initialement, est la même pour toutes les périodes. Cinq récompenses sont issues d'une distribution de probabilité. Elles sont définies à l'aide d'une distribution uniforme sachant que pour chaque récompense successive la valeur (bornes de la distribution) diminue. Pour chaque période un opérateur est associé à une récompense, et à chaque période la récompense change. L'idée de ce modèle est d'étudier le comportement des stratégies de sélection quand il y a un changement de performance pour un opérateur. Par contre, ce changement est dépendant de la façon dont l'algorithme d'optimisation fait appel à la fonction d'évaluation.

D'autres scénarios sont proposés par Da Costa *et al.* [DaC+08] basés sur la modélisation de Thierens. Deux scénarios sont proposés : (i) un scénario booléen, et (ii) un scénario aberrant. Dans le scénario booléen, la récompense est soit 0 soit 10, elle est donnée en fonction d'une loi de probabilité de Bernouilli. Le meilleur opérateur a la meilleure récompense de 10 avec une probabilité de $0,5$ et le second meilleur opérateur avec une probabilité de $0,4$. Dans le scénario aberrant, tous les opérateurs reçoivent la plus petite récompense de 0 avec une probabilité $0,9$; le meilleur opérateur reçoit une récompense de 50 avec une probabilité $0,1$, le second meilleur reçoit une récompense de 40 avec une probabilité de $0,1$ et ainsi de suite. Ces scénarios sont utilisés afin d'analyser des stratégies de sélection adaptative lorsqu'il y a une transition de phase.

Le modèle proposé par Goëffon [GLS16] prend en compte le nombre de fois qu'un opérateur est utilisé pour déterminer la prochaine récompense de celui-ci. Plus l'opérateur est utilisé, plus la récompense est réduite.

2.3 ALGORITHMES D'OPTIMISATION : LES MÉTAHEURISTIQUES

Dans cette section, nous présentons les principaux algorithmes d'optimisation combinatoire pour les problèmes difficiles dont aucun algorithme efficace n'est connu. On distingue en général les algorithmes à une solution et les algorithmes à population de solutions (classification établie par [Tal09]). Les algorithmes sont présentés par ordre chronologique.

Les algorithmes peuvent avoir d'autres paramètres, par exemple un paramètre de l'algorithme choisi en fonction de la représentation de la solution et du paysage de fitness du problème.

OPÉRATEUR DE VARIATION La plupart des métaheuristiques reposent sur un ou plusieurs opérateurs de variation, encore appelés opérateur de recherche locale. Un opérateur de variation modifie une solution pour en former une autre.

Dans le contexte des algorithmes évolutionnaires, l'opérateur de variation fait partie de la composante exploration et est déterminant sur les performances de celui-ci. Il existe une infinité d'opérateurs de variation, dépendant de la représentation numérique de la solution [EFMoo].

Par exemple, l'opérateur de mutation peut prendre la valeur d'une variable selon une loi de probabilité. Pour les problèmes dont l'espace de recherche est l'espace des permutations, il est possible de réaliser les opérateurs de mutation par permutation : mutation par échange de deux valeurs (choisir deux valeurs aléatoirement de la chaîne et les permuter), mutation par insertion (choisir deux valeurs aléatoirement de la chaîne et les insérer de façon contiguë, mutation par brouillage (prendre un sous-ensemble de la chaîne et mélanger les valeurs aléatoirement) ou encore mutation par inversion (prendre un sous-ensemble de la chaîne est inverser la sous-chaîne entre eux).

TAXONOMIE Les algorithmes comme le recuit simulé, la recherche tabou, et la recherche locale itérée sont des algorithmes à une solution ; les algorithmes comme les méthodes évolutionnaires, les algorithmes à colonies de fourmis sont des algorithmes à population de solutions. Le critère d'arrêt est un critère commun à toutes ces méthodes ; deux choix principaux sont couramment utilisés, soit un critère d'arrêt avec un budget, soit un critère d'arrêt fixé avec un niveau de fitness à atteindre.

2.3.1 *Algorithme à une solution*

LE RECUIT SIMULÉ – *simulated annealing* en anglais – est introduit indépendamment par Kirkpatrick [KGV83] en 1983, et par Cerny en 1985 [Čer85]. Cet algorithme est inspiré de la technique du recuit utilisé dans le domaine de la métallurgie. Cela consiste à porter le matériau à haute température, puis à l'abaisser lentement. Par transposition à l'algorithme du recuit simulé, la fonction objectif du problème ΔE est apparentée à l'énergie d'un matériau avec une température fictive. Contrôler la température, c'est contrôler le dilemme exploration/exploitation. À partir d'une solution initiale, une mutation élémentaire de la solution est réalisée, la nouvelle solution est acceptée, si elle est meilleure ; sinon elle est acceptée avec une probabilité $\exp(-\Delta E/T)$.

Il y a équilibre thermodynamique après un certain nombre d'itérations généralement déterminé empiriquement. Nous abaissons la température, avant de réaliser une nouvelle série de mutations. La loi de décroissance par paliers de la température est empirique, comme le critère d'arrêt de l'algorithme [Ing89]. Cet algorithme a été notamment appliqué avec succès au placement des composants électroniques dans l'objectif de minimiser la distance de communications [KGV83]. Dans cet algorithme, les paramètres principaux à régler sont : la température initiale, la loi de décroissance de la température et l'opérateur de mutation des solutions.

LA RECHERCHE TABOU – *tabu search* en anglais – proposée par Glover en 1986 [Glo86]. Cette méthode est construite autour de l'idée d'une mémoire tabou de solutions. À chaque itération, comme pour le recuit simulé, une mutation de la solution est effectuée. Si la solution est meilleure, elle est acceptée et utilisée à la prochaine itération sinon la solution est ignorée. Pour éviter qu'une solution soit considérée plusieurs fois, la mutation vers une solution devient tabou et ne peut plus être réalisée pendant un certain nombre d'itérations. La liste tabou est une mémoire à court terme des solutions visitées récemment. Il existe plusieurs stratégies de liste tabou possibles, la version classique est une file avec un nombre limité de solutions. Les paramètres de cette méthode sont l'implémentation de la liste tabou et l'opérateur de voisinage.

LE GRIMPEUR – *hill-climber* en anglais – est à la base de beaucoup de métaheuristiques. Il consiste à chercher une solution à partir d'une solution et d'une opération élémentaire et en sélectionner le meilleur voisin. La recherche de proche en proche s'arrête dès que plus aucun voisin n'est améliorant. Généralement, quand le problème est multimodal la meilleure solution trouvée se trouve dans un optimum local. Aucun élément de diversification n'apparaît dans ces algorithmes. Atteindre une solution globalement optimale est difficile.

Deux variantes de recherche locale peuvent être évoquées : le *first-improvement* et *best-improvement*. La différence entre ces deux variantes se situe dans le choix de la solution. Le *first-improvement* sélectionne aléatoirement la première solution améliorant disponible, et le *best-improvement* choisit la meilleure solution voisine parmi l'ensemble des solutions voisines accessibles par un opérateur de mutation élémentaire [OVT10].

Dans les recherches locales, les principaux paramètres sont l'opérateur de mutation et le critère d'arrêt.

LA RECHERCHE LOCALE ITÉRÉE – *Iterated local search* en anglais – à deux composants : un élément de diversification avec un algorithme de perturbation de la solution et un élément d'intensification avec une recherche lo-

cale (voir l'algorithme 1). Contrairement à l'algorithme de recherche locale comme le grimpeur, la recherche locale itérée permet d'échapper à un optimum local grâce à l'opérateur de perturbation. La meilleure solution est utilisée à chaque itération de l'algorithme, la composante de perturbation utilise la meilleure solution ; la solution résultante est utilisée par la recherche locale. La solution du résultat de la recherche locale sera retenue si et seulement si elle est considérée comme meilleure que la meilleure solution.

Trois paramètres sont à considérer dans cet algorithme : le choix de l'opérateur perturbateur, le choix de l'algorithme de recherche locale et le critère d'arrêt.

Algorithm 1 L'algorithme de la recherche locale itérée (cf. [LMS03])

1: $s_0 \leftarrow$ Génération_de_la_solution_initiale()
2: $s^* \leftarrow$ Recherche_Locale(s_0)
3: **repeat**
4: $\quad s' \leftarrow$ Perturbation(s^*, histoire)
5: $\quad s^{*'} \leftarrow$ Recherche_Locale(s')
6: $\quad s^* \leftarrow$ Acceptation_de_la_solution(s^*, $s^{*'}$, histoire)
7: **until** le critère d'arrêt est vrai

2.3.2 *Algorithme à population de solutions*

Les algorithmes à population de solutions [Hol92] font évoluer un ensemble de solutions à l'aide d'opérateurs de mutation, de croisement puis de sélection. Les solutions sont parfois nommées individus.

Nous présentons ici deux techniques parmi les plus utilisées : les algorithmes évolutionnaires et les algorithmes de colonies de fourmis. Ces algorithmes peuvent être portés assez aisément sur des architectures distribuées de par leur manipulation de population ; chaque nœud peut accueillir une sous-population voire un individu. La communication entre les nœuds permet de transmettre des individus et ainsi maintenir la diversité de population.

ALGORITHMES ÉVOLUTIONNAIRES – *evolutionary computation* en anglais –

Les algorithmes évolutionnaires font évoluer un ensemble de solutions afin de trouver la ou les solutions d'un problème d'optimisation. À l'origine, ces méthodes sont inspirées par la théorie de l'évolution de Charles Darwin. De nouveaux individus sont obtenus par variation aléatoire (mutation et croisement). Les individus les mieux adaptés mesurés par la fitness survivent et forment la génération suivante.

L'algorithme évolutionnaire est présenté à la figure 3. Pour construire une nouvelle population, l'algorithme utilise des opérateurs de mutation et de

croisement. Les opérateurs de mutation modifient un individu pour en former un autre ; les opérateurs de croisement engendrent un ou plusieurs enfants à partir de la combinaison d'au moins deux des parents.

1. Initialisation : l'initialisation de la population parente de μ individus est réalisée soit aléatoirement, soit par une autre méthode. Pour chaque individu une fitness est calculée ;
2. Sélection : la sélection a pour but de choisir λ individus qui engendreront la génération suivante. Il est possible de réaliser, par exemple une sélection par tournoi. Pour le tournoi, n individus sont choisis aléatoirement et seul le meilleur est sélectionné pour faire partie de la population suivante.

   Il existe d'autres manières de sélectionner les individus pour former la population suivante [EFM00].
3. Mutation : chaque individu de la population est muté et éventuellement croisé par paire pour engendrer des enfants. Les opérateurs de mutation et de croisement permettent d'exploiter ou d'explorer les solutions ;
4. Évaluation : l'évaluation de la population calcule la fitness à l'aide de la fonction objectif pour chaque solution ;
5. Remplacement : le remplacement de la population parent par une population enfant : le but est de constituer à partir des populations parent et enfant une nouvelle population pour l'itération suivante. En fonction des critères choisis, l'ensemble des parents peuvent être remplacés $(\mu, \lambda) - ES$, ou seulement retenir une proportion des meilleurs individus des parents et des enfants $(\mu + \lambda) - ES$;
6. Si le critère d'arrêt est satisfait, alors l'algorithme se termine sinon continuer à l'étape 2.

Cet algorithme a de nombreux paramètres : la taille de la population, le critère de sélection, le ou les opérateurs de mutation et de croisement, et le critère de remplacement. Ces choix sont dépendants du problème à résoudre.

La programmation génétique (PG) – *genetic programming* en anglais – est un cas particulier d'algorithmes évolutionnaires initié par [Koz92]. Les arbres sont utilisés comme représentation de la solution pour former un programme ou une expression algébrique.

L'idée générale est de faire évoluer l'expression par une série de croisements-mutations des nœuds et des feuilles de l'arbre. C'est utilisé par exemple par Fonlupt *et al.* [FR00] qui utilisent la PG pour évaluer la concentration d'éléments en fonction la couleur de l'océan. Robilliard *et al.* [RF16] l'ont appliqué afin de construire une intelligence artificielle dans le jeu 7 Wonder qui a la caractéristique d'être un jeu avec des informations cachées et des éléments aléatoires.

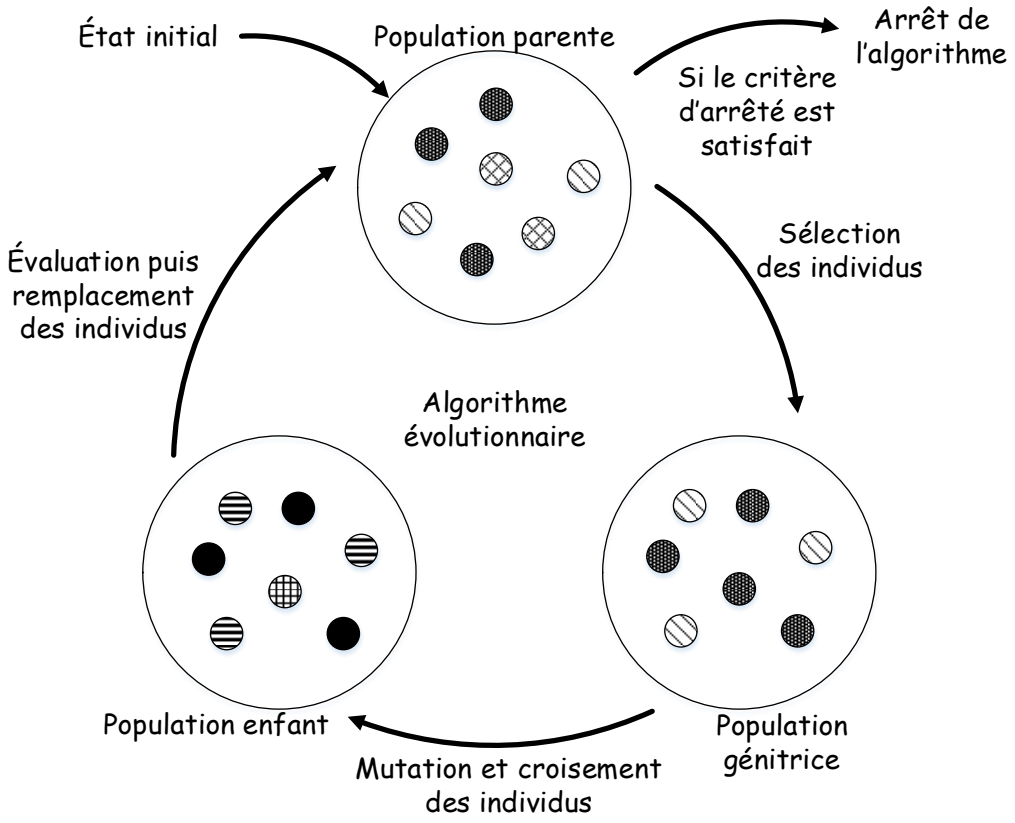


FIGURE 3 – Algorithme évolutionnaire avec les différentes étapes d'évolution de la population.

ALGORITHME DE COLONIES DE FOURMIS – *Ant colony optimization algorithms* en anglais –

Les algorithmes à colonie de fourmis sont inspirés par le fourragement des colonies de fourmis naturelles. Les fourmis sont des insectes sociaux ; elles vivent en communauté où chaque individu a une tâche qui lui est assignée en fonction de ses caractéristiques physiologiques.

L'une des tâches les plus importantes pour la fourmilière est la recherche de la nourriture. Un chemin pouvant le plus souvent être le plus court émerge par le mécanisme suivant. Pour trouver le plus court chemin, les fourmis utilisent pour communiquer de façon indirecte entre elles des molécules chimiques appelées phéromones. Elles en déposent sur le chemin afin que d'autres fourmis aient tendance à emprunter ce même chemin.

Plus un chemin est utilisé par les fourmis, plus il a des dépôts de phéromones, et plus il sera utilisé, c'est le renforcement positif. L'évaporation permet d'oublier un chemin. Si les fourmis n'empruntent plus le chemin, il est peu à peu oublié, on parle alors de renforcement négatif.

La figure 4 présente la découverte du plus court chemin par les fourmis. Au départ (a) les fourmis vont de A à E en ligne droite, puis un obstacle est introduit (b), les fourmis opèrent un choix probabiliste entre H et C en fonction de la quantité de phénomènes. À l'état stable en (c) le chemin ayant le plus de phéromone est préféré par les fourmis sachant que le chemin C à en moyenne un taux de phéromone plus important. Une fourmi seule ne peut trouver le plus court chemin, c'est l'interaction de groupes ou intelligence collective qui est à l'œuvre dans l'émergence ou l'apparition du plus court chemin. Les auteurs [CDM91 ; DMC96] ont proposé en premier l'algorithme bio-inspiré Ant System pour résoudre un problème d'optimisation combinatoire basé sur des fourmis artificielles.

2.4 PARAMÉTRAGE AUTOMATIQUE EN OPTIMISATION

La recherche automatique du paramétrage le plus approprié est étudiée depuis le début des algorithmes évolutionnaires. Ces algorithmes ont beaucoup de paramètres souvent difficiles à définir de façon précise ; dans le cadre d'une optimisation, chercher le paramétrage adapté pour un algorithme d'optimisation ou rechercher l'algorithme d'optimisation le plus performant est considéré comme étant un problème difficile.

Dans le cadre d'une résolution de problème de type boîte noire, il existe de nombreuses métaheuristiques comme la recherche locale itérée, le recuit simulé, la recherche par voisinage variable (variable neighborhood search) ou encore les algorithmes évolutionnaires.

Sans connaissance a priori des caractéristiques du problème, dans ces conditions, il n'est pas aisé d'établir l'algorithme le mieux adapté à la situation. Dans cette optique, Rice [Ric76] propose le premier d'extraire des

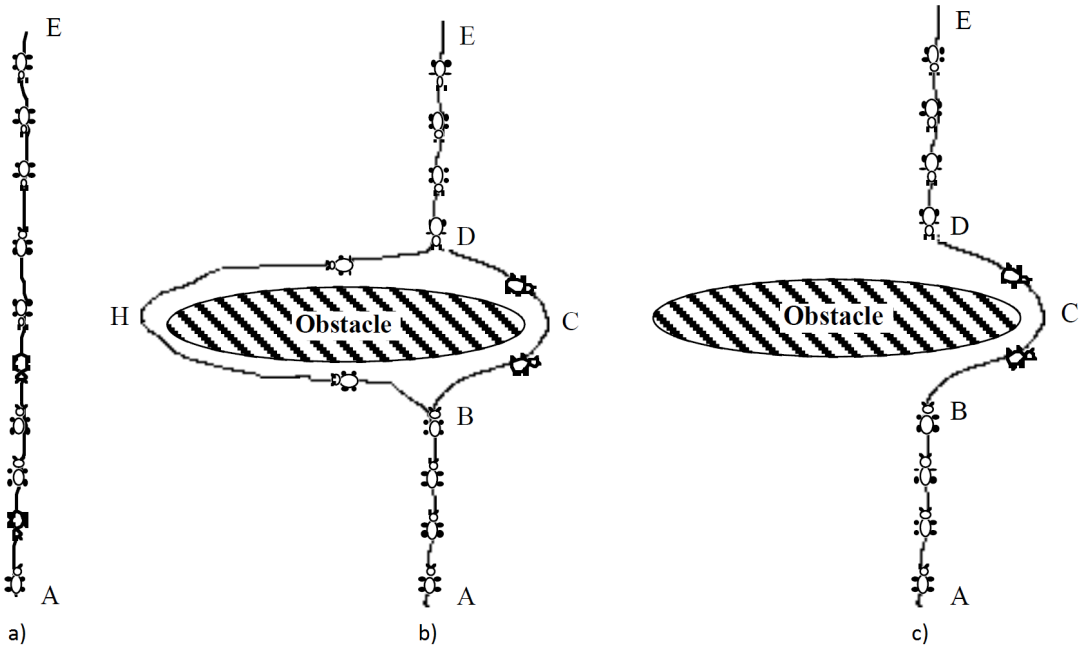


FIGURE 4 – Optimisation du chemin par les fourmis après l'apparition d'un obstacle. Figure de [CDM91]. En a) le chemin parcouru par les fourmis pour aller de A à E, en b) un obstacle apparaît et les fourmis doivent le contourner, en c) à l'état stable, la colonie de fourmis tend à choisir le chemin plus court et comme le chemin le plus court est parcouru plus rapidement par les fourmis. En moyenne, le taux de phéromone sera plus important sur le plus court chemin.

caractéristiques du problème (*cf.* figure 5) et en fonction de ces caractéristiques de sélectionner un algorithme afin de le mesurer sur le problème. Par exemple, la mesure peut-être réalisée en appliquant un algorithme et en analysant l'impact sur la solution. Ce cadre est devenu de plus en plus utilisé en optimisation.

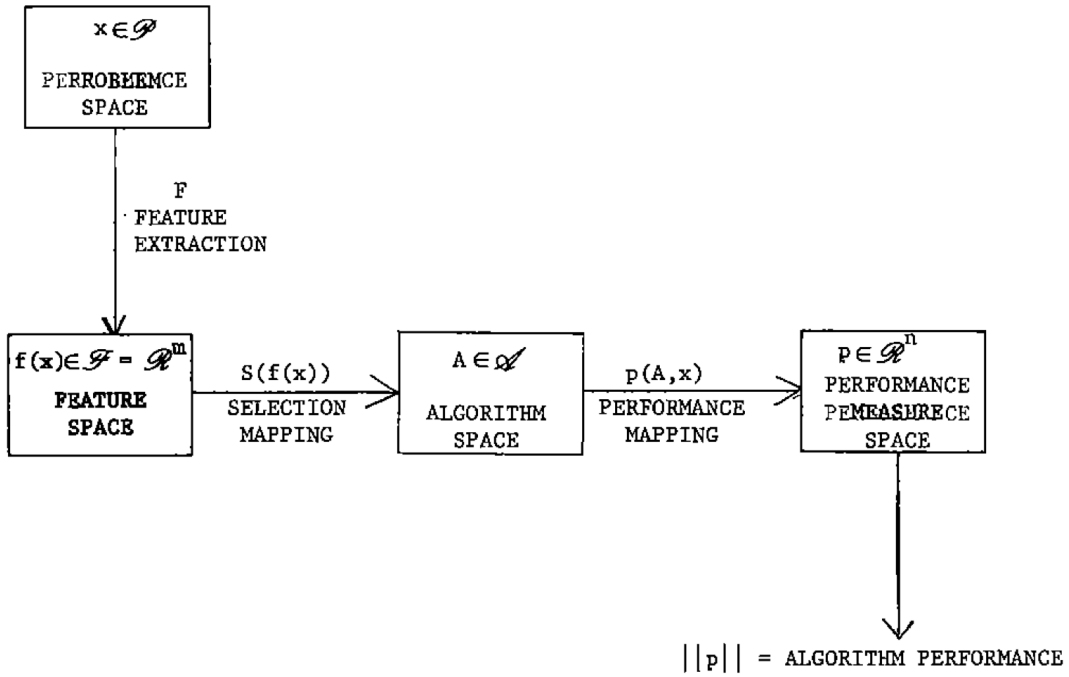


FIGURE 5 – Proposé par Rice [Ric76] en 1976, principe de sélection d'un algorithme basé sur l'espace des caractéristiques du problème.

Eiben *et al.* [Eib+07] (figure 6), proposent de classifier les méthodes de réglage des paramètres en deux classes : les méthodes avant l'optimisation (off-line) et les méthodes durant l'optimisation (on-line).

Dans les méthodes de sélection avant l'optimisation, la sélection de l'algorithme est réalisée avant la résolution du problème. Dans les méthodes durant l'optimisation, la sélection de l'algorithme est réalisée pendant la résolution du problème. Le choix de l'algorithme peut s'effectuer parmi un ensemble défini de paramètres – métaheuristiques – rassemblés dans un portefeuille de paramètres ou à partir d'une distribution de paramètres. Généralement, le choix de l'algorithme se fait en fonction de récompenses établies pour chaque algorithme de résolution.

Dans les méthodes de sélection durant l'optimisation, trois types de techniques se distinguent : déterministe, adaptative et auto-adaptative. Les techniques déterministes utilisent une règle déterministe définie en fonction du temps pour modifier les paramètres. Cela peut-être par exemple, utiliser un paramétrage pendant i itérations, et ensuite en changer pendant j itérations. Les techniques adaptatives cherchent à trouver le bon paramétrage

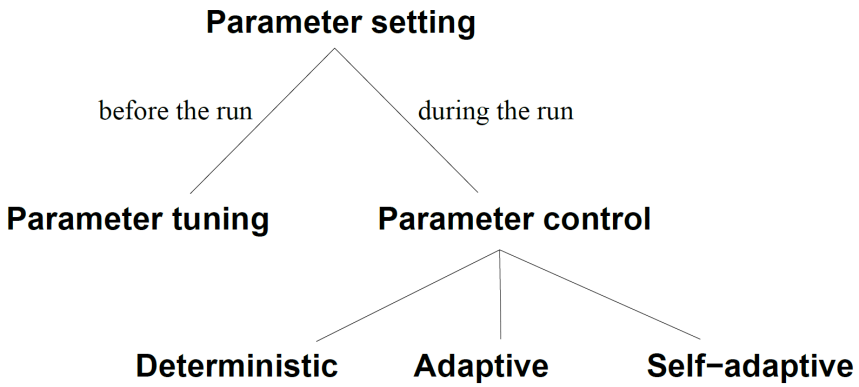


FIGURE 6 – Classification des types de paramétrage automatique, hiérarchie établie par Eiben dans [Eib+07].

pour un algorithme en fonction de l'état de la recherche. Les techniques auto-adaptatives ont une capacité à modifier ce propre paramétrage afin de s'adapter au problème à chaque itération. Dans cette technique, le choix du paramétrage est codé dans la solution, le paramétrage est donc soumis au processus de mutation au même titre que la solution du problème.

Nous présentons dans cette section les différentes stratégies de sélection dans le cadre off-line et on-line. Les stratégies de sélection adaptatives on-line sont présentées plus en détail aux sections 2.5 et 2.6. Dans cette section, nous discuterons en particulier des stratégies de sélection adaptative, déterministe et auto-adaptative.

2.4.1 *Avant l'optimisation*

Le réglage automatique des paramètres avant l'optimisation, également nommé *off-line tuning*, est la recherche du meilleur paramètre pour un algorithme réalisé avant l'optimisation. Quand le meilleur paramétrage est déterminé, l'optimisation est effectuée.

Plusieurs méthodes utilisent la performance de prédiction basée sur les caractéristiques du problème pour sélectionner l'algorithme ou le paramétrage tel que SATzilla [Xu+08].

L'optimisation de paramètres peut également être réalisée de manière statistique. La technique *iRace* [Lóp+11] [BBS07] [Bir+02] fonctionne comme suit : échantillonnage de nouvelles configurations en fonction d'une distribution statistique pour ensuite sélectionner les meilleures configurations et ainsi mettre à jour la distribution. L'algorithme recommence tant que le critère d'arrêt n'est pas atteint. Actuellement, les méthodes de tunning off-line peuvent bénéficier des systèmes de calcul parallèle pour effectuer de longues expérimentations pour apprendre les bons paramètres.

CALIBRA [AL06] est basé sur un échantillonnage de solutions réalisées expérimentalement. Pour chaque paramètre, la meilleure et la pire valeurs sont retenues. Une recherche locale en utilisant les résultats expérimentaux est réalisée pour réduire la distance entre la meilleure et la pire valeur.

La technique MADS [AO06] est basée sur le maillage de l'espace de recherche, la taille du maillage, du rayon de recherche et de la direction. À chaque itération, chaque maille est échantillonnée, le centre du maillage est déplacé vers la meilleure valeur.

L'optimisation de paramètres est réalisable également avec des métaheuristiques. Grefenstette [Gre86] propose de trouver les paramètres d'un algorithme génétique avec un algorithme génétique. L'idée est de sélectionner une représentation et un opérateur. Un test est effectué et une évaluation de la prédiction du modèle est réalisée. Si le résultat de la prédiction est concluant l'algorithme évolutionnaire est exécuté.

2.4.2 *Durant l'optimisation*

Dans les stratégies de sélection durant l'optimisation (figure 7), les méthodes de contrôle *on-line* ou les stratégies de sélection adaptative, le paramétrage est sélectionné tout au long du processus d'optimisation.

À chaque itération, un paramétrage est sélectionné depuis un portefeuille d'algorithmes selon les performances obtenues à la précédente itération. Ce problème de sélection on-line peut être modélisé par le problème du bandit manchot dynamique [ACF02] : chaque bras est un algorithme à optimiser, la récompense est basée sur la qualité de la solution produite par chaque algorithme. L'objectif est de sélectionner le bras durant le processus d'optimisation afin de maximiser la qualité de la solution finale.

Les stratégies de sélection adaptative durant l'optimisation permettent :

1. de définir à chaque état de la recherche un paramétrage le plus performant pour des environnements non-stationnaires,
2. une succession de paramétrage aux propriétés différentes de telle sorte qu'un paramétrage utilisé au temps t peut influencer un autre paramétrage au temps $t+1$,
3. Comme la recherche d'un paramétrage performant et l'optimisation se font conjointement. À la fin de l'optimisation, l'expert tire deux informations de l'optimisation : le paramétrage à utiliser pour les prochaines optimisations d'une même classe de problème et la solution du problème à optimiser.

Plusieurs stratégies de sélection ont été proposées afin d'optimiser le problème du bandit manchot. Thierens [Thi05] utilise adaptive pursuit pour effectuer la sélection.

Fialho *et al.* [Fia+10a] proposent différentes stratégies de sélection basée sur la stratégie Upper Confidence Bounds avec une technique de redémarrage dynamique. Ils seront explicités dans la section 2.5.2.

Les hyper-heuristiques construisent en fonction de règles une heuristique dans l'espace des heuristiques possibles. Burke [Bur+10] propose une classification des hyper-heuristiques. Il distingue notamment deux types d'hyper-heuristiques : les hyper-heuristiques de sélection et les hyper-heuristiques de génération. Les hyper-heuristiques de sélection produisent une combinaison d'heuristiques pré-existantes.

Ils correspondent aux stratégies de sélection par portefeuille étudiées dans ces travaux. Les hyper-heuristiques de génération construisent une heuristique en utilisant des composants de base.

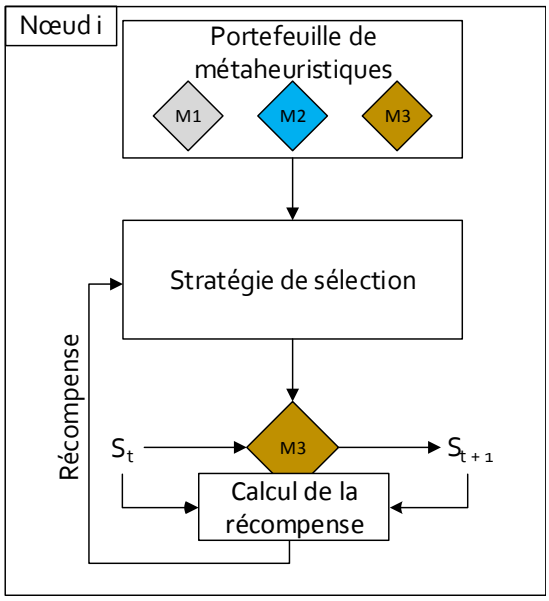


FIGURE 7 – Stratégie de sélection indépendante choisissant une métaheuristique parmi un ensemble en fonction des récompenses obtenues.

2.4.3 *Distinction entre paramètre, algorithme et opérateur*

La stratégie de sélection choisit un paramétrage parmi un portefeuille (voir figure 8). En fonction du contexte, la stratégie de sélection fait un choix entre des paramètres, des algorithmes, ou des opérateurs. Par abus de langage, nous parlons la plupart du temps de paramètres, mais il pourra arriver de parler d'opérateurs, ou de métaheuristiques indifféremment. Le composant paramétrable peut avoir une portée plus ou moins réduite dans l'algorithme. Il est possible de concevoir un algorithme dont son paramètre désigne un algorithme sous-jacent particulier, par exemple opter entre un recuit-simulé

ou une recherche tabou. Il peut être aussi un élément plus atomique de l'algorithme comme le choix de l'opérateur : le taux de l'opérateur de mutation, ou encore le type de croisement. La stratégie de sélection peut elle-même avoir des paramètres, pour lever toute ambiguïté, nous les nommerons méta-paramètres.

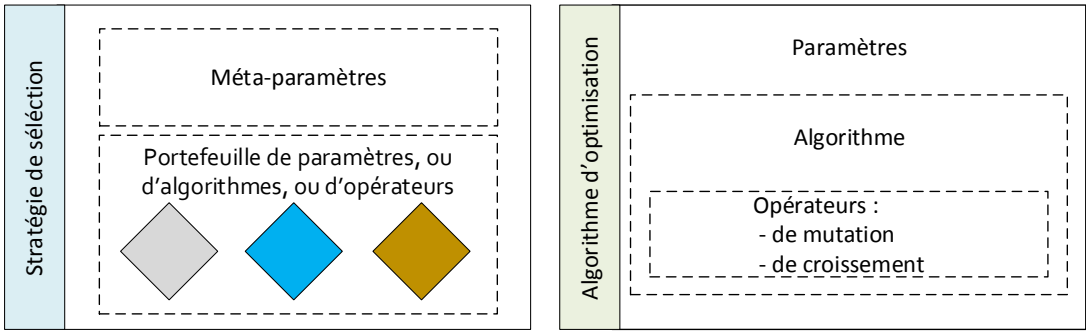


FIGURE 8 – Une stratégie de sélection peut choisir entre des paramètres, des algorithmes ou des opérateurs. Le choix peut se porter sur les paramètres de l'algorithme, sur l'algorithme, ou sur les opérateurs de l'algorithme d'optimisation.

2.5 SÉLECTION ADAPTATIVE SÉQUENTIELLE D'ALGORITHMES

Nous présentons dans un premier temps la définition des méthodes de sélection d'algorithmes adaptatifs séquentielles, dans un second temps différentes stratégies de sélection et par la suite le calcul de la récompense des stratégies de sélection.

2.5.1 *Définition de la stratégie de sélection*

Un algorithme d'optimisation a des paramètres. Un choix du paramétrage détermine le temps de résolution de l'optimisation et la qualité de la solution. La stratégie de sélection adaptative choisit un paramétrage à chaque itération durant l'optimisation. Ce choix s'effectue suivant les récompenses obtenues par le passé pour un paramétrage donné et une technique d'apprentissage automatique.

Définition 1 (La récompense d'un paramètre pour une stratégie de sélection) *La récompense est l'agrégation des mesures de la performance d'un paramétrage.*

Soit $\theta = (\theta_0, \theta_1, \cdots, \theta_{Tmax})$ un vecteur de paramètres. Nous appelons un algorithme d'optimisation itératif **A**. Cet algorithme applique à chaque itération l'opérateur de paramètre θ_t à la population de recherche courante :

1: $x \leftarrow$ initialisation()
2: **repeat**
3: $\quad x \leftarrow \text{operateur}_{\theta_t}(x)$
4: **until** le critère d'arrêt soit vrai
5: **return** x

Soit Coût($\mathbf{A}, f$) une mesure de coût de l'algorithme **A** pour optimiser la fonction f. Une configuration optimale des paramètres est une configuration qui minimise la fonction de coût :

$$\theta^* = \underset{\theta \in \Theta}{argmin}\ \text{Coût}(\mathbf{A}_\theta, f) \tag{4}$$

Un réglage de paramètres avant l'optimisation consiste à trouver θ_* tel que $\forall t \leqslant 0, \theta_t = \theta_*$. Lorsque les valeurs θ_t n'ont pas toutes la même valeur, on parle de réglage pendant l'optimisation (on-line). Une stratégie de sélection peut se définir ainsi :

1: $\theta_0 \leftarrow$ initialisation()
2: $x_t \leftarrow$ initialisation_de_la_solution()
3: **repeat**
4: $\quad x_{t+1} \leftarrow \text{operateur}_{\theta_t}(x_t)$
5: $\quad R_t \leftarrow calcul\_recompense(x_{t+1}, x_t)$
6: $\quad \theta_{t+1} \leftarrow \text{sélection}((\theta_0, \cdots, \theta_N), (R_0, \cdots, R_t), (x_0, \cdots, x_{t+1}))$
7: **until** le critère d'arrêt soit vrai

2.5.2 *Différentes méthodes*

Les stratégies de sélection de référence nous donnent une base de comparaison avec des stratégies de sélection. Nous les utilisons tout au long de ce manuscrit. Les stratégies de sélection de référence sont : constantes, uniformes, aléatoires, et oracle. La stratégie constante à l'initialisation détermine pour chaque nœud de calcul aléatoirement une métaheuristique. Ensuite à chaque itération, les métaheuristiques utilisées sont toujours les mêmes.

Pour la stratégie de sélection uniforme, une métaheuristique est choisie comme paramètre pour l'ensemble des itérations et des nœuds. La stratégie de sélection aléatoire affecte à chaque itération une métaheuristique à un nœud aléatoirement.

Une variante de la stratégie de sélection aléatoire est une sélection aléatoire avec proportion. Chaque métaheuristique est choisie en fonction d'une proportion pour un nœud. La proportion est fixée avant l'exécution comme paramètre.

La stratégie de sélection oracle affecte à chaque nœud et à chaque itération la meilleure métaheuristique.

Intuitivement, la meilleure stratégie de référence est l'oracle, aucune stratégie de sélection adaptative ne fait mieux que l'oracle. À l'opposé de l'oracle, la stratégie de sélection aléatoire est la pire stratégie de sélection. La stratégie constante dépend du contexte.

Un certain nombre de techniques d'apprentissage automatique a été proposé pour la sélection en ligne et adaptative des algorithmes à partir d'un portefeuille donné. Dans les premiers travaux de Grefenstette [Gre86], une technique standard consiste à prédire la performance d'un ensemble d'opérateurs en utilisant une régression linéaire simple et l'état de la forme moyenne actuelle de la population. Cette méthode permet ainsi de sélectionner le meilleur opérateur à choisir en fonction de la prédiction donnée par la régression.

Les travaux récents intègrent ce problème de sélection dans le cadre du bandit multi-bras traitant plus explicitement du compromis entre l'exploitation du meilleur paramétrage identifié et l'exploration des algorithmes potentiellement sous-estimés restants.

Une stratégie parmi les plus simples est la stratégie ϵ-Greedy [SB98] qui consiste à sélectionner l'algorithme avec la meilleure performance estimée au taux $(1-\epsilon)$ et un algorithme aléatoire avec un taux ϵ. Cependant, des stratégies plus avancées sont disponibles comme nous les décrirons par la suite.

La stratégie Upper Confidence Bound (UCB) [ACF02] est une méthode importante. Elle fait partie de la classe des algorithmes de probability matching.

Cette méthode consiste à estimer la borne supérieure de l'intervalle de confiance de l'espérance de la récompense de chaque bras par $\hat{\mu}_i + C \cdot e_i$; où $\hat{\mu}_i$ est la récompense moyenne (empirique) estimée, et e_i est l'écart-type de la prédiction. Elle sélectionne ensuite les algorithmes à la meilleure borne supérieure (pour un problème de maximisation). Le paramètre C permet de régler le compromis exploitation/exploration. Dans le cadre de la sélection d'algorithmes [Fia+10a] où les bras ne pouvaient être ni indépendants ni stationnaires, l'estimation de la récompense attendue est affinée à l'aide d'une fenêtre glissante W où seules les observations précédentes de la performance sont prises en considération.

Une version dynamique de UCB à été proposée par DaCosta [DaC+08] connue sous le nom de Page-Hinkley. L'ensemble des variables internes à la stratégie est réinitialisé quand le test Page-Hinkley détecte statistiquement un changement, lorsque le meilleur bras a une performance qui diminue brusquement.

La stratégie Adaptive Pursuit (AP) [Thi05] est une autre technique utilisant une moyenne pondérée pour estimer la récompense attendue avec un

paramètre α – nommé aussi *adapation rate* en anglais – pour régler le taux d'adaptation de l'estimation. Plus formellement :

$$\hat{q}_{i,t+1} = (1 - \alpha) \cdot \hat{q}_{i,t} + \alpha \cdot r$$

où $\hat{q}_i$ est l'estimation de la récompense et r la récompense pour un paramétrage.

Ceci sert à définir la probabilité p_i sélectionnant chaque paramétrage du portefeuille. À chaque itération, la valeur de probabilité p est mise à jour pour chaque paramétrage i selon un taux d'apprentissage β – nommé aussi *learning rate* en anglais –, ce qui permet pour le meilleur paramétrage de voir sa probabilité augmentée et diminuée pour les autres. Plus formellement :

$$\begin{cases} i_t^* = arg\ max_{i=1\ldots K}\{\hat{q}_{i,t}\} \\ p_{i,t+1} = \begin{cases} p_{i,t} + \beta\ .\ (p_{max} - p_{i,t}), & \text{si } i = i_t^* \\ p_{i,t} + \beta\ .\ (p_{min} - p_{i,t}), & \text{sinon} \end{cases} \end{cases}$$

Les stratégies de sélection ont des paramètres. Ces paramètres règlent le facteur d'oubli des récompenses passées et le compromis entre l'exploitation et l'exploration du paramétrage. Le compromis entre l'exploitation et l'exploration apparaît de différente forme. Dans ϵ-Greeedy, il y a une exploration aléatoire des paramétrages quand le taux est plus grand que $1 - \epsilon$. Dans UCB, la variable C joue ce rôle. Dans AP (Adaptive Pursuit) c'est la variable β ainsi que le mécanisme de roulette qui jouent ce rôle.

2.5.3 *Récompense dans les méthodes adaptatives*

La récompense – également nommée *reward*, *credit assignment* en anglais – de chaque paramétrage permet à la stratégie de sélection de prendre une décision sur le bon paramétrage à utiliser à une itération donnée. Il faut distinguer les mesures réalisées pour ensuite construire la récompense d'un paramétrage. Deux questions peuvent être dégagées :

— Quelle est la granularité de la mesure ?
— Comment calculer à partir des mesures une récompense pour un paramétrage ?

Quand nous parlons de granularité de la mesure, cela signifie que la mesure peut-être réalisée à différents niveaux dans l'algorithme. La récompense peut être calculée pour la différence entre deux itérations de l'algorithme [DV11] ou peut être calculée entre chaque mutation de solution [Fia+09].

À partir des mesures, le calcul de la récompense se pose. Il est possible d'avoir plusieurs mesures pour un paramétrage : soit des mesures réalisées

par le passé, soit des mesures réalisées en parallèle, soit les deux. Deux éléments peuvent être intégrés dans le calcul de la récompense : le facteur d'oubli et l'agrégation des mesures par paramétrage.

LE FACTEUR D'OUBLI. L'ensemble des récompenses passées dans le calcul de la pertinence du paramétrage est étudié avec un facteur d'oubli par fenêtre glissante ou par test statistique. Lobo [LLG97] propose de mettre un poids c sur la récompense passée W_t et R sur la récompense actuelle $W_t + 1 = W_t(1 - c) + c \times R$ afin de prendre plus ou moins en compte la nouvelle information. Fialo [Fia10] compare pour la technique bandit multi-bras un facteur d'oubli basé sur une fenêtre glissante et le test Page-Hinkley. La fenêtre glissante considère qu'après W itérations l'information n'est plus pertinente. Le test Page-Hinkley permet un effacement de la mémoire des récompenses passées en supposant que les nouvelles récompenses sont plus pertinentes. D'ailleurs, certains auteurs ont montré que la récompense maximale sur une fenêtre coulissante améliore la performance par rapport à la moyenne sur certains problèmes combinatoires [Fia+10a ; Can+13]. Aucune analyse fondamentale de ce résultat n'a été donnée.

L'AGRÉGATION DES MESURES. Plusieurs types d'agrégations de récompenses sont proposés : la moyenne de l'ensemble des mesures, la médiane ou encore la mesure maximale. Quand la mesure est négative, il est possible de considérer une mesure nulle à la place. Candan *et al.* [Can+13] considèrent le point de vue global du modèle en île. Il agrège l'ensemble des mesures produites par les îles par une moyenne et la fonction maximum.

Dans les algorithmes génétiques, la récompense peut être calculée non seulement en fonction de la qualité mais aussi de la diversité de la population [Mat+09]. Dans d'autres études [Fia+10b], une approche basée sur le classement est adoptée afin d'éviter les problèmes de normalisation des performances. Dans le cadre de la sélection parallèle d'algorithmes adaptatifs, l'estimation de la qualité de chaque algorithme disponible est également une question difficile car non seulement une, mais plusieurs instances d'algorithmes peuvent être exécutées à chaque itération.

2.6 SÉLECTION ADAPTATIVE DISTRIBUÉE D'ALGORITHMES

L'environnement parallèle est intéressant lorsque le coût de calcul de la fitness est plus élevé que la gestion de l'environnement distribué.

Deux modèles de parallélisations sont étudiés : (i) le modèle en île, et (ii) le modèle maître-esclave. (i) Le modèle en île est défini comme un ensemble de nœuds de calcul (île) réalisant des tâches de façon indépendante, il n'y a pas de coordinateur global. Cependant les nœuds de calcul peuvent partager des informations. Dans le modèle en île, chaque île prend des décisions

localement en fonction des informations voisines. Elles sont organisées en réseau, en fonction d'une topologie, pour communiquer des informations. (ii) Le modèle maître-esclave possède un ensemble de nœuds esclaves indépendants réalisant des tâches communiquées par le nœud maître. C'est une architecture centralisée parallèle.

Pour résoudre un problème d'optimisation, l'échange d'information est un élément important qui impacte la qualité de l'optimisation. C'est un élément de collaboration entre les nœuds de calcul. Deux formes de communication sont envisageables : (i) synchrone, et (ii) asynchrone. (i) En mode synchrone, l'ensemble des nœuds communique au même moment et généralement à la même fréquence. Pour le modèle en île, à chaque itération de l'algorithme l'ensemble des nœuds communique avec son voisinage sur l'état de l'optimisation. Pour le modèle maître-esclave, le maître communique avec l'ensemble des nœuds esclave en même temps. (ii) En mode asynchrone, chaque nœud de calcul communique une information quand elle est nécessaire. Lorsqu'un nœud a terminé une tâche, il en communique les résultats à un ou d'autres nœuds de calcul. L'avantage du mode synchrone est sa facilité d'implémentation. Par contre, un nœud de calcul doit attendre le bon moment pour communiquer, il perd du temps de calcul à attendre. L'avantage du mode asynchrone est le traitement immédiat des communications, l'inconvénient est la gestion des informations parfois désuètes par rapport à l'état de la recherche. L'ensemble des nœuds de calcul peut travailler à une allure différente (type de processeurs ou tâche différente). L'information sur l'état de la recherche peut être de qualité hétérogène.

Pour chaque modèle, un compromis global-local intervient dans le partage de l'information. Chaque nœud réalise une somme de tâches. Pour la réaliser aux mieux, le nœud de calcul prend des décisions locales qui peuvent impacter l'optimisation dans son ensemble.

Bien que plus difficile à concevoir, notamment la synchronisation entre les nœuds de calcul, le modèle en île se prête mieux au passage à l'échelle que le modèle maître-esclave. Le passage à l'échelle pour le modèle maître-esclave est tributaire de l'architecture réseau sous-jacente. Un sous dimensionnement du réseau entre le maître et les esclaves ralentit voire interrompt les communications. Il y a donc un compromis entre la quantité d'informations à communiquer pour l'algorithme et la quantité supportée par le réseau.

2.6.1 *Modèle en île*

Les modèles en île ont beaucoup été étudiés notamment par [Can98 ; Tom05 ; Skooo]. Ils permettent de résoudre des problèmes avec des algorithmes de résolution naturellement parallélisable. Il existe de nombreuses façons de construire un modèle en île en fonction de caractéristiques comme la taille de population sur chaque île, les algorithmes associés aux îles, et la migra-

tion des solutions entre les îles. Dans ce type de modèle, nous rencontrons le compromis entre les décisions locales et globales ; c'est-à-dire une décision optimale locale est-elle une décision globalement optimale ? Dans le modèle en île homogène, l'ensemble des îles utilise les mêmes algorithmes contrairement à un modèle en île hétérogène. Dans un modèle en île à grain grossier (coarse-grained), chaque île représente une sous-population contrairement à un modèle en île à grain fin (fine-grained) où l'ensemble des îles représente une population avec sur chaque île un individu. Pour les modèles à grain grossier, la migration des individus peut être une sous-population voire un individu. Pour les modèles à grain fin, cela est généralement une migration à singleton pour chaque île. Dans un modèle en île hétérogène, chaque île applique un algorithme évolutionnaire potentiellement différent d'un autre nœud de calcul. Contrairement au modèle à île homogène, chaque île applique le même algorithme.

Dans ces travaux, nous nous inscrivons dans un modèle en île hétérogène et à grain fin. La taille de population correspond aux nombres de nœuds de calcul. Chaque île peut exécuter un algorithme différent par rapport à une autre île, et chaque île a un seul individu.

Un ensemble de métriques a été étudié pour mieux comprendre la dynamique de population en fonction de paramètres. Il en ressort deux paramètres importants dans le modèle en île : le paramètre de la politique de migration et le paramètre qui définit l'algorithme sur chaque nœud.

Candan *et al.* [**Skolicki:2007** ; Can+12 ; Sor+15] proposent une politique de migration adaptative avec un modèle en île hétérogène où chaque île peut appliquer son propre algorithme évolutionnaire. Le paramètre p_{ij} est utilisé pour définir le taux de migration entre les îles i et j. Selon la performance de l'île à produire des solutions prometteuses, le taux sera mis à jour en utilisant le principe d'apprentissage par renforcement. Ferdandez *et al.* [Fer+14] proposent une méthode de contrôle de la politique de migration d'algorithme évolutionnaire quand la population est spatialement structurée suivant une grille dans un plan. La migration est contrôlée par le déplacement des solutions sur la grille, soit de façon aléatoire, soit vers une cellule de solutions similaires.

Pour le contrôle des paramètres de l'algorithme évolutionnaire : au lieu d'utiliser les mêmes paramètres sur chaque île, pour les modèles en île hétérogène, chaque île applique un algorithme particulier. Afin de démontrer l'utilité d'un tel modèle hétérogène, Tanable *et al.* [GF11 ; TF13] montrent qu'une collection de paramètres aléatoires offre de meilleures performances qu'un réglage constant. L'étude se concentre sur l'optimisation continue et les algorithmes d'évolution différentielle ainsi que sur deux classes de problèmes combinatoires (QAP, TSP) utilisant un algorithme génétique simple. Suivant des idées similaires, Garcia-Valdez *et al.* [Gar+14] ont montré que pour l'EA pool-based distribué qui est un autre modèle d'île hétérogène, un

ensemble aléatoire de paramètres utilisé par un algorithme génétique simple sur les problèmes P-Peaks dépasse un cadre de paramétrage constant.

Cependant, dans les modèles en île hétérogène, le réglage aléatoire des paramètres n'est pas la seule possibilité. En fait, chaque algorithme évolutionnaire associé à chaque île peut être contrôlé durant le processus d'optimisation selon l'état de la recherche de l'itération précédente. Par exemple Tongchim *et al.* [TC02] proposent de sélectionner les paramètres (croisement et mutation) avec un simple algorithme évolutionnaire adaptatif. Deux ensembles de paramètres sont comparés sur le même nœud d'ordinateur et le meilleur réglage de paramètre avec la meilleure solution est envoyée aux autres îles. Les auteurs affirment que des techniques adaptatives améliorent les performances par rapport à des stratégies statistiques ou aléatoires.

Le cadre DAMS (Distributed Adaptive Metaheuristic Selection) [DV11] propose de sélectionner localement à chaque itération et pour chaque nœud une métaheuristique depuis un portefeuille de métaheuristiques afin de maximiser les performances de l'ensemble du système distribué. Chaque nœud de calcul utilise une stratégie de sélection, une métaheuristique est sélectionnée non seulement selon la performance précédente du nœud mais aussi selon la performance observée et communiquée par les autres nœuds. Derbel *et al.* [DV11] proposent une simple stratégie de sélection, mais également une stratégie efficace appelée Select-Best-and-Mutate (SBM). Cette dernière est une méthode dérivée de ϵ-Greedy dans le cadre distribué. Dans ce papier, les auteurs proposent d'analyser d'autres alternatives de stratégies de sélection en s'inspirant des stratégies existantes des bandits multi-bras, mais dans un modèle distribué (île).

2.6.2 *Modèle maître-esclave*

L'architecture maître-esclave est une organisation classique en informatique, le nœud maître regroupe les informations et prend des décisions pour l'ensemble des nœuds esclaves. Les nœuds esclaves ont une autonomie relativement restreinte puisqu'ils ne font qu'exécuter les commandes du nœud maître. L'architecture maître-esclave a été largement étudiée dans le calcul évolutionnaire [DGP06]. En fait, cette architecture est simple à mettre en œuvre et ne nécessite pas d'opérations parallèles sophistiquées.

Le nombre de messages est généralement proportionnel au nombre de nœuds ce qui empêche les problèmes d'inondation du réseau. Cependant, son inconvénient majeur et bien connu est le goulot d'étranglement de la communication autour du maître qui peut conduire à des problèmes de latence et une perte de robustesse, par exemple en cas de pannes ou d'arrêt du maître.

Le mode synchrone peut permettre d'avoir une vision plus globale du système distribué qui peut être d'une importance cruciale pour mieux co-

ordonner les esclaves [WRM16]. À chaque itération, toute l'information est disponible sur le maître et peut être utilisée pour prendre une meilleure décision pour la prochaine génération de l'algorithme parallèle.

3

MODÈLE EN ÎLE À RÉCOMPENSE COLLECTIVE

Dans l'optique de résoudre des problèmes d'optimisation avec une infrastructure informatique complètement décentralisée, le modèle en île est utilisé comme modèle distribué pour résoudre des problèmes d'optimisation. Comme introduit dans le chapitre 2, ils sont une des manières les plus naturelles de parallélisation des algorithmes évolutionnaires (EA) [Tom05]. À travers cette architecture, nous étudions différentes stratégies de sélection adaptative qui partagent de l'information entre les nœuds de calcul. En utilisant les informations des nœuds voisins, l'objectif est d'améliorer la décision du choix d'un opérateur par la stratégie de sélection du nœud local.

Le modèle en île est composé d'îles correspondant à des nœuds de calcul et des liens entre les îles en fonction d'une topologie. Des règles de migration entre les îles des solutions et des récompenses sont définies, elles peuvent être déterminées à l'initialisation ou dynamiques durant l'optimisation. À chaque île est associé un opérateur qui est appliqué sur une sous-population. Cet opérateur peut changer dans le temps et être différent entre deux îles. Ces travaux sont une extension du cadre DAMS proposée par Derbel et Verel en 2011 [DV11]. Les auteurs proposent un cadre générique pour résoudre des problèmes d'optimisation dans un environnement distribué, nous le présentons plus en détail par la suite. L'objectif est de minimiser le temps d'optimisation en maximisant la performance du système distribué dans son ensemble. Dans ce type d'organisation, la notion de global-local intervient. Chaque nœud est autonome par rapport aux autres, il prend des décisions, mais ces décisions locales sont-elles globalement optimales ?

La conception de stratégie de sélection dans un environnement distribué dépend des récompenses disponibles pour le nœud local, la manière de calculer la récompense, et les techniques d'apprentissage automatique. Nous nous sommes inspirés des techniques d'apprentissage existant dans le contexte séquentiel pour proposer de nouvelles techniques pour les environnements distribués. Nous analysons leur performance, et les comparons. Ces travaux visent à répondre aux questions suivantes :

1. Comment élaborer une stratégie de sélection au sein d'un environnement distribué de modèle en île ?
2. Quel type d'informations partager entre les îles ?
3. Quelles sont les performances des différentes stratégies de sélection d'algorithmes ? Comment leurs paramètres respectifs influencent-ils les performances de ces stratégies ?

Nous présentons en détail l'algorithme DAMS, ainsi que les stratégies de sélection indépendante et collective. Dans un second temps nous étudions ce cadre avec des expérimentations sur le problème OneMax et le paysage NK. Nous présentons les comparaisons des stratégies de sélection entre ceux indépendants et collectifs.

3.1 SÉLECTION ADAPTATIVE DE MÉTAHEURISTIQUES DISTRIBUÉES

Nous présentons le cadre distribué permettant de résoudre des problèmes d'optimisation à l'aide de portefeuille d'opérateurs. Différentes stratégies de sélection sont étudiées afin de choisir l'opérateur le plus adapté à chaque itération de l'algorithme de l'optimisation. Deux catégories stratégie de sélection en environnement distribué sont présentées : les stratégies de sélection indépendante et collective présentées ci-dessous.

3.1.1 *Le cadre « Distributed Adaptive Metaheuristic Selection » (DAMS)*

L'algorithme Distributed Adaptive Metaheuristic Selection (DAMS) est basé sur le modèle en île hétérogène où chaque île peut appliquer une métaheuristique différente. Il utilise une stratégie de sélection de métaheuristiques génériques et propose une manière originale de faire communiquer les nœuds entre eux.

Les échanges d'information s'effectuent de manière synchrone, c'est-à-dire que chaque nœud échange des messages au même moment avec ses nœuds voisins. Contrairement à une implémentation asynchrone qui a l'avantage d'occuper les nœuds de calcul sans attente, l'échange synchrone peut potentiellement perdre du temps de calcul lorsque les temps de calcul des fonctions objectifs sont fortement hétérogènes. Par contre, il permet de garantir l'adéquation entre l'information communiquée et l'état de la recherche. En effet, par la synchronisation, les nœuds voisins sont dans des états de recherches proches avec des solutions candidates proches, et l'information communiquée par un nœud est donc aussi potentiellement pertinente pour un nœud voisin qui reçoit l'information.

L'algorithme DAMS est présenté par l'algorithme 2. L'initialisation concerne à la fois la population, qui est peut-être initialisée avec des solutions aléatoires de l'espace de recherche, mais aussi la première métaheuristique sélectionnée pour l'exécution locale et la valeur par défaut de la récompense correspondante.

À chaque itération, trois étapes sont distinguées : (i) la distribution qui permet le partage d'information entre nœuds, (ii) la sélection de métaheuristiques, et (iii) le niveau local d'exécution de métaheuristique. Comme pour les modèles en îles classiques, à l'étape (i) de distribution, l'information entre des nœuds voisins est échangée. Seulement, en plus de la migration des so-

Algorithm 2 Algorithme DAMS pour chaque nœud de calcul

Inputs : Un portefeuille de métaheuristiques $\mathcal{M}$
$r \leftarrow$ INIT_REWARD()
$M \leftarrow$ INIT_META($\mathcal{M}$)
$P \leftarrow$ INIT_POP()
repeat
    /* Niveau distribué : migration et partage de l'information */
    Envoyer $\mathsf{Msg}(r, M, P)$ à chaque voisin
    $\mathcal{P} \leftarrow \{\}$; $\mathcal{S} \leftarrow \{\}$
    for Chaque voisin w **do**
        Recevoir $\mathsf{Msg}(r', M', P')$ depuis w
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{P'\}$
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(r', M')\}$
    end for
    $P \leftarrow$ UPDATE_POPULATION($P, \mathcal{P}$)
    /* Niveau stratégie de sélection des métaheuristiques */
    $M \leftarrow$ SELECT_META($\mathcal{M}, (r, M), \mathcal{S}$)
    /* Niveau atomique : appliquer la métaheuristique et calculer une récompense*/
    $P_{new} \leftarrow$ APPLY(M, P)
    $r \leftarrow$ REWARD(P, P_{new})
    $P \leftarrow P_{new}$
until La condition d'arrêt est satisfaite

lutions candidates, les récompenses locales calculées sur chaque nœud par l'exécution de la métaheuristique sont également échangées. Ainsi, chaque nœud envoie un triplet constitué de la récompense, de la métaheuristique exécutée et de la population. Ensuite, chaque nœud reçoit les informations des nœuds voisins. À l'étape (ii) de la sélection de métaheuristiques, une métaheuristique est sélectionnée parmi celles du portefeuille selon les précédentes récompenses collectées. (iii) À la dernière étape de l'algorithme, appelé aussi niveau atomique, une itération de la métaheuristique sélectionnée est appliquée et la récompense correspondante est calculée. Encore une fois, une itération de la métaheuristique peut se réduire à un opérateur local, ou un paramètre particulier de cet opérateur, ou encore à un algorithme plus sophistiqué comme la recherche locale itérée – *Iterated Local Search* –.

Afin que l'algorithme DAMS s'arrête, un signal d'arrêt est propagé à tous les nœuds de calcul. Tous les critères d'arrêt sont possibles. Toutefois, deux critères sont généralement utilisés : l'algorithme s'arrête lorsqu'un niveau de qualité suffisant de solution est trouvé ou lorsqu'un nombre d'itérations défini est atteint.

Les performances du système distribué DAMS reposent en partie sur la stratégie de sélection de métaheuristiques. En effet, la sélection locale sur

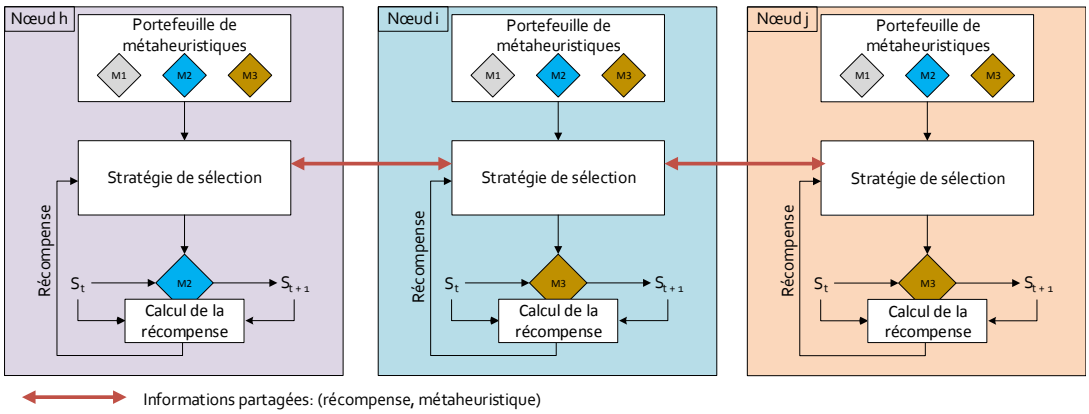


FIGURE 9 – Les entrées-sorties de la fonction « stratégie de sélection » collective dans un environnement distribué. Elle dispose d'un portefeuille de métaheuristiques (paramètres), de la récompense de la métaheuristique exécutée à l'itération précédente entre les solutions s_t et s_{t+1} et des informations (récompenses) relatives aux exécutions des métaheuristiques (flèches brunes) données par les nœuds voisins. En sortie, une métaheuristique/paramètre à appliquer à l'itération suivante est sélectionnée.

chaque nœud de calcul doit conduire à une sélection optimale de l'ensemble des métaheuristiques globalement au niveau du système. Afin de distinguer les différentes stratégies de sélection possibles de métaheuristiques, la figure 9 illustre le fonctionnement détaillé de la stratégie de sélection dans ce contexte distribué. La stratégie de sélection peut prendre en compte à la fois les informations fournies par la métaheuristique exécutée sur le nœud propre à travers la récompense calculée localement, mais aussi l'information fournie par les nœuds voisins. Ainsi, on différencie les stratégies de sélection indépendante qui n'utilisent que les récompenses calculées localement sur le nœud, des stratégies de sélection collective qui utilisent aussi les informations fournies par les nœuds voisins. Quelle que soit la stratégie de sélection, individuelle ou collective, la migration des solutions s'opère, par contre seule dans le cas collectif les récompenses sont partagées.

Comme chaque nœud est indépendant, l'algorithme DAMS est résilient aux problématiques de panne. Si un nœud ne répond plus le système peut continuer à résoudre le problème. Par contre une des difficultés à régler est la prise de décision locale de chaque stratégie de sélection pour que l'ensemble soit globalement efficace.

Afin d'illustrer le fonctionnement de l'algorithme, la figure 10 présente à différents états de la recherche, l'opérateur choisi par la stratégie de sélection Select-Best-Mutate. Un portefeuille de deux opérateurs est considéré (mutation de 1 bit et mutation de 4 bits). La stratégie de sélection à chaque itération doit choisir l'opérateur considéré le plus performant. Dans cet exemple, la

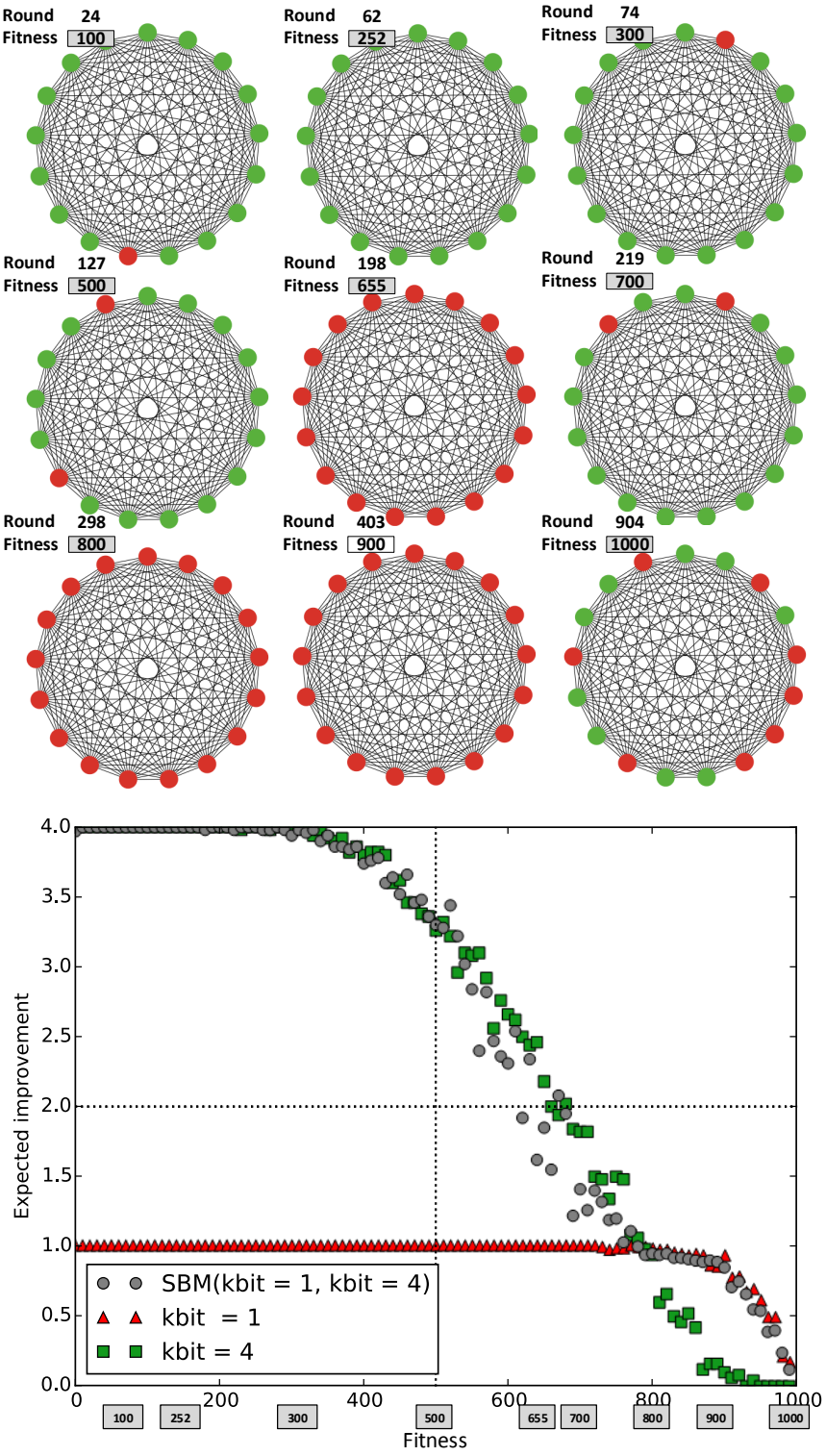


FIGURE 10 – Illustration de l'exécution de la stratégie DAMS-SBM pour One-Max (N = 1000) et un portefeuille de 2 opérateurs mutation de 1 bit (kbit = 1) et mutation de 4 bits (kbit = 4).

stratégie de sélection ne se trompe quasiment pas sur l'opérateur à choisir. Dans une stratégie de sélection, un facteur de diversité est appliqué, toutefois la proportion de nœuds de calcul réalisant l'exploration est faible. La majorité des nœuds de calcul est dédiée à l'opérateur considéré comme ayant les meilleures performances.

Dans les sous-sections suivantes, nous présentons les stratégies de sélection indépendante et collective qui sont analysées expérimentalement sur le problème OneMax et les problèmes NK à la prochaine section.

3.1.2 *Stratégie de sélection indépendante*

Une stratégie de sélection indépendante utilise uniquement les informations disponibles calculées par le nœud local. Les nœuds de calcul sont indépendants concernant le choix de l'opérateur par rapport aux autres nœuds de calcul. Les stratégies de sélection n'échangent pas d'information entre les nœuds de calcul. Seule la migration des meilleures solutions est réalisée. Comme dans le cas de la sélection séquentielle (voir section 2.5), les stratégies de sélection indépendantes utilisent l'historique des récompenses pour sélectionner un opérateur.

Les versions indépendantes (l'indice i est utilisé pour signifier indépendant) notées ici ϵ-Greedy$_i$ et AP$_i$ sont identiques aux versions séquentielles. Pour rappel, la stratégie ϵ-greedy sélectionne la meilleure métaheuristique estimée sur la fenêtre glissante des W dernières récompenses avec un taux $1-\epsilon$ et une métaheuristique aléatoirement avec un taux ϵ. La stratégie Adaptive Pursuit (AP$_i$) utilise par contre le principe de pondération exponentielle des récompenses passées et sélectionne proportionnellement à la qualité des métaheuristiques.

Nous notons également UCB$_i$-W la stratégie d'Upper Confidence Bound qui utilise une fenêtre glissante des W plus récentes récompenses pour calculer la récompense estimée relative à chaque métaheuristique (voir section 2.5.2). UCB$_i$-PH est la variante de UCB qui utilise, en plus de la fenêtre glissante, le test de Page-Hinkley ; le test Page-Hinkley réinitialise l'estimation moyenne quand la récompense de la meilleure métaheuristique diminue brutalement.

Nous proposons une nouvelle stratégie indépendante inspirée de la stratégie Select Best Mutate [DV11]. La stratégie Select Best Mutate indépendante, notée SBM$_i$, sélectionne avec le taux $1-p$ la métaheuristique ayant la meilleure récompense dans une fenêtre glissante de taille W, et sélectionne aléatoirement de manière uniforme une métaheuristique avec le taux p. La stratégie SBM$_i$ est équivalente à une stratégie ϵ-Greedy où l'estimation de

la récompense serait calculée par le maximum et non pas par la moyenne empirique.

$$\begin{cases} \text{Sélectionne le bras } i \text{ avec la meilleure} \\ \quad \text{récompense dans la fenêtre } W \\ \text{Selectionne aléatoirement un bras } i \end{cases} \quad \begin{array}{l} \textbf{Si } \text{rand}(0,1) < P_{mut} \\ \textbf{Sinon} \end{array} \tag{5}$$

3.1.3 *Stratégie de sélection collective*

Chacune des stratégies de sélection indépendantes mentionnées à la sous-section 3.1.2 précédente est utilisée pour définir de nouvelles stratégies de sélection collective qui prennent en compte les informations de récompense communiquées par les nœuds voisins.

La stratégie Select-Best-and-Mutate (SBM_c) est proposée par [DV11]. Avec un taux (probabilité) $1 - p_{mut}$, SBM_c sélectionne une métaheuristique ayant la meilleure récompense à l'itération précédente parmi toutes les récompenses voisines et en incluant le nœud courant, et avec un taux p_{mu}, SBM_c sélectionne une métaheuristique aléatoirement de manière équiprobable. La stratégie SBM_c a une composante d'intensification qui sélectionne la métaheuristique la mieux récompensée lors de l'itération précédente parmi les métaheuristiques voisines, et une composante de diversification qui permet d'explorer de nouvelles métaheuristiques sélectionnées au hasard.

Nous proposons la stratégie Adaptative Poursuit collective, notée AP_c, où les récompenses communiquées par l'ensemble des nœuds voisins sont itérativement utilisées pour mettre à jour l'estimation de la moyenne $\hat{q}_i$ de chaque métaheuristique i. Par contre, la probabilité p_i de sélectionner la métaheuristique i est mise à jour lorsque toutes les récompenses voisines sont prises en compte.

Les stratégies de type UCB peuvent être aussi étendues à des versions de type sélection collectives. La stratégie Upper Confidence Bound collective, UCB_c, estime la moyenne empirique $\hat{r}_i$ et le nombre d'utilisations de chaque métaheuristique $n_{i,t}$ avec les récompenses communiquées par les nœuds voisins. Notons que dans ce cas, l'ordre de communication des nœuds voisins n'a pas d'importance. De même que dans la version de sélection indépendante, une fenêtre glissante avec les W dernières récompenses connues en prenant en compte les récompenses voisines permet de définir la stratégie UCB collective avec fenêtre UCB_c-W. Si de plus, le test Page-Hinkley est utilisé comme dans la version indépendante, on peut définir la version UCB_c avec redémarrage Page-Hinkley noté UCB_c-PH.

La stratégie de sélection ϵ-Greedy collective prend en compte l'ensemble des récompenses passées et l'ensemble des récompenses reçues des voisins dans le calcul de la moyenne empirique. Ensuite, la sélection de l'opérateur dépend de la même façon que la version indépendante du paramètre ϵ.

3.2 EXPÉRIMENTATIONS : LES STRATÉGIES DE SÉLECTION COLLECTIVE ET INDÉPENDANTE

3.2.1 *Protocole expérimental*

Dans dans cette section, nous étudions les stratégies de sélection adaptatives présentées auparavant à l'aide du problème OneMax et de la famille de problèmes NK. Le problème OneMax est utilisé dans un cadre similaire dans différents travaux comme [TC02] [DV11] [Fia+10a] [Can+12] sur la sélection adaptative d'opérateurs. Le portefeuille étudié ici contient quatre variantes d'algorithme $(1+\lambda)$-ES qui diffèrent par les opérateurs. Depuis une solution parente, l'algorithme produit λ solutions selon un opérateur de mutation stochastique spécifique et sélectionne la meilleure solution pour l'itération suivante. Pour résoudre le problème OneMax, le portefeuille utilisé est composé de quatre opérateurs de mutation : trois opérateur qui mute respectivement exactement 1, 3 et 5 bits, et un opérateur « bit-flip » qui mute uniformément chaque bit avec un taux $1/N$, où $N = 1000$ est la taille de la chaîne binaire. Pour le problème NK, nous utilisons le même algorithme avec des opérateurs différents. Ce sont des opérateurs bit-flip mutant uniformément chaque bit avec un taux c de $\frac{1}{N}, \frac{2}{N}, \frac{4}{N}, \frac{8}{N}$ et $\frac{16}{N}$. À noter que dans ce cas le portefeuille est composé de 5 opérateurs.

Nous utilisons un mécanisme de migration élitiste. Chaque nœud (île) envoie sa solution courante, donc la meilleure trouvée par le nœud, à ses nœuds voisins. Alors, la meilleure solution reçue remplace la solution courante du nœud si elle est meilleure. L'algorithme DAMS s'arrête quand le maximum global est trouvé par un nœud du système distribué ou quand le nombre d'itérations excède $T_{limit} = 5.10^4$ itérations. Pour un système parallèle synchrone, la performance est mesurée ici en nombre d'itérations de l'algorithme d'optimisation. Pendant une itération, l'ensemble des noeuds de calcul du système exécute un opérateur et calcule λ évaluations de la fonction d'optimisation. Le nombre d'évaluations total est le produit d'un nombre d'itérations par la valeur de λ et du nombre de nombres de noeuds de calcul. Pour chaque stratégie et topologie possible, 100 exécutions d'algorithme sont effectuées. La performance de l'algorithme est mesurée soit avec le nombre d'itérations pour atteindre l'optimum global, soit utilise l'expected running time (ERT). ERT est le temps moyen d'exécution pour atteindre un niveau de qualité (fitness) donné en simulant un redémarrage de l'algorithme [AH05]. Formellement, il est égal à $E_s[T] + (1-\hat{p}_s)/\hat{p}_s.T_{limit}$ où $\hat{p}_s$ est l'estimation du taux de succès (calculé empiriquement), et $E_s[T]$ est le temps moyen (calculer ici en itérations) quand le niveau de la qualité est atteint, *c.-à-d.* en cas de succès.

Nous étudions quatre topologies de réseau :

TABLE 1 – Les méta-paramètres utilisés pour choisir les meilleurs méta-paramètres pour chaque stratégie de sélection indépendante (i) et collective (c) lors de l'étude de robustesse.

Stratégie de sélection	Méta-paramètre	Signification	Plage de paramètres
SBM_i	p_{mut}	taux de mutation	$\{0.001, 0.003, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$
	W	taille de la fenêtre glissante	$\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 300, 500\}$
SBM_c	p_{mut}	taux de mutation	$\{0.001, 0.003, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$
AP_i et AP_c	α	taux d'adaptation	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
	β	taux d'apprentissage	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
	p_{min}	valeurs extrêmes min	0.1
	p_{max}	valeurs extrêmes max	1
UCB_i et UCB_c	C	exploitation/exploration	$\{0.0001, 0.05, 0.1, 0.5, 1, 10, 25, 30, 100\}$
$UCB\text{-}W_i$ et $UCB\text{-}_c$	C	exploitation/exploration	$\{0.0001, 0.05, 0.1, 0.5, 1, 10, 25, 30, 100\}$
	W	taille de la fenêtre glissante	$\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 300, 500\}$
$UCB\text{-}PH_i$ et $UCB\text{-}PH_c$	C	exploitation/exploration	$\{0.0001, 0.05, 0.1, 0.5, 1, 10, 25, 30, 100\}$
	δ	différence minimale entre deux opérateurs	$\{0.0001, 0.05, 0.1, 0.5, 0.8, 1, 2, 3, 5\}$
	γ	seuil de redémarrage	$\{0.1, 0.5, 1, 2, 3, 4, 5, 10, 15, 20\}$
$\epsilon\text{-Greedy}_i$ et $\epsilon\text{-Greedy}_c$	ϵ	composant de diversité	$\{0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
$\epsilon\text{-Greedy-W}_i$	ϵ	composant de diversité	$\{0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
	W	fenêtre glissante	$\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90\}$

— la topologie complète où chaque nœud est connecté à l'ensemble des autres nœuds ;
— la topologie aléatoire où il y a un arc entre deux nœuds avec la probabilité de 0.1 ;
— la topologie grille en deux dimensions (grille carrée) où chaque nœud est connecté à ses quatre plus proches voisins ;
— la topologie cercle où les nœuds sont connectés à deux autres nœuds afin de former un cercle.

La taille du réseau étudié est $n \in \{4, 16, 32, 64\}$. Pour avoir le même nombre d'évaluations au cours d'une itération quelques soit la taille du réseau n, le paramètre λ de l'algorithme est initialisé $(1 + \lambda)$-ES à $64/n$.

Plusieurs de méta-paramètres sont utilisés dans les différentes stratégies de sélection (cf. 1). Le choix des méta-paramètres pour UCB-PH sont basés sur [Fia+10a], pour AP la valeur des méta-paramètres p_{min} et p_{max} sont issus sur [Thi07]. De plus, deux stratégies de base sont utilisées : la stratégie aléatoire sélectionne aléatoirement de manière uniforme à chaque itération une métaheuristique, et la stratégie constante hétérogène sélectionne toujours la même métaheuristique durant l'optimisation laquelle est choisie aléatoirement initialement.

L'implémentation simule des communications parallèles et synchrones entre les nœuds. Les échanges d'informations de la stratégie de sélection et pour la migration des solutions se font à l'aide de message de type MPI [AT02]. Que ce soit dans une implémentation MPI ou une simulation des communications, deux types d'échange d'information peuvent être envisagés. Soit, le nœud i interroge un nœud voisin j afin d'obtenir l'information voulue, c'est le mode « get » ; soit, un nœud voisin j envoie les informations au nœud i afin qu'il puisse en faire un traitement, c'est le mode « push ». Dans cette implémentation, seul le modèle « get » est considéré.

3.2.2 *Comparaison des stratégies de sélection : le problème OneMax*

VUE D'ENSEMBLE DES PERFORMANCES D'un point de vue purement distribué, la première mesure intéressante est le nombre d'itérations qu'il faut pour un algorithme pour atteindre le maximum global. Le nombre d'itérations donne une idée à propos du degré de parallélisme d'un scénario idéal où le coût de la communication est supposé négligeable par rapport au coût de l'évaluation de la fonction.

Pour chaque stratégie de sélection, les paramètres les plus performants sont résumés à la figure 11 suivant un plan factoriel pour une topologie complète de 64 nœuds de calcul. La performance relative des différentes stratégies est résumée dans le tableau 2 pour les quatre topologies : complet, cercle, grille et aléatoire en fonction du nombre de nœuds de calcul et de λ.

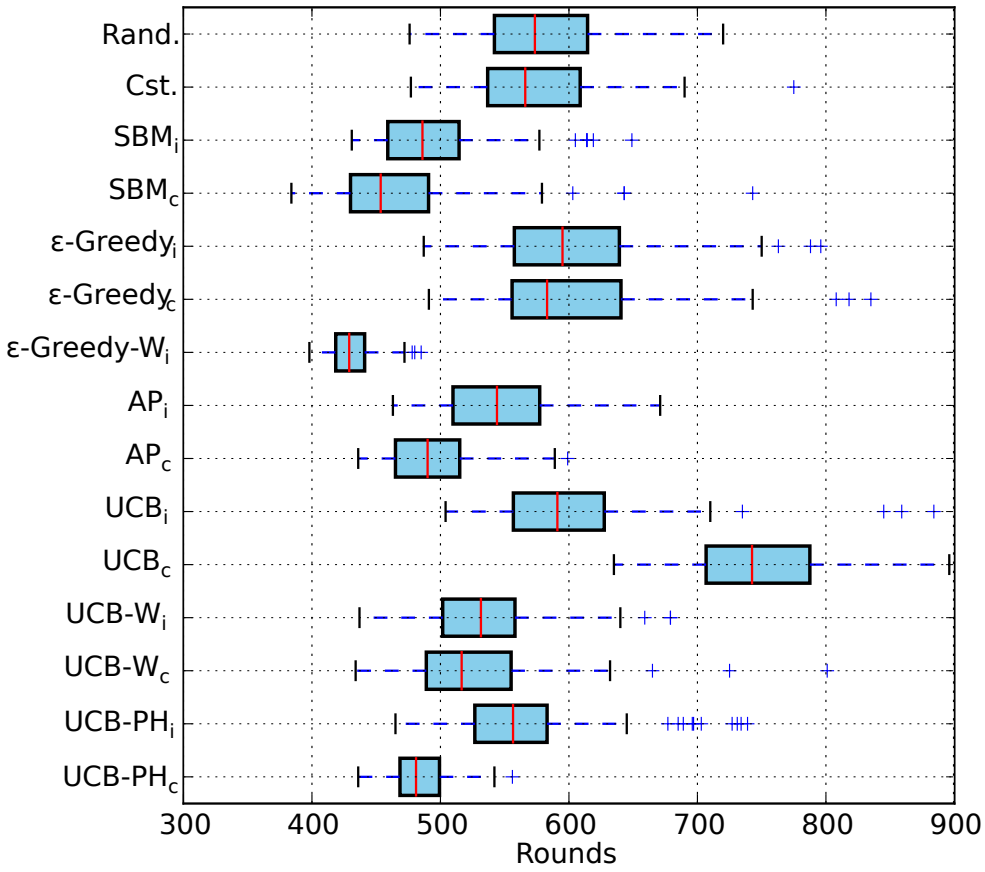
Plusieurs observations peuvent être extraites de tableau 2. Les performances des différentes stratégies de sélection peuvent être classées de façon globale indépendamment du type de topologie ou de la taille du réseau. Ce premier classement rend ainsi compte de la pertinence de la stratégie de sélection sans pour autant surévaluer les performances de tel ou tel cas particulier.

Selon ce classement global, les stratégies de sélection obtenant des performances inférieures à la stratégie de sélection aléatoire sont les stratégies ϵ-Greedy et UCB. En revanche, les stratégies de sélection indépendante SBM_i, AP_i, UCB_i-W et UCB_i-PH adaptées à un cadre collectif améliorent la performance.

Globalement, les trois stratégies de sélection pouvant être considérées comme celles ayant les meilleures performances sont dans l'ordre ϵ-Greedy-W, UCB_c et SBM_c. La stratégie faisant la plus mauvaise performance est ϵ-Greedy_i ainsi que sa variante collective.

Les deux stratégies de sélection les moins performantes considèrent toutes les récompenses pour chaque opérateur depuis le début de l'optimisation. Le facteur d'oubli et une ré-exploration n'existant pas, la méthode est en quelque sorte bloquée par les informations prises au cours des premières itérations. Cela est favorable dans un contexte stationnaire où le meilleur opérateur ne change pas au cours de l'optimisation, mais pas dans un contexte où les performances relatives des opérateurs évoluent en fonction de l'état de la recherche.

Enfin, une stratégie de sélection performante pour la topologie complète est souvent aussi performante pour d'autres topologies. L'échange d'information des récompenses associé aux opérateurs entre les nœuds de calcul a un impact positif sur la performance de l'optimisation. La stratégie de sélection ayant le plus d'information portée à sa connaissance augmente la justesse de son choix d'opérateur.



Nom	Meilleur paramètre	Itération	Rang
Aléatoire (Aléa.)	-	592 ± 10	8
Constant (Cst.)	-	601 ± 12	9
SBM$_i$	$\epsilon = 0.2$, $W = 60$	484 ± 6	2
SBM$_c$	$\epsilon = 0.01$	460 ± 6	1
ϵ-Greedy$_i$	$\epsilon = 0.90$	606 ± 10	9
ϵ-Greedy$_c$	$\epsilon = 0.90$	607 ± 10	10
ϵ-Greedy$_i$-W	$\epsilon = 0.01$, $W = 50$	435 ± 10	0
AP$_i$	$\alpha = 0.2$, $\beta = 0.2$	548 ± 10	5
AP$_c$	$\alpha = 0.2$, $\beta = 0.2$	507 ± 8	3
UCB$_i$	$C = 30$	592 ± 10	8
UCB$_c$	$C = 30$	759 ± 13	14
UCB$_i$-W	$C = 0.05$, $W = 70$	534 ± 9	5
UCB$_c$-W	$C = 0.05$, $W = 80$	529 ± 10	5
UCB$_i$-PH	$C = 0.05$, $\delta = 0.01$, $\gamma = 1$	578 ± 9	8
UCB$_c$-PH	$C = 0.05$, $\delta = 0.01$, $\gamma = 1$	497 ± 5	3

FIGURE 11 – Mesure des performances, pour les meilleures valeurs de méta-paramètres, des stratégies de sélection pour le problème One-Max N = 1000 avec portefeuille composé de 4 opérateurs de mutation qui mute exactement 1, 3, 5 bits et de l'opérateur bit-flip ($c = 1/N$) pour $\lambda = 1$ et 64 nœuds. La topologie utilisée est la topologie complète. En haut, les performances en nombre d'itérations des stratégies de sélection. En bas, les meilleurs méta-paramètres avec le nombre d'itérations et le rang associé de la stratégie.

TABLE 2 – Pour chaque topologie et taille du réseau, nombre de stratégies de sélection qui est meilleur statistiquement (selon le test de Wilcoxon-Mann-Whitney au niveau de confiance p-value$= 0,05$) qu'une autre méthode de stratégie donnée pour le problème One-Max avec $N = 1000$. La stratégie de rang 0 est la meilleure : aucune autre stratégie n'est significativement supérieure à celle envisagée.

T	n	λ	Aléa.	Cst.	SBM_i	SBM_c	ϵ-G_i	ϵ-G_c	ϵ-G_i W	AP_i	AP_c	UCB	UCB_c	UCB_i W	UCB_c W	UCB_i PH	UCB_c PH
Compl.	4	16	9	13	3	3	10	9	1	7	6	10	13	0	0	8	3
	8	8	9	10	1	1	9	9	0	7	6	9	13	1	1	8	4
	16	4	9	10	1	1	10	10	0	7	6	10	14	4	3	8	3
	32	2	8	9	1	1	10	10	0	7	3	9	14	4	3	8	3
	64	1	8	9	2	1	9	10	0	5	3	8	14	5	5	8	3
Cercle	4	16	9	14	4	3	9	9	1	7	4	9	9	0	0	8	4
	8	8	9	9	3	0	9	10	1	7	6	9	9	3	0	8	3
	16	4	10	9	3	0	13	13	1	7	6	10	10	2	2	8	2
	32	2	10	8	2	0	13	12	1	7	5	10	10	4	4	8	2
	64	1	10	8	2	1	13	13	0	7	4	10	10	4	4	8	3
Grille	4	16	9	14	4	2	9	10	2	7	6	9	10	0	0	8	4
	8	8	9	11	1	0	11	11	0	7	6	9	9	1	0	8	3
	16	4	9	9	2	0	11	10	1	7	6	10	10	3	2	8	2
	32	2	9	9	2	0	9	11	0	7	5	9	9	5	3	8	2
	64	1	10	8	2	1	11	11	0	7	4	11	11	4	4	8	3
Aléa.	4	16	9	11	3	6	9	9	0	4	4	9	9	0	0	7	3
	8	8	9	9	0	0	9	9	0	6	6	9	9	0	0	8	0
	16	4	9	9	2	0	10	9	1	6	6	9	9	2	2	8	2
	32	2	9	9	2	0	11	10	0	7	4	12	9	4	3	8	2
	64	1	9	8	2	1	10	11	0	5	4	10	10	4	4	8	2
Moyenne			9.1	9.8	2.1	1.05	10.25	10.3	0.45	6.55	5	9.55	10.55	2.5	2	7.95	2.65

COMPORTEMENT ADAPTATIF Afin de mieux comprendre les propriétés des différentes stratégies de sélection, nous étudions à la figure 12 la dynamique du choix des opérateurs au cours de l'optimisation et en fonction de la fitness. Ces expériences sont réalisées avec 64 nœuds sur une topologie complète. La fréquence reportée sur chaque graphique en fonction de la fitness correspond à la proportion de nœuds sélectionnant un opérateur donné.

La stratégie de sélection SBM_c passe par trois phases : la première phase l'opérateur 5 bits est choisi, puis le 3 bits et le 1 bit. L'utilisation de ces opérateurs est attendue dans le sens où cet enchaînement correspond la succession proche de l'optimalité : au début de la recherche, la progression est plus rapide en mutant un plus grand nombre de bits ; au fur et à mesure, le changement d'un grand de bit aléatoirement dégrade la progression moyenne, d'où l'utilisation des opérateurs de mutation 3 bits et 1 bit. Les nœuds ont tendance à sélectionner l'opérateur le plus performant – qui est l'opérateur 5 bits au début de l'optimisation –. Dès qu'un nœud a détecté que cet opérateur a réellement une bonne récompense, les informations sont communiquées aux autres nœuds et le mécanisme de sélection du meilleur opérateur oblige tous les nœuds à choisir également cet opérateur.

Les deux stratégies de sélection obtenant des performances plus faibles qu'un choix aléatoire, UCB_c et ϵ-Greedy, apparait clairement sur le choix de l'opérateur au cours des itérations. Pour UCB, à chaque itération, l'ensemble des nœuds de calcul sélectionne un opérateur différent. Par contre, dans la stratégie ϵ-$Greedy_c$ pour l'ensemble nœuds de calcul, chaque opérateur est utilisé à proportion équivalente avec une préférence pour l'opérateur de mutation 5 bits sur certaines exécutions. Pour comprendre les deux stratégies le comportement, nous rappelons que cette stratégie utilise les récompenses moyennes empiriques calculées depuis le début de l'exécution. Par conséquent, en ce qui concerne l'ensemble de l'exécution, la composante exploratoire d'UCB_i a tendance à maintenir un choix juste parmi tous les opérateurs. Étant donné que les différences dans les valeurs de la récompense entre les opérateurs ne sont pas si prononcées dans l'ensemble de l'exécution, la détection de l'opérateur pertinent qui maximise le bénéfice est difficile pour la composante d'exploitation de la stratégie UCB_i. Ceci explique pourquoi les mécanismes de redémarrage et les mécanismes de la fenêtre glissante agissent bien mieux lorsqu'ils sont utilisés dans la variante de base d'UCB_i. Cela explique également pourquoi l'incorporation de l'information des voisins dégrade les performances pour UCB_i. Plus un nœud obtient d'informations des voisins, plus il a tendance à réduire l'incertitude, c.-à-d. la borne supérieure, de la composante d'exploration, et plus la sélection a tendance à favoriser la composante exploitation.

Les stratégies UCB_c-PH, UCB_c avec fenêtre et AP_c ont un mécanisme de sélection sur chaque nœud et sont moins élitistes par rapport à SBM_c car l'historique récent des récompenses est considéré. Les trois stratégies sont

en mesure de faire face à un compromis raisonnable entre l'exploitation et l'exploration lors du choix de l'opérateur et de tirer bénéfice de l'information des nœuds voisins.

Les mécanismes d'exploration sont essentiels pour les stratégies basées sur UCB. Ces mécanismes sont de deux types : fenêtre glissante et redémarrage avec le test Page-Hinkley. Dans le contexte séquentiel, les paramètres relatifs à ces mécanismes contrôlent le degré d'exploration temporelle de la stratégie. Sans ces mécanismes, les changements du meilleur opérateur durant l'optimisation ne pourraient pas être pris en compte par la stratégie UCB. Le contexte distribué apporte une quantité d'information supplémentaire : à chaque itération, l'ensemble des noeuds voisins met à disposition la récompense associée à l'opérateur sélectionné sur le noeud. L'augmentation de l'information instantanée accroît naturellement la tendance à l'exploitation des stratégies de sélection. Afin de pouvoir s'adapter au changement, les mécanismes d'exploration doivent prendre en compte ce surplus d'information spatial.

ROBUSTESSE AUX PARAMÈTRES Nous avons analysé le comportement des différentes stratégies pour une configuration de méta-paramètres fixés. En fait, nous pouvons nous demander quel est l'impact des valeurs de méta-paramètres utilisés pour chaque stratégie, c.-à-d. la robustesse des stratégies adaptatives de sélection au choix des méta-paramètres. Ceci est illustré dans la figure 13 où nous donnons des exemples représentatifs sur la sensibilité des stratégies AP, ϵ-Greedy, SBM, AP, UCB, UCB-W et UCB-PH avec différentes valeurs des méta-paramètres, tant dans le cas d'une stratégie indépendante que collective. Pour l'ensemble de l'étude de la robustesse des paramètres, le nombre total d'exécutions du programme réaliser est de 3×10^5.

Pour la stratégie de sélection SBM, il est conseillé d'utiliser une petite valeur p que ce soit dans le cadre individuel ou collectif. Pour stratégie SBM_i la différence de performance entre des mauvaises et bonnes valeurs de p est d'environs 150 itérations. La valeur de la fenêtre W est certainement moins sensible que le paramètre p si les valeurs extrêmes ne sont pas utilisées.

Les paramètres de la stratégie AP sont robustes dans une large gamme de valeurs avec l'exception du taux d'adaptation $\alpha = 1$ qui doit être évité, car il favorise une forte convergence dans l'estimation de la récompense.

Le paramètre C réglant le compromis entre exploration et exploitation joue un rôle important dans la stratégie UCB et ses variantes. Il est surtout sensible à la valeur de la récompense. Le paramètre C normalise la récompense pour avoir une bonne balance exploration/exploitation. Pour UCB, la valeur C ne semble ne pas avoir d'impact et ne peut pas aider à obtenir des résultats meilleurs. Par contre, une faible valeur pour C pour les stratégies UCB-PH, et UCB-W semble donner un bon compromis.

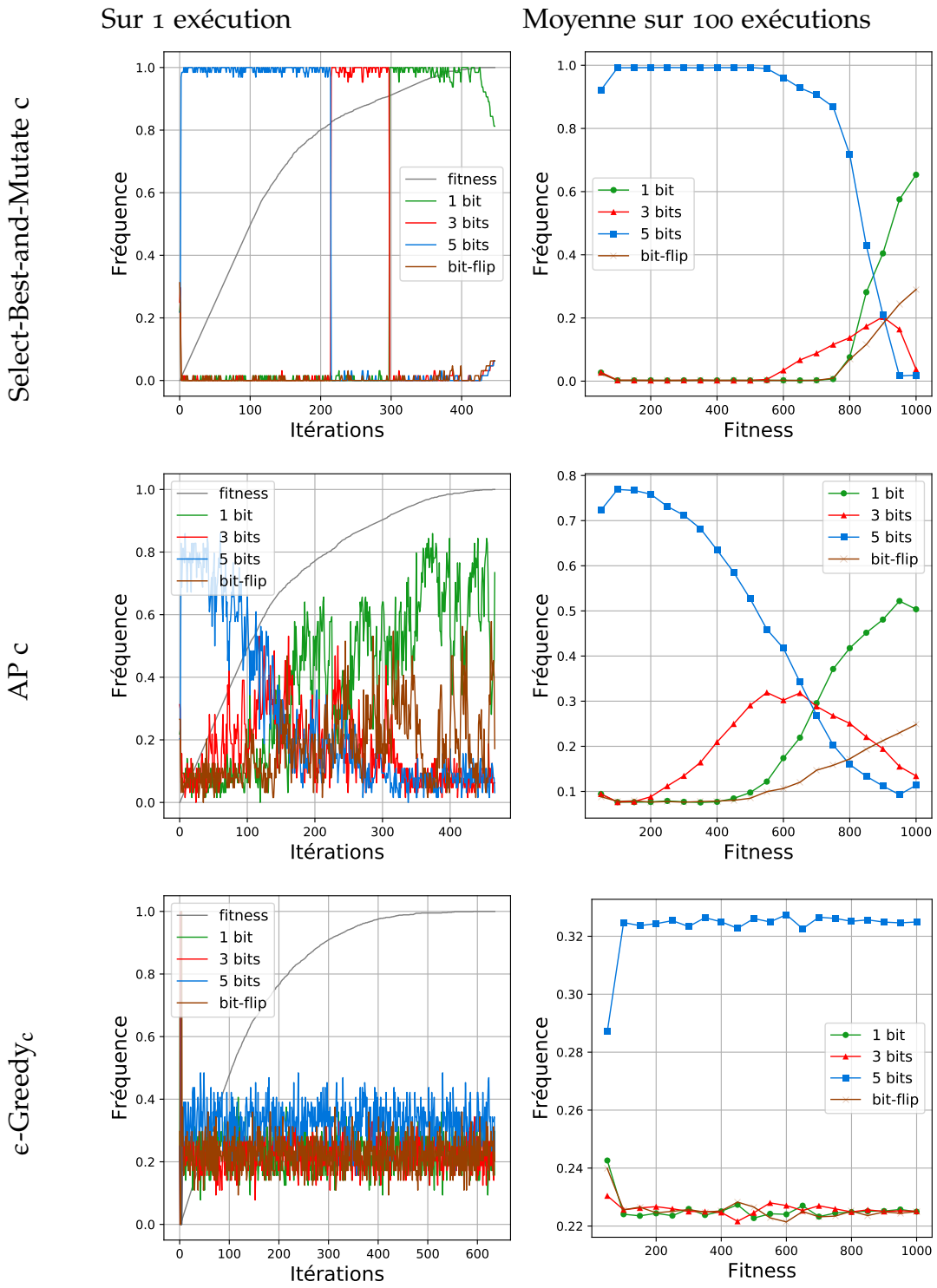


FIGURE 12 – Fréquence moyenne pour chaque opérateur pour une topologie complète avec 64 nœuds.

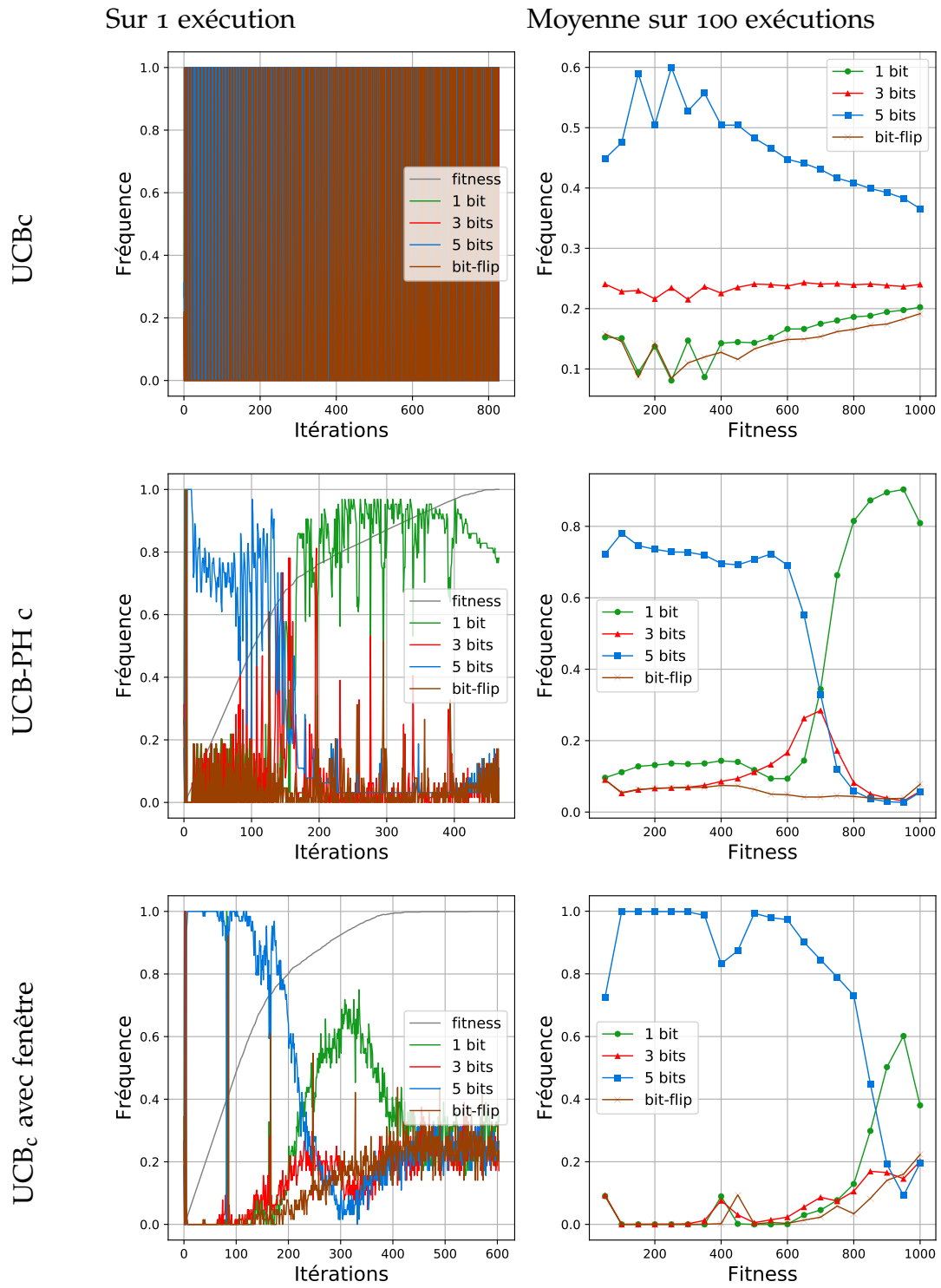


FIGURE 12 – Fréquence moyenne pour chaque métaheuristique pour le graphe complet avec 64 nœuds.

Pour la stratégie UCB-PH, le paramètre γ semble n'avoir qu'un faible impact hormis pour les valeurs extrêmes proches de 0. Par contre, les performances sont sensibles à la valeur de δ. Nous préférerons une valeur faible. Pour la stratégie UCB-W, la valeur de la taille de la fenêtre est très robuste. Il faut seulement éviter les valeurs extrêmes comme 0 ou trop grandes par rapport au nombre d'itérations. La fenêtre joue le rôle de facteur d'oubli.

L'influence de la valeur des méta-paramètres est variable selon les stratégies de sélection. De cette étude, nous pouvons recommander les valeurs de méta-paramètres suivants qui sont suffisamment performants et robustes : pour la stratégie ϵ-Greedy, la valeur du paramètre ϵ peut être comprise dans $[0.01, 0.1]$; pour la stratégie UCB, le paramètre C dans $[0.001, 0.1]$; pour la stratégie UCB avec le test de Page-Hinkley, le paramètre δ dans $[0.01, 0.1]$ et $\gamma = 1$, pour la taille de fenêtre glissante W dans $[70; 700]$; et enfin pour la stratégie AP, le paramètre α et β peuvent avoir une même valeur proche de 0.2 convient.

PARALLÉLISME Dans les discussions précédentes, nous avons analysé le comportement des stratégies indépendamment des différentes topologies. Par exemple, les résultats de la table 2 ne nous permettent pas d'apprécier l'impact des différentes topologies sur la performance de chaque stratégie. La figure 14 donne la performance relative de SBM_c, AP_c, ϵ-Greedy, UCB_c-W et UCB-PH_c dans différentes configurations du réseau de nœuds de calcul. Il est important de rappeler que le nombre d'évaluations à chaque itération et pour toutes les configurations considérées est identique ($\lambda = 64/n$). Cela signifie que le nombre d'évaluations pour l'ensemble des configurations envisagées est du même facteur que le nombre d'itérations de la figure 14.

Le nombre de messages échangés est un indicateur important de l'efficacité d'un système de calcul parallèle. Il est possible d'obtenir différents compromis entre le nombre de messages échangés et le temps de calcul sur chaque nœud lors du déploiement effectif des stratégies adaptatives dans un contexte réellement distribué. En effet, le nombre de messages échangés est exactement le nombre d'itérations de l'algorithme multiplié par le nombre d'arêtes utilisées dans la topologie considérée. Dans le cas de la topologie complète (et respectivement pour la topologie cercle, la topologie grille et la topologie aléatoire), le nombre d'arêtes est $n(n-1)/2$ (et respectivement $n-1$, $O(n)$, $O(p.n^2)$) où n est le nombre de nœuds de calcul.

Nous constatons que le nombre d'itérations reste stable pour la topologie complète et aléatoire, quelque soit le nombre de nœuds, la topologie complète étant légèrement meilleure. Cependant, le nombre d'itérations augmente fortement avec le nombre de nœuds pour la topologie en cercle et en grille que nous attribuons à l'augmentation du diamètre du graphe. Globalement, bien que l'augmentation du nombre d'itérations pour les topologies cercle et grille soit au plus d'un facteur 2, le nombre de messages nécessaires

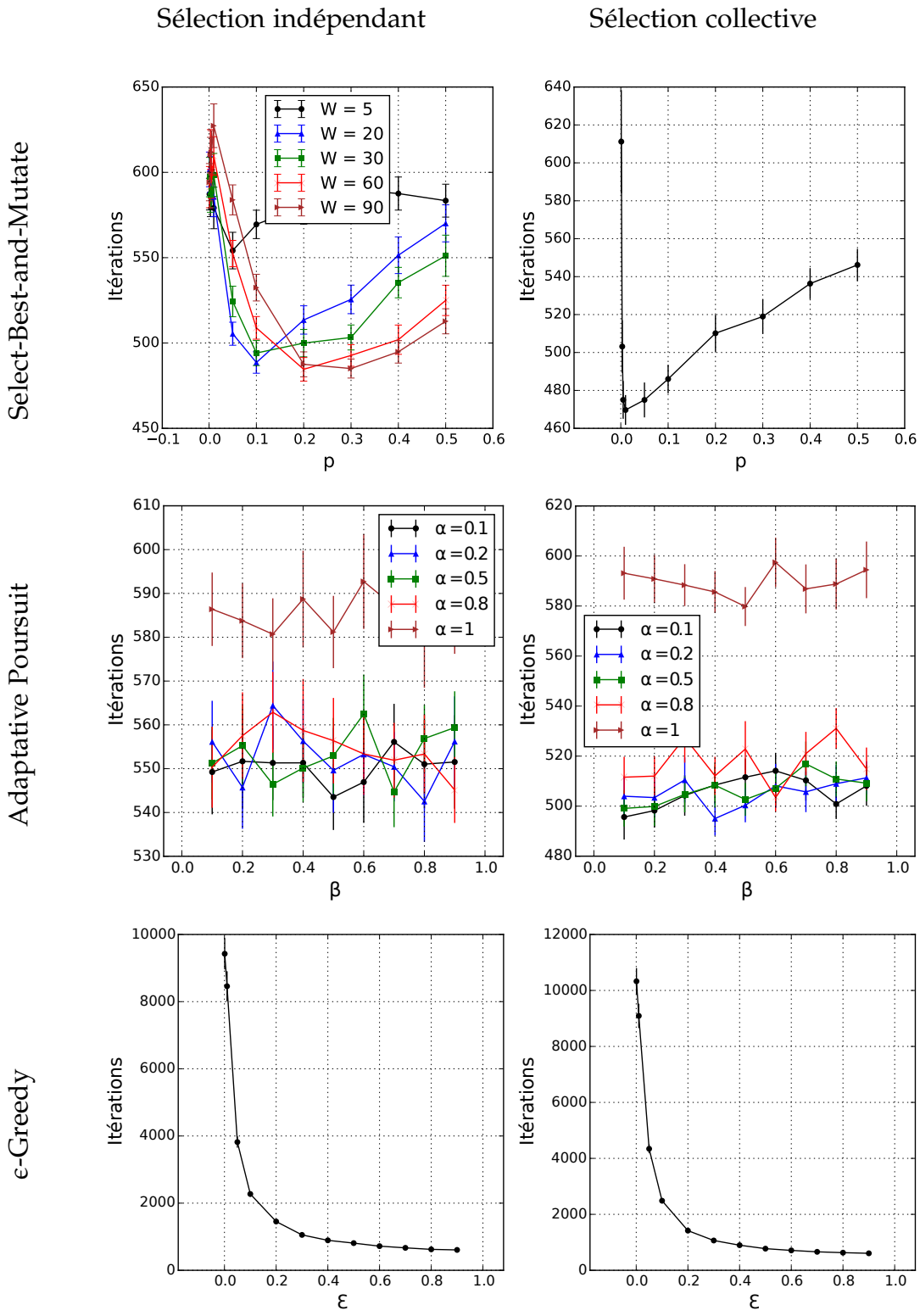


FIGURE 13 – *Robustesse des méta-paramètres* Nombre moyen d'itérations pour atteindre le maximum du problème OneMax N = 1000 avec une topologie complète et $n = 64$ en fonction des méta-paramètres des différentes stratégies de sélection. Pour UCB avec les trois variantes : SBM, AP et ϵ-Greedy.

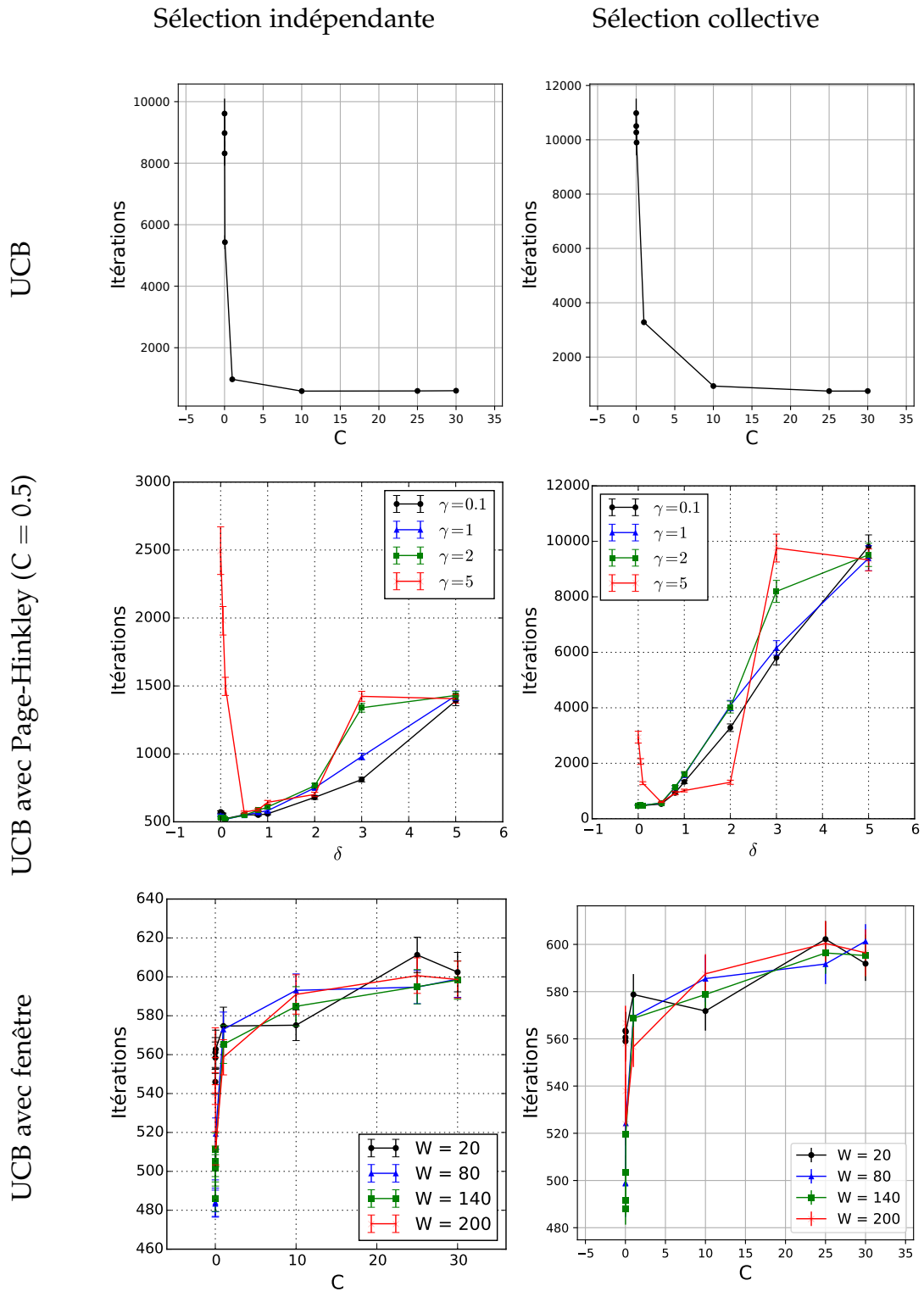


FIGURE 13 – *Robustesse des méta-paramètres* Nombre moyen d'itérations pour atteindre le maximum du problème OneMax N = 1000 avec une topologie complète et n = 64 en fonction des méta-paramètres des différentes stratégies de sélection. Pour UCB avec les trois variantes : UCB, UCB-PH et UCB avec fenêtre glissante.

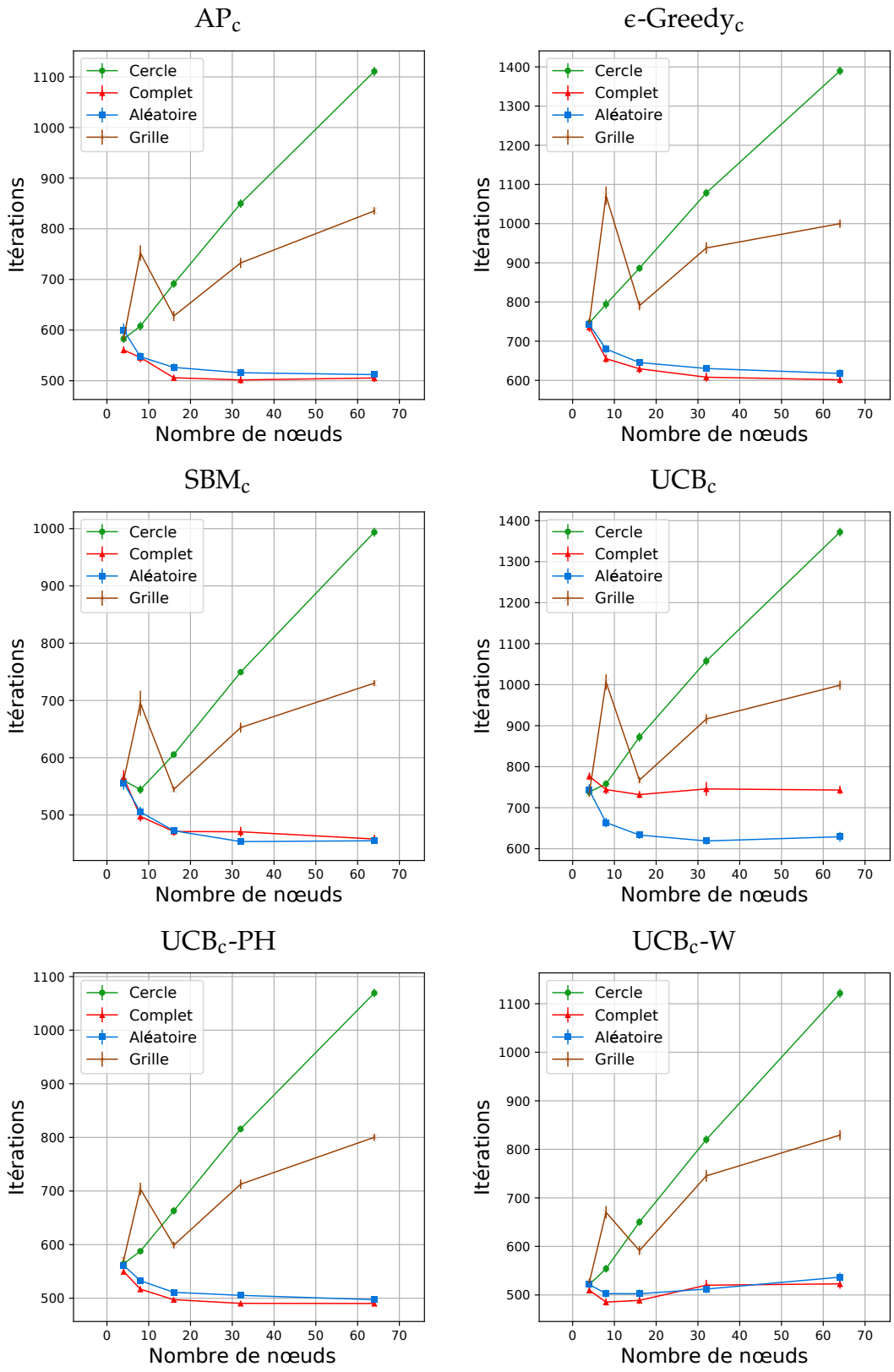


FIGURE 14 – Nombre moyen d'itérations pour trouver le maximum pour le problème OneMax N = 1000 avec 64 nœuds en fonction du nombre de nœuds. Avec les meilleurs méta-paramètres. La topologie aléatoire est engendrée avec p = 0.5 (taux de liens entre les nœuds est de 50%).

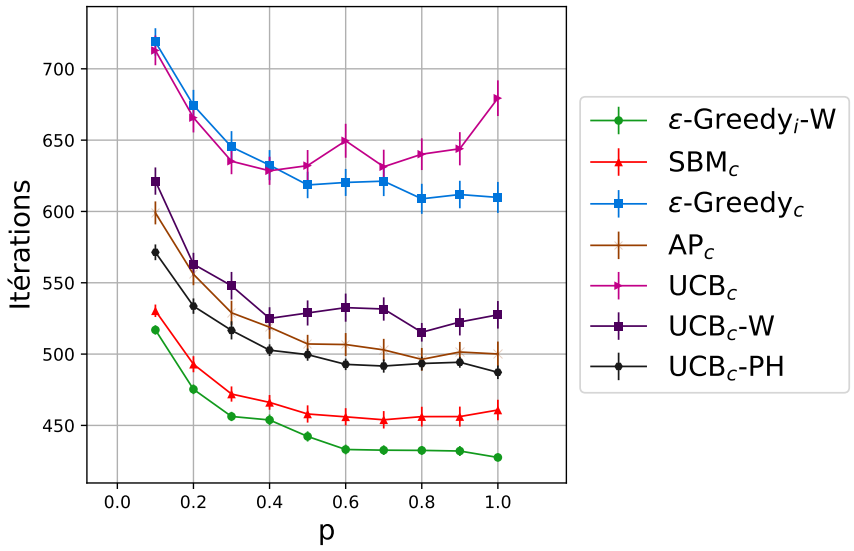


FIGURE 15 – Nombre moyen d'itérations pour trouver la fitness maximum pour le problème OneMax N = 1000 avec 64 nœuds en fonction du nombre de nœuds pour la topologie aléatoire en fonction du taux p de liens entre les nœuds.

augmente linéairement avec le nombre de nœuds. Ceci est à contraster avec la topologie complète où l'augmentation du nombre de messages est polynomiale. Par conséquent, dans un contexte parallèle réel où le coût d'échange des messages n'est pas négligeable, le meilleur choix d'une topologie aléatoire présenterait un compromis plus favorable en termes du nombre d'itérations par rapport au nombre total de messages échangés.

Pour la topologie aléatoire, il est possible de faire varier le taux de lien entre les nœuds directement proportionnel au nombre de messages échangés. Initialement fixé arbitrairement à $p = 0.5$, nous avons étudié l'impact de différents taux comme présenté à la figure 15. Nous observons qu'à partir de 50% de connexions entre les nœuds, le temps d'optimisation est relativement stable pour l'ensemble des stratégies de sélection sauf pour UCB_c. UCB_c est sensible au surplus d'information. Dans ce cas, tous les nœuds envoient de l'information, ce qui entraîne une surcharge d'information, certaines datant des premières itérations. Pour les autres stratégies, il est donc possible en fonction des caractéristiques de l'environnement de calcul parallèle de choisir un taux de connexion réduisant le temps de calcul total.

3.2.3 *Comparaison des stratégies de sélection : le problème de paysages NK*

L'intérêt pour les paysages NK (voir section 2.2.2) est double. Cette famille de problèmes est un modèle de problème d'optimisation pseudo-booléen avec différents degrés de non-linéarité. De plus, selon le taux de mutation,

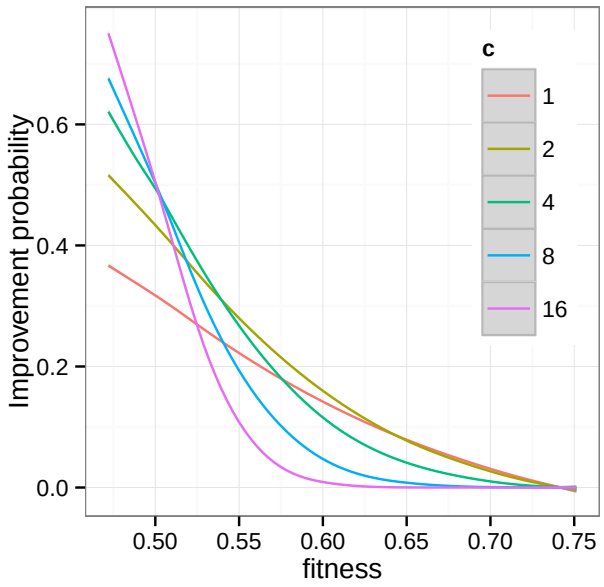


FIGURE 16 – Probabilité d'amélioration empirique en fonction de la qualité de la solution (fitness) pour le paysage NK avec $N = 1000$ et $K = 4$.

les performances des opérateurs sont différentes et peuvent fournir un cadre pour l'étude de sélection adaptative. Pour illustrer ce fait, la figure 16 donne la probabilité d'amélioration d'une solution en fonction de sa qualité (fitness) dans le cas des opérateurs de mutation bit-flip avec différents taux c/N.

La probabilité d'amélioration des opérateurs dépend du niveau de fitness. Par conséquent, ce type de paysages semble être particulièrement intéressant à étudier dans le cadre de DAMS. De plus, la difficulté d'optimisation est réglable par le paramètre K qui règle la non-linéarité du problème : plus K est élevé, plus il y a de liens épistatiques entre les variables, plus le problème est non-linéaire et difficile à résoudre [KK93]. Quand $K = 0$, le problème est linéaire et la difficulté correspond à celle du problème OneMax. Comme précédemment, nous effectuons les mêmes expériences tout en considérant les différents paysages NK avec $N = 1000$ et $K \in \{1, 4, 8, 16\}$.

Le portefeuille de métaheuristiques est composé de cinq algorithmes $(1+1)$-ES utilisant cinq opérateurs bit-flip de taux c/N de mutation par bit différents avec $c = \{1, 2, 4, 8, 16\}$. Nous utilisons les valeurs des méta-paramètres obtenus par l'étude sur le problème OneMax (voir la figure 11). Les topologies complète, cercle, grille et aléatoire sont également étudiés. Pour la topologie aléatoire, la valeur du paramètre p du taux de connexion est fixée à 0.1. Différentes tailles de graphe sont étudiées, $n = \{8, 16, 32, 64\}$, en sachant que le nombre d'évaluations par itération est constant $\lambda \times n = 64$ quelle que soit la taille de la topologie.

TABLE 3 – Classement des différentes stratégies adaptatives selon la topologie et le nombre de nœuds de calcul pour le problème de paysage NK avec $N = 1000$ et $K = 1$. Le rang 0 correspond aux meilleures stratégies selon le classement par les médiane et le test statistique de Wilcoxon-Mann-Whitney à 5%.

K = 1																	
T	n	Unif.	Aléa.	Cst.	SBM_i	SBM_c	ϵ-G_i	ϵ-G_c	ϵ-G_i W	AP_i	AP_c	UCB_i	UCB_c	UCB_i W	UCB_c W	UCB_i PH	UCB_c PH
Compl.	8	0	4	0	4	9	4	2	2	6	2	2	4	3	2	2	2
	16	0	3	0	3	3	3	3	2	3	3	3	3	5	3	3	3
	32	15	2	0	3	0	0	3	0	0	4	3	3	3	4	3	3
	64	0	0	0	0	0	9	6	0	0	0	0	0	0	3	0	6
Cercle	8	0	5	0	2	9	2	3	1	3	4	3	3	3	3	3	3
	16	0	3	0	2	2	2	4	2	2	2	2	2	4	4	2	3
	32	0	5	0	0	0	4	4	0	5	5	2	3	3	5	5	4
	64	15	0	0	0	0	3	3	0	0	3	0	7	0	0	0	0
Grille	8	0	2	0	1	2	3	1	0	5	3	3	3	1	2	2	3
	16	0	3	0	2	3	2	2	2	5	3	3	6	3	3	3	3
	32	0	4	0	3	3	3	4	2	3	3	3	4	3	6	6	6
	64	0	0	15	0	0	0	0	0	0	6	4	5	0	0	0	0
Aléa.	8	0	4	0	3	3	3	3	2	5	4	4	7	4	4	5	4
	16	0	4	0	3	4	4	4	2	4	4	4	9	4	5	4	4
	32	0	6	0	3	5	6	8	2	3	3	6	7	5	6	6	4
	64	0	6	0	3	6	6	6	2	3	3	10	9	8	9	6	10
Moyenne		1.8	3.1	0.9	2	3.0	3.3	3.5	1.1	2.9	3.2	3.2	4.6	3	3.6	3.1	3.625

K = 4																	
T	n	Unif.	Aléa.	Cst.	SBM_i	SBM_c	ϵ-G_i	ϵ-G_c	ϵ-G_i W	AP_i	AP_c	UCB_i	UCB_c	UCB_i W	UCB_c W	UCB_i PH	UCB_c PH
Compl.	8	1	7	1	2	0	7	6	1	1	1	6	6	7	7	7	7
	16	1	7	2	0	0	7	7	0	2	0	5	10	8	7	7	7
	32	2	6	2	2	0	7	7	0	4	2	6	7	7	6	7	7
	64	3	7	3	1	1	8	7	0	2	1	8	7	8	7	4	8
Cercle	8	1	7	1	2	0	6	6	4	1	1	4	6	9	6	6	7
	16	1	7	1	1	0	7	7	1	1	1	7	7	7	8	9	7
	32	3	6	3	1	0	7	7	0	3	2	7	7	7	7	9	7
	64	2	9	2	2	1	8	7	0	2	3	6	7	7	6	7	6
Grille	8	5	7	2	0	0	7	6	0	1	2	6	7	7	7	7	8
	16	1	7	2	1	0	7	7	1	1	1	7	7	7	7	7	7
	32	3	7	5	1	1	7	7	0	1	1	7	7	8	7	7	7
	64	2	7	2	1	1	6	7	0	3	2	7	7	6	8	7	7
Aléa.	8	3	7	3	1	4	6	7	0	2	1	6	6	7	12	7	7
	16	3	7	3	0	3	7	7	0	2	2	7	7	7	7	8	7
	32	3	7	2	1	5	7	6	0	2	2	7	7	11	7	7	7
	64	4	7	4	1	4	7	7	0	2	2	8	7	7	7	7	7
Moyenne		2.3	7	2.3	1	1.2	6.9	6.7	0.4	1.8	1.5	6.5	7	7.5	7.2	7	7

K = 8																	
T	n	Unif.	Aléa.	Cst.	SBM_i	SBM_c	ϵ-G_i	ϵ-G_c	ϵ-G_i W	AP_i	AP_c	UCB_i	UCB_c	UCB_i W	UCB_c W	UCB_i PH	UCB_c PH
Compl.	8	0	7	0	3	1	7	7	4	3	2	7	7	7	7	7	7
	16	1	10	0	3	0	7	7	0	4	4	8	7	7	7	7	7
	32	0	7	0	3	3	6	7	0	5	4	8	8	7	7	8	8
	64	0	7	1	4	3	7	7	0	4	4	7	7	7	8	8	7
Cercle	8	0	7	0	3	0	7	7	3	3	3	7	7	7	7	7	7
	16	1	7	1	3	0	7	7	2	4	4	7	7	7	7	9	7
	32	0	7	0	4	0	7	7	0	4	4	7	7	9	8	7	7
	64	1	7	0	4	3	7	9	0	3	4	7	7	8	7	7	7
Grille	8	0	7	0	3	2	7	8	2	2	4	7	7	7	7	7	8
	16	0	8	0	3	0	7	7	2	4	5	7	7	7	7	7	7
	32	0	7	0	3	3	8	7	0	3	3	8	8	8	8	7	7
	64	0	7	0	4	1	8	7	0	4	4	7	7	7	9	7	7
Aléa.	8	0	7	0	3	5	7	7	0	3	3	7	7	7	7	7	7
	16	0	9	0	3	6	9	8	0	4	4	7	7	7	7	7	7
	32	1	7	1	3	6	7	7	0	4	4	7	7	7	6	7	7
	64	1	7	1	3	6	6	7	0	3	4	7	6	7	7	7	8
Moyenne		0.3	7.3	0.2	3.2	2.4	7.1	7.2	0.8	3.5	3.7	7.1	7.0	7.2	7.2	7.2	7.1

K = 16																	
T	n	Unif.	Aléa.	Cst.	SBM_i	SBM_c	ϵ-G_i	ϵ-G_c	ϵ-G_i W	AP_i	AP_c	UCB_i	UCB_c	UCB_i W	UCB_c W	UCB_i PH	UCB_c PH
Compl.	8	0	7	0	3	3	7	7	0	4	3	7	7	7	7	7	7
	16	1	7	0	4	1	7	7	1	4	4	7	7	7	7	7	7
	32	0	7	0	4	3	7	7	0	5	4	8	7	7	7	7	7
	64	0	7	0	4	3	6	6	0	4	3	7	7	7	6	7	7
Cercle	8	0	8	0	9	4	10	6	0	4	3	6	5	11	6	6	6
	16	0	7	0	3	3	7	6	0	3	4	10	8	10	6	6	8
	32	0	7	0	3	3	7	7	0	3	3	7	7	7	7	10	7
	64	0	7	0	5	3	5	6	0	3	3	6	6	7	6	5	6
Grille	8	0	8	0	3	4	7	7	1	3	3	7	7	7	7	7	7
	16	0	6	0	4	2	7	6	0	9	4	7	7	6	6	7	7
	32	1	7	0	3	5	7	7	0	3	3	10	7	7	7	7	7
	64	0	6	0	6	3	6	6	0	3	3	6	7	6	6	6	7
Aléa.	8	0	6	0	3	6	6	6	1	4	4	6	6	6	6	6	6
	16	0	9	0	3	6	6	6	1	3	4	9	6	6	6	6	6
	32	0	6	0	3	6	6	6	0	3	3	6	6	6	6	6	6
	64	0	6	0	4	6	6	6	0	3	4	6	6	9	6	10	6
Moyenne		0.1	6.9	0	4	3.8	6.6	6.3	0.2	3.8	3.4	7.1	6.6	7.2	6.3	6.8	6.6

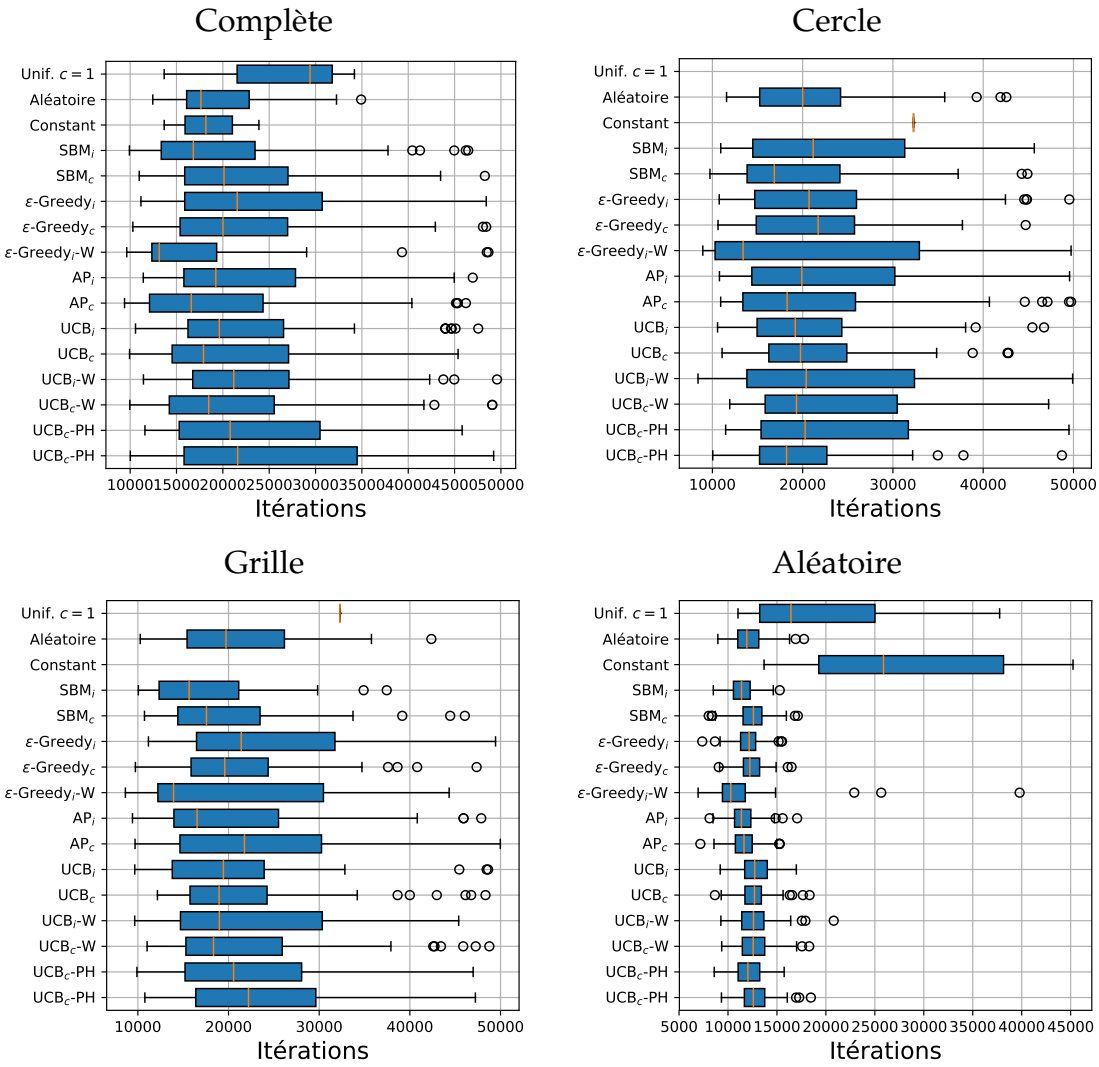


FIGURE 17 – Distributions (boites à moustache) du nombre d'itérations pour chacune des stratégies de sélection dans le cas du problème NK avec N = 1000 et K = 1. Le nombre de nœuds est 64 nœuds pour chacune des topologies suivantes : complète, cercle, grille et aléatoire.

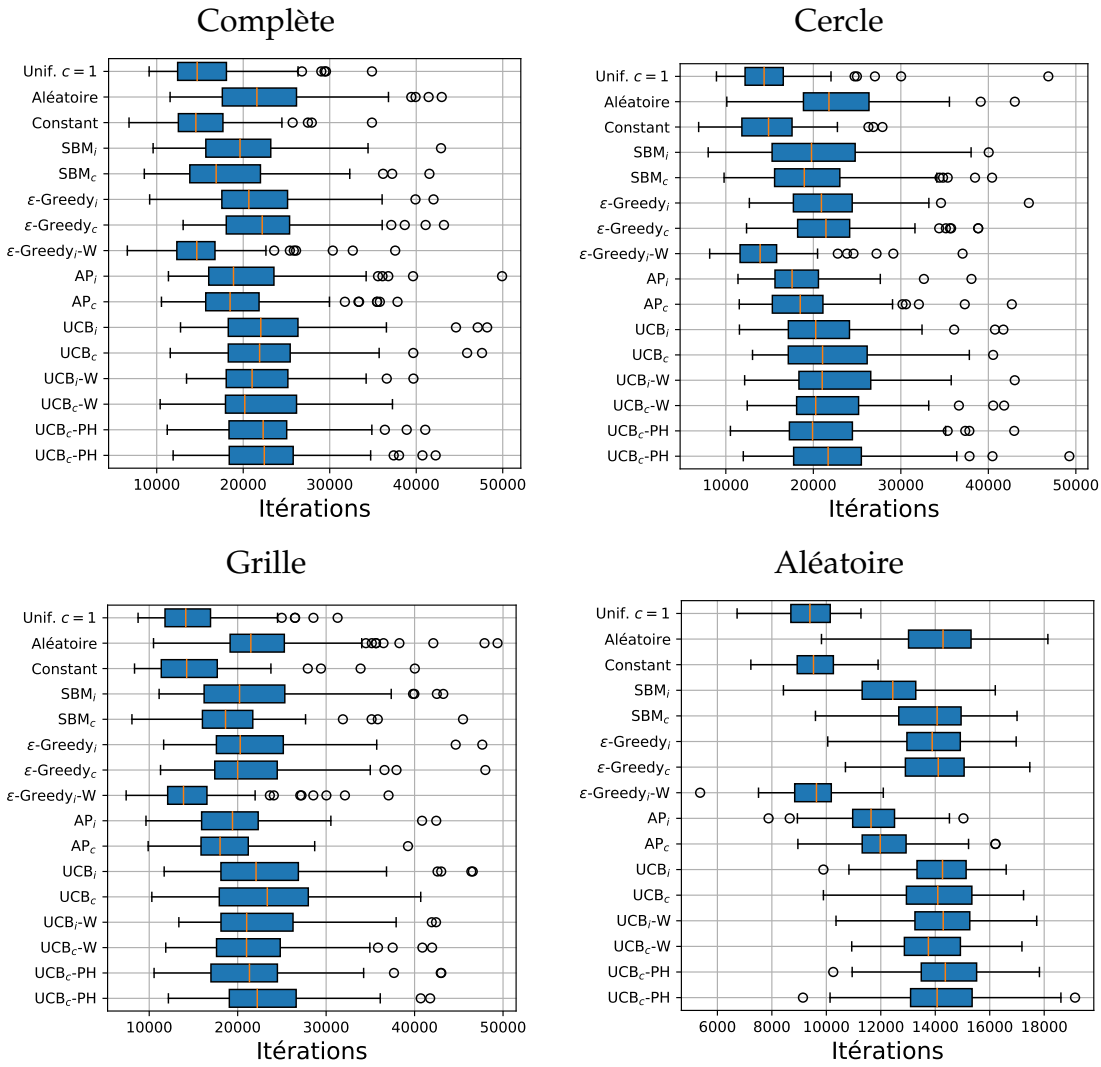


FIGURE 18 – Distributions (boites à moustache) du nombre d'itérations pour chacune des stratégies de sélection dans le cas du problème NK avec N = 1000 et K = 16. Le nombre de nœuds est 64 nœuds pour chacune des topologies suivantes : complète, cercle, grille et aléatoire.

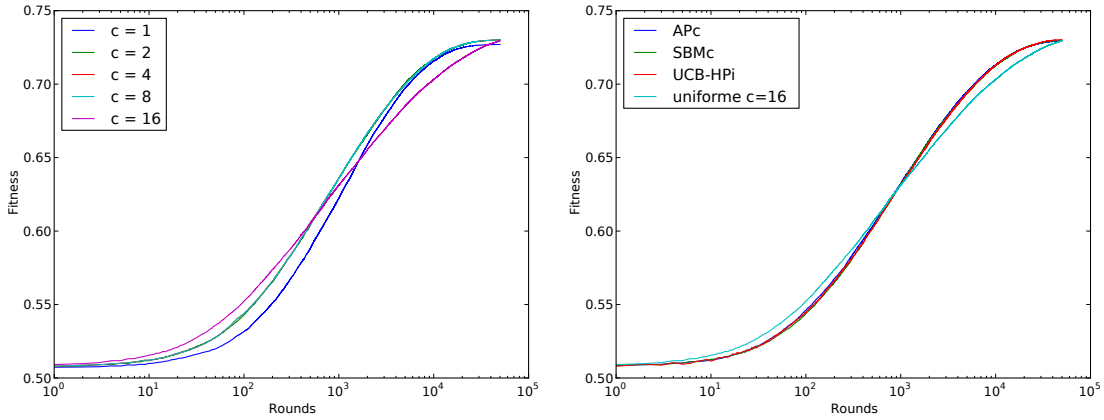


FIGURE 19 – Qualité de la solution *vs.* itération en échelle log pour le paysage NK avec $N = 1000$ et $K = 4$. À gauche : stratégies uniforme, à droite : différent stratégies. Le paysage NK avec $N = 1000$ et $K = 4$.

Dans cette étude, la médiane du nombre d'évaluations est utilisée comme mesure de performance afin de réaliser le classement des stratégies de sélection. En effet, il est à noter que la limite du nombre d'évaluation est très rarement atteinte et la médiane fournie une meilleure statistique que l'ERT. Pour chaque degré de non-linéarité K et pour chaque topologie et nombre de nœuds de calcul, le tableau 3 reporte, selon la médiane, le rang des différentes stratégies de sélection en tenant compte du test statistique de Wilcoxon-Mann-Whitney pour le problème NK. Le rang d'une stratégie est incrémenté lorsque la médiane estimée sur l'échantillon est significativement supérieur, et ainsi lorsque la stratégie est moins avantageuse. Le rang 0 correspond aux meilleures stratégies. Les mêmes stratégies de sélection adaptatives utilisées avec le problème OneMax sont étudiées. Les stratégies de sélection naïves : uniforme, aléatoire et constant hétérogène sont respectivement notée Unif., Aléa. et Cst. Pour rappel, la stratégie uniforme utilise pour l'ensemble des nœuds de calcul le même opérateur. Dans ce cas, la mutation bit-flip avec $c = 1$ surpasse les autres opérateurs sur l'ensemble de l'optimisation. La stratégie de sélection aléatoire détermine un opérateur aléatoirement pour ses nœuds et à chaque itération. La stratégie de sélection constante hétérogène choisit un opérateur aléatoirement pour l'ensemble des nœuds fixé à l'initialisation de l'optimisation.

COMPARAISON DES STRATÉGIES DE SÉLECTION Le tableau 3 indique le classement de chaque stratégie de sélection pour les différentes valeur de K. Le rang 0 correspond aux meilleures stratégies. Quelque soit le paramètre K, les stratégies de sélection de la famille UCB et AP ont de moins bonne performance qu'une stratégie uniforme avec un taux $1/n$ ou une stratégie constante. Par contre, ϵ-Greedy$_i$-W a de meilleur performance que UCB et

AP. Nous pouvons classer certaines stratégies de sélection adaptative par rapport aux autres dans certains cas. Par exemple, lorsque le degré de non-linéarité est égale à $K = 4$, la stratégie ϵ-Greedy avec fenêtre obtient le meilleur rang moyen sur l'ensemble des topologies et de la tailles, suivi des stratégies Select-Best-Mutation SBM et Adaptive Poursuit AP.

IMPACT DU PARALLÉLISME Bien que la différence de performance est faible entre une stratégie de sélection individuelle et collective (cf. tableau 3), nous constatons de façon générale que plus le problème est difficile à résoudre (c'est-à-dire avec une valeur de K élevée), plus la version collective est performante. Quand $K = 1$, seules deux stratégies de sélection adaptatives dans leur version collective (ϵ-Greedy et AP) améliorent les performances de l'optimisation. Par contre, quand $K = 16$, l'ensemble des stratégies de sélection adaptatives collectives obtiennent de meilleures performances que leur version indépendante. Cela signifie que les informations apportées par les nœuds voisins sont pertinentes et améliorent la décision finale.

IMPACT DE LA NON-LINÉARITÉ (ÉPISTASIS) DU PROBLÈME En fonction de la difficulté de l'optimisation déterminée par le paramètre K, la pertinence des stratégies de sélection adaptative varie par rapport aux stratégies de sélection naïve (uniforme, aléatoire, constant hétérogène). Nous constatons que la stratégie de sélection uniforme et constante hétérogène est la meilleure par rapport aux stratégies de sélection adaptatives lorsque la difficulté du problème est grande ($K = 16$). À mesure que la difficulté du problème s'accroît, les stratégies de sélection adaptatives ont plus de difficulté à trouver l'opérateur qui convient tout au long de l'optimisation. Mis à part, la stratégie de sélection ϵ-Greedy$_i$-W qui est relativement robuste aux changements de difficultés du problème pour $K < 16$. Néanmoins, pour $K = 16$, elle est moins performante que les stratégies uniforme et constante hétérogènes.

Les faibles performances des stratégies de sélection adaptatives par rapport aux stratégies uniformes peuvent aussi être expliquées de la manière suivante. La figure 19 donne la dynamique de recherche de la fitness en fonction du nombre d'itérations. La qualité (fitness) de la solution courante augmente très brusquement pour les paysages NK au début de la recherche lorsque les valeurs de fitness sont faibles. Par ailleurs, ces niveaux de fitness faibles correspondent aussi également à la plus grande différence de performance entre les différents opérateurs (voir figure 16). Pour les niveaux de fitness plus élevés où la recherche passe le plus de temps, la différence entre opérateurs est faible. Pour les problèmes NK, une stratégie de sélection adaptative ne peut donc gagner que peu d'itérations au début de la recherche, et peut par contre potentiellement « perdre » des itérations à explorer des opérateurs de qualité proche et difficile à discerner pour les niveaux de fitness où la recherche passe le plus de temps.

Les expériences précédentes soulèvent la question de savoir si nous devons vraiment adapter les algorithmes de recherche au moment de l'exécution (online). Les expériences sur les problèmes NK permettent de formuler les points suivants : tout d'abord, dans un scénario boîte noire, l'intervalle de fitness pendant lequel une métaheuristique est la meilleure dépend fortement de la forme du paysage de fitness. Par conséquent, apprendre ce paysage au moment de l'exécution est une alternative plausible. Deuxièmement, nous devons étudier plus attentivement le coût de la phase d'apprentissage de la stratégie de sélection en fonction du paysage considéré, et concevoir une nouvelle stratégie d'adaptation alternative qui pourrait minimiser les coûts d'apprentissage.

3.3 SYNTHÈSE

Nous avons étudié de nouvelles stratégies de sélection adaptative dans le cadre distribué à îles hétérogènes. Chaque nœud de calcul accueille une solution, un portefeuille d'opérateurs, et une stratégie de sélection. La stratégie de sélection et le portefeuille sont identiques à tous les nœuds de calcul. La migration permet d'échanger deux types d'information entre les nœuds : la solution et la récompense obtenues associées à un opérateur après application de l'évaluation.

Cette étude porte sur des variantes de stratégie de sélection telles que ϵ-Greedy, SBM, Adaptive Pursuit et UCB existantes dans le cadre séquentiel. Nous comparons les performances de celles-ci pour les problèmes OneMax et paysages NK, et comparons les stratégies de sélection indépendante et collective où les récompenses des nœuds sont ou non prises en compte. Cette étude expérimentale cherche à établir une meilleure compréhension des principaux composants des méthodes afin de mieux concevoir les stratégies de sélection adaptative distribuée.

Plusieurs éléments de compréhension peuvent être apportés. Les stratégies de sélection collective donnent de meilleures performances par rapport aux stratégies de sélection indépendante. Intuitivement, nous pouvons affirmer que plus il y a d'informations à disposition de la stratégie de sélection locale plus le processus de décision sera pertinent. Les stratégies de sélection collective partagent les récompenses de chaque opérateur à leurs nœuds voisins. Les récompenses partagées sont celles dernièrement obtenues. Les stratégies de sélection qui mémorisent tout l'historique des récompenses peuvent être pénalisées par le poids des récompenses passées dans leurs décisions de sélection. En effet, dans un environnement non-stationnaire [1], le changement de meilleur opérateur est possible selon l'état de la recherche. Ce point sur l'historique des récompenses est encore plus accentué dans les variantes des

1. ici, l'environnement est qualifié de non-stationnaire abusivement bien que la fonction d'optimisation ne dépend pas du temps.

stratégies collectives où l'information des nœuds voisins augmente la quantité d'information disponible. Dans ce contexte, il est plus efficace de limiter l'information passée, par exemple, par une fenêtre glissante (ϵ-Greedy, UCB) ou d'augmenter le facteur d'oubli (AP) qui prend aussi en compte l'information fournie par les nœuds voisins.

Les stratégies de sélections peuvent-être synthétisées selon deux dimensions principales : les dimensions exploitation/exploitation et mémoire des récompenses (cf. la figure 20). Nous avons tout d'abord expérimenté les stratégies de sélection sans mémoire. Elles ne prennent en considération aucune récompense comme la stratégie de sélection aléatoire, constante ou uniforme. Les stratégies de sélection collective sans mémoire considèrent seulement les récompenses obtenues à la dernière itération pour prendre leurs décisions (sauf les stratégies de sélection naïve). Il est profitable d'utiliser des stratégies de sélection collective sans mémoire lorsque le meilleur opérateur change fréquemment pour un autre opérateur (contexte dit non-stationnaire). Le cadre collectif améliore significativement les performances des stratégies de sélection sans mémoire en fournissant plus de récompenses et donc d'informations utiles par nœuds. À l'opposé, les stratégies de sélection avec mémoire prennent leurs décisions sur l'ensemble des récompenses obtenues depuis le début de l'optimisation. Dans le contexte non-stationnaire, ces stratégies de sélection sont pénalisées à la fin de l'optimisation, lorsque la recherche est proche des meilleures solutions, où un opérateur efficace peut être différent du début de la recherche autour des solutions de faible performance. Une façon de contrebalancer le poids du passé est d'augmenter la part d'exploration des méthodes. Toutes les stratégies de sélection ont généralement un paramètre contrôlant le compromis exploitation/exploitation, le choix de la valeur de ce paramètre dépendra principalement du contexte : peu d'exploration dans un contexte stationnaire, beaucoup d'exploitation dans un contexte non-stationnaire. Notons qu'un contexte non-stationnaire peut utiliser avantageusement une stratégie de sélection du contexte stationnaire, par exemple lorsqu'il n'y a pas davantage à changer l'opérateur ou lorsque la détection du meilleur opérateur est difficile (faible différence entre opérateurs).

L'étude sur les paysages NK permet de compléter l'analyse des stratégies de sélection effectuée sur le problème OneMax. En effet, la famille des paysages NK permet de régler une difficulté d'optimisation, la non-linéarité, et ainsi d'en étudier les conséquences. Nous avons pu constater les limites des méthodes adaptatives pour sélectionner le meilleur opérateur. Plus la non-linéarité du problème est grande, moins les stratégies de sélection semblent efficaces.

Deux facteurs pourraient augmenter la difficulté de sélection : la faible différence de performance entre les opérateurs et la durée d'optimisation (en nombre d'évaluation) pendant lequel un opérateur est le meilleur des opérateurs. En effet, discriminer l'opérateur peut-être difficile si la différence de

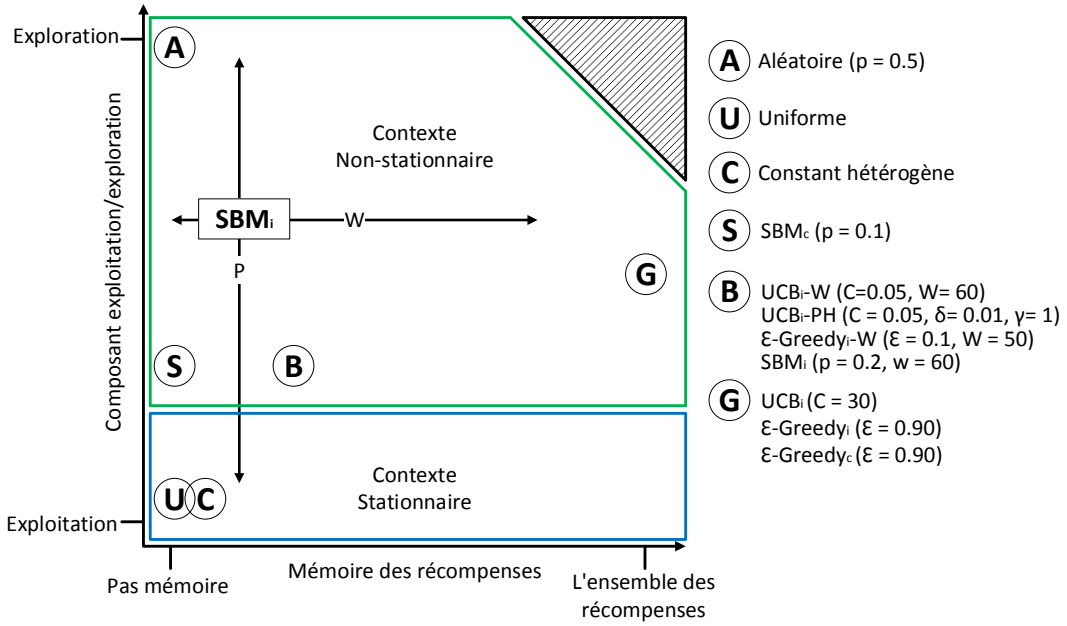


FIGURE 20 – Synthèse des stratégies de sélection selon les dimensions exploitation/exploitation et mémoire de l'historique des récompenses.

performance est faible, cela demande plus de ressource de calcul pour obtenir un échantillon de récompenses suffisamment grand afin de déterminer le meilleur opérateur. Le facteur le plus important semble être le second. Bien souvent la progression de la recherche est d'autant plus difficile, et donc plus longue, proche des solutions optimales, c.-à-d. en fin d'optimisation, et l'opérateur optimal peut être différent que celui pour les solutions de faible performance et en plus difficile à détecter pour les stratégies de sélection adaptative.

Ce chapitre à travers l'analyse de nombreuses stratégies de sélection sur les problèmes OneMax et paysages NK donne des pistes de compréhension des stratégies de sélection en environnement de calcul distribué et peut guider vers la conception de stratégies efficaces. Toutefois, les facteurs importants, encore hypothétiques, exprimés dans cette synthèse demandent de dépasser le cadre des problèmes spécifiques OneMax et NK pour traduire de manière générique les propriétés des problèmes d'optimisation qui poussent à utiliser une méthode de réglage de paramètres online adaptative ou offline.

4

BENCHMARK : MODÈLE FITNESS CLOUD

Approfondir la compréhension du comportement des stratégies de sélection permet de saisir la relation entre le problème d'optimisation et les différents opérateurs de recherche locale, ou encore, d'un point de vue plus pratique, cela permet également d'identifier la plus appropriée à utiliser en fonction des caractéristiques du problème à optimiser. Dans ce dernier cas, l'objectif est d'améliorer le temps d'optimisation et de diminuer les besoins en ressource de calcul. Nous proposons d'étudier les stratégies de sélection à l'aide d'une modélisation. Le modèle Fitness Cloud cherche à modéliser conjointement les opérateurs de recherche locale et le problème d'optimisation. Le fitness cloud (cf. 2.1.2) est un modèle probabiliste qui définit la distribution de probabilité de la qualité (fitness) des solutions produites par un opérateur en fonction de l'état de la recherche qui est aussi donné par la qualité courante de la solution. Cette modélisation semble adaptée aux problèmes OneMax et paysage NK comme nous avons pu le remarquer dans le chapitre précédent.

Étudier les facteurs influençant le comportement des stratégies de sélection dans un environnement contrôlé comme avec le modèle Fitness Cloud, nous permet : (i) de distinguer quand une stratégie de sélection adaptative est plus pertinente qu'une stratégie de sélection naïve à l'instar de la stratégie aléatoire ou une stratégie utilisant un seul opérateur pour l'ensemble de l'optimisation (ii) de proposer des éléments de conception pertinente des stratégies de sélection. À travers cette étude, nous répondons donc aux questions suivantes :

— Comment les caractéristiques du problème influencent-elles les performances des stratégies de sélection ?
— Quelle est la composition optimale d'un portefeuille d'opérateurs ?
— Dans quelle situation, une stratégie de sélection est pertinente par rapport aux stratégies naïves – c'est-à-dire de sélection aléatoire, ou de sélection constante ?

À partir de l'expressivité du modèle fitness cloud, une multitude de scénarios d'optimisation peuvent être décrits. Nous nous sommes appuyés en particulier sur les deux problèmes utilisés dans cette thèse : le problème OneMax et le paysage NK pour établir des scénarios capturant des propriétés influençant les stratégies de sélection.

Ce chapitre s'organise de la manière suivante. Nous définissons le modèle et analysons le modèle dans un premier scénario simple à un seul opérateur. Puis, trois scénarios d'optimisation seront définis et nous étudions sur

Algorithm 3 Un opérateur à métaheuristique basique à une solution

1: $x_0 \leftarrow$ initialisation()
2: **repeat**
3: **for** $i = 1 \dots \lambda_t$ **do**
4: $y_i \leftarrow$ operateur(x_t)
5: **end for**
6: $x_{t+1} \leftarrow$ selection$(x_t, y_1, \dots, y_{\lambda_t})$
7: **until** le critère d'arrêt soit vrai

les stratégies de sélection à travers ces scénarios à portefeuille d'opérateurs. Nous comparons les stratégies de sélection entre elles, et notamment avec les stratégies de référence dites uniformes et aléatoires. Enfin, nous comparons les scénarios du modèle fitness cloud avec les problèmes d'optimisation comme le OneMax et paysage NK pour approfondir les connaissances des stratégies de sélection adaptatives sur ces problèmes typiques.

4.1 BENCHMARK POUR LA STRATÉGIE DE SÉLECTION

4.1.1 *Introduction au Modèle Fitness Cloud*

Avant d'introduire les détails du modèle, et bien que le modèle proposé soit indépendant d'une métaheuristique particulière, considérons, pour plus de clarté, le modèle de l'algorithme 3 d'une métaheuristique à solution unique basée sur un seul opérateur. L'algorithme itératif considéré comporte deux parties. Tout d'abord, un opérateur de recherche locale stochastique est appliqué à la solution actuelle x_t pour produire un ensemble de λ_t solutions candidates y_i. Un tel opérateur pourrait être la mutation bit-flip uniforme lorsque l'espace de recherche est l'ensemble des chaînes binaires. Deuxièmement, une nouvelle solution actuelle x_{t+1} est sélectionnée parmi l'ensemble des solutions candidates. Ceci est généralement effectué selon les valeurs de la qualité des solutions nouvellement générées y_i et la solution courante x_t. Un exemple classique de sélection est le $(1+\lambda) - EA$ qui sélectionne la meilleure solution à chaque itération de l'algorithme. Notez que malgré sa simplicité, un tel modèle englobe une large gamme d'algorithmes.

Le modèle Fitness Cloud définit la distribution de la qualité (aussi appelée fitness) de la solution après une itération en fonction de la qualité de la solution courante. Ainsi dans le modèle Fitness Cloud, l'état de la recherche est uniquement donné par la qualité $f_t = f(x_t)$ de la solution actuelle x_t (voir Algo. 3). En supposant que la règle de sélection ne prend en compte que les valeurs de la qualité de solution (ce qui est une pratique courante pour un large éventail de métaheuristiques), aucun autre modèle particulier n'est requis pour l'étape de sélection. Mais la connaissance de l'opérateur de sélec-

tion de l'algorithme 3 est toutefois nécessaire pour finaliser le modèle sur une itération de l'algorithme. L'idée fondamentale du modèle Fitness Cloud est de supposer que la qualité de la solution après l'application d'un opérateur stochastique est donnée par une distribution de probabilité conditionnelle :

$$\Pr(f(y) = z' \mid f_t = z) \tag{6}$$

Différents choix de cette distribution de probabilité peuvent être réalisés comme des distributions discrètes (binomial, Poisson, etc.), ou des distributions continues (normale, Weibull, etc.). Compte tenu de ses propriétés de convergence, dans cette thèse, nous choisissons l'utilisation d'une distribution normale.

4.1.2 *Espérance de l'amélioration*

Pour les problèmes dit boîte noire, la distribution de probabilité et l'amélioration moyenne des opérateurs stochastiques de recherche locale ne sont pas connues a priori. Ici, de par les propriétés de convergence des lois de probabilités, nous choisissons de définir la distribution de la qualité des solutions du modèle fitness cloud par une loi normale, notée $\mathcal{N}$:

$$\Pr(f(y) = z' \mid f_t = z) \sim \mathcal{N}(\mu(z), \sigma^2(z)) \tag{7}$$

où $\mu(z)$ et $\sigma^2(z)$ sont respectivement la moyenne et la variance de la distribution normale qui dépendent de la qualité z de la solution. Les fonctions $\mu(z)$ et $\sigma^2(z)$ peuvent choisies selon le scénario d'optimisation cible. En conséquence, l'évolution de la qualité de la solution au cours d'une itération suit une distribution de probabilité conditionnelle qui intègre la qualité de la solution précédente. Une caractéristique importante pour de cette distribution est l'espérance de l'amélioration de la qualité de la solution d'une itération à l'autre de l'algorithme, désigné par $E^+(z)$. L'espérance de l'amélioration est la progression attendue de la qualité de la solution étant donné que la valeur de la qualité de la solution courant est z qui est définie formellement pour un problème de maximisation par :

$$E^+(z) = \int_z^\infty \Pr(f_{t+1} = z' \mid f_t = z)\, z'\, dz' \tag{8}$$

Il est possible de calculer analytiquement l'espérance de l'amélioration pour un modèle FC basé sur une loi normale. Ce cadre nous permet de montrer le lien entre l'espérance de l'amélioration – *expected improvement* en anglais ou EI – et les paramètres du modèle μ et σ^2.

Plus formellement l'espérance de l'amélioration est donnée en fonction du paramètre μ et d'écart-type σ :

$$\begin{aligned} E^{+}_{\mu,\sigma^2}(z) &= E[max(0, \Delta fitness)] \text{ où } \Delta fitness = f(operateur(x)) - z \\ &= \int_{-\infty}^{0} 0 \cdot \varphi_{\mu,\sigma^2}(t)dt + \int_{0}^{+\infty} t \cdot \varphi_{\mu,\sigma^2}(t)dt \\ &= \int_{0}^{+\infty} t \cdot \varphi_{\mu,\sigma^2}(t)dt \\ &= \int_{0}^{\infty} \frac{t}{\sigma\sqrt{2\pi}} exp(\frac{-(t-\mu)^2}{2\sigma^2})dt \\ &= \mu.(1 - \Phi(\frac{-\mu}{\sigma})) + \frac{\sigma}{\sqrt{2\pi}} exp(\frac{-\mu^2}{2\sigma^2}) \end{aligned} \tag{9}$$

φ_{μ,σ^2} est la densité de probabilité de la loi normale d'espérance μ et de variance σ^2, et Φ est la fonction de répartition de la loi normale centrée réduite.

De nombreuses combinaisons de moyenne μ et variance σ^2 peuvent donner une même espérance d'amélioration (voir figure 22). Plus l'espérance de l'amélioration est élevée plus les deux paramètres μ et σ sont élevés et inversement. Naturellement, plus l'espérance de l'amélioration à chaque itération est élevée, plus le nombre d'itération pour atteindre le maximum du problème est faible. Appelons $fitness_{max}$ la qualité (fitness) maximale à atteindre du maximum global. Le temps moyen de résolution en nombre d'évaluations – *Expected Runing Time* en anglais ou ERT – pour atteindre le maximum peut être calculé simplement dans un scénario où les paramètres de moyenne et variance de la loi normale sont indépendantes de la qualité de la solution (voir figure 21). En effet, dans ce scénario, le nombre d'évaluation est le rapport entre la fitness atteindre et gain moyen (espérance de l'amélioration) à chaque itération :

$$ERT(\mu, \sigma^2) = \frac{fitness_{max}}{E^{+}_{\mu,\sigma^2}} \tag{10}$$

Pour une moyenne μ fixée, plus la variance σ^2 est grande, plus l'espérance de l'amélioration est grande et le temps moyen de résolution est court. Pour une variance σ^2 fixée, plus la moyenne est grande, plus l'espérance de l'amélioration sera grande et le temps moyen de résolution est court (voir figure 21).

4.2 DÉFINITION DES TROIS SCÉNARIOS

Le modèle Fitness Cloud peut se décliner en de nombreux scénarios d'optimisation. Ici, nous cherchons à comprendre à l'aide de celui-ci le comporte-

FIGURE 21 – Scénario à un seul opérateur à espérance constante.

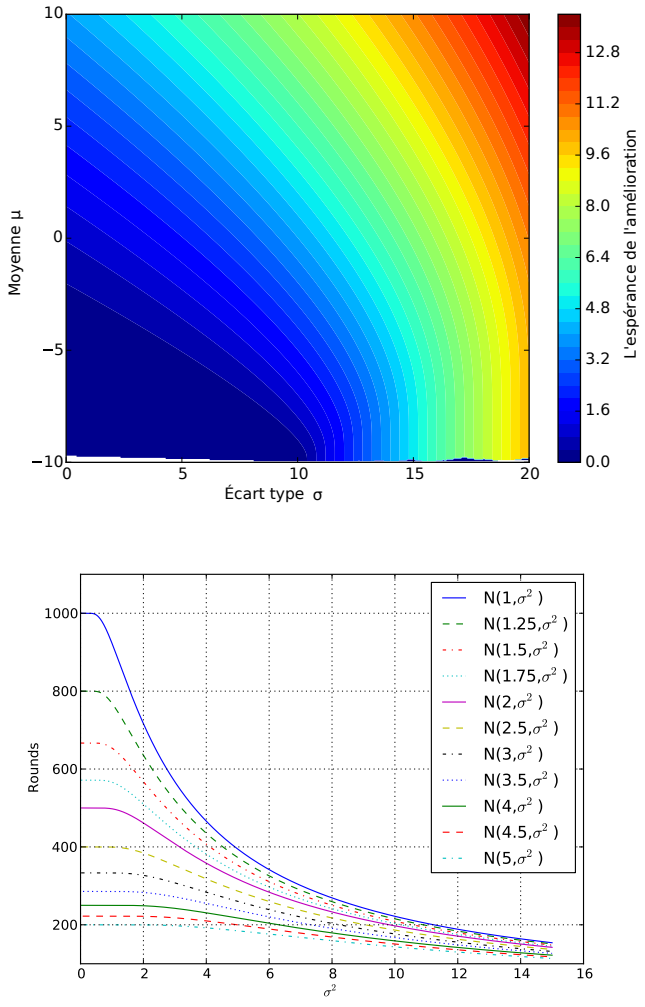


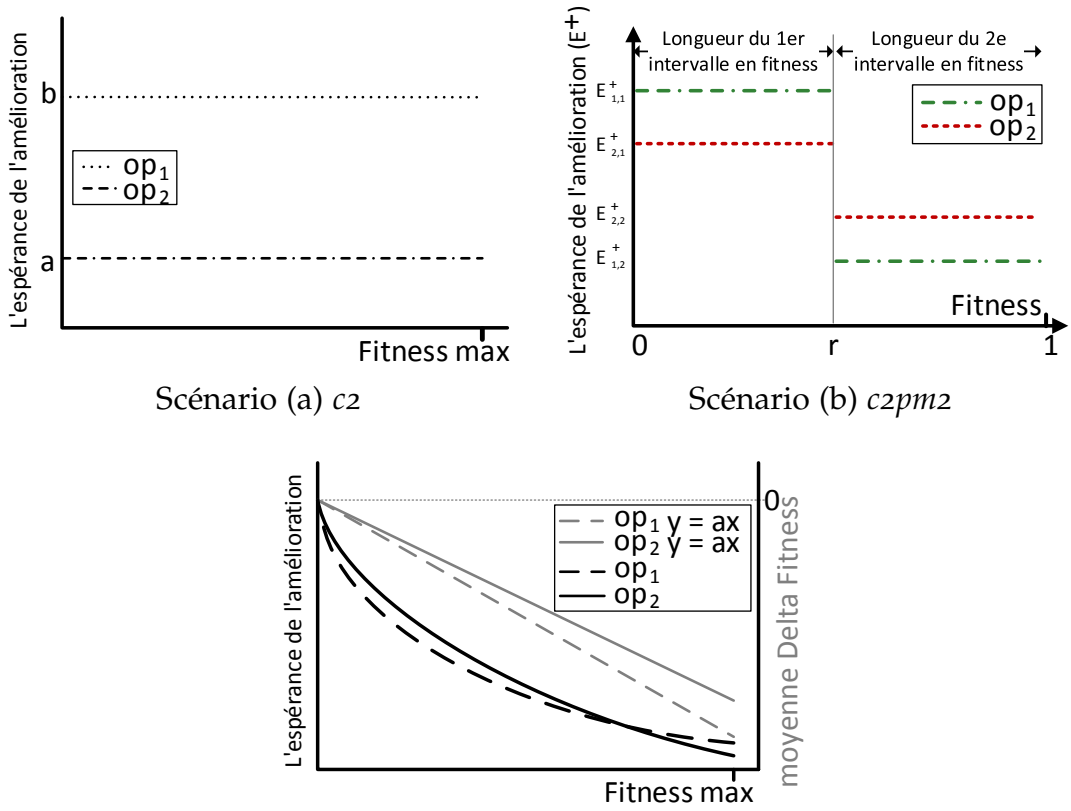
FIGURE 22 – En haut, espérance de l'amélioration en fonction de μ et σ^2 de la loi normale. En bas, temps moyen de résolution en nombre d'évaluations (Expected Runing Time).

ment et les performances des stratégies de sélection en fonction des caractéristiques du problème d'optimisation et du portefeuille de paramètres/opérateurs/algorithmes. Ainsi, nous pouvons analyser les éléments de conception des algorithmes adaptatifs visés. Un scénario à un opérateur est décrit et analysé au début de ce chapitre, le portefeuille ne comporte qu'un seul opérateur dont l'amélioration moyenne est constante. Ce scénario de base ne permet pas d'analyser une stratégie de sélection, mais sert de référence. Pour ce faire, nous proposons d'étudier en particulier trois scénarios au nombre de caractéristiques croissant. Ils sont résumés à la figure 23. Pour ces scénarios, le portefeuille est composé de 2 opérateurs ou plus. Dans le deuxième scénario *c2*, chaque opérateur a une performance constante, quelle que soit la valeur de fitness de la solution. Dans le troisième scénario *c2pm2*, les performances de chaque opérateur sont constantes par intervalle. Enfin, dans le dernier scénario *ld2*, la performance moyenne de chaque opérateur décroit linéairement en fonction de la fitness de la solution.

Dans le scénario *c2* deux opérateurs sont considérés. Il permet d'étudier le comportement de stratégies de sélection dans une situation de sélection sans changement d'opérateur optimal au cours de la recherche. Dans le scénario *c2pm2*, il y a dégradation et inversion des performances des opérateurs pour une certaine valeur de fitness. Il nous permet de comprendre comment le réapprentissage se comporte, c'est-à-dire la capacité de détecter un nouveau meilleur opérateur au cours de la recherche (environnement non-stationnaire). Le scénario *ld2* s'appuie sur une modélisation plus fine des problèmes de type OneMax et le NK. Nous cherchons à mieux saisir le comportement des stratégies de sélection dans le cadre d'une situation où la dégradation de la performance des opérateurs est continue. Les sous-sections suivantes précisent les scénarios.

4.2.1 *Scénario c2 : deux opérateurs de qualité constante*

Dans ce scénario, les deux opérateurs du portefeuille ont des performances constantes, quelle que soit la valeur de fitness des solutions. Comme montrer à la section 2.2, la qualité des opérateurs (et le temps de résolution) est caractérisée par l'espérance de l'amélioration. Ainsi, le scénario est défini par deux espérances de l'amélioration. Toutefois, chaque espérance d'amélioration de chaque opérateur est déterminée par les deux paramètres (moyenne et variance) de la loi normale, et ce scénario dépend de quatre paramètres, dont deux réellement indépendants. Notons que de fixer l'espérance de l'amélioration pour chaque opérateur revient à déterminer la différence de l'espérance de l'amélioration entre les deux opérateurs. À l'aide de ce scénario, nous étudions la convergence et le temps de convergence de la stratégie de sélection vers le l'opérateur le plus pertinent, c'est-à-dire celui qui a de meilleures performances, en fonction principalement de leur différence de qualité.



Scénario (a) *c2* Scénario (b) *c2pm2*

Scénario (c) *ld2*

FIGURE 23 – Espérance de l'amélioration en fonction de la fitness de la solution. Scénario (a) *c2* a deux opérateurs d'amélioration moyenne constante. Scénario (b) *c2pm2* a deux opérateurs d'amélioration moyenne constante par intervalle. Scénario (c) *ld2* a deux opérateurs dont les performances moyennes (fitness) décroit linéairement et l'amélioration moyenne se dégrade rapidement.

4.2.2 *Scénario c2pm2 : deux opérateurs de qualité constante par intervalle*

Dans ce scénario, nous définissons aussi un portefeuille composé de deux opérateurs. À la différence du scénario, les performances des opérateurs ne sont pas constantes, elles sont constantes seulement par intervalle. En effet, elles changent à une valeur de fitness fixées définissant deux intervalles de fitness (voir figure 24). Plus précisément, nous supposons d'abord que les valeurs de la qualité de solution sont comprises entre 0 et 1. Nous supposons que la recherche commence par la valeur de qualité de solution 0 et s'arrête lorsque la valeur $fitness_{max} = 1$ est atteinte. Cet intervalle $[0; 1]$ est ensuite divisé en deux intervalles : le premier de 0 à $r \leqslant 1$, et le second de r à 1. Le paramètre r est donc un premier paramètre de ce scénario définissant la longueur des intervalles. Nous considérons alors un portefeuille de deux opérateurs ayant des performances relatives différentes dans ces

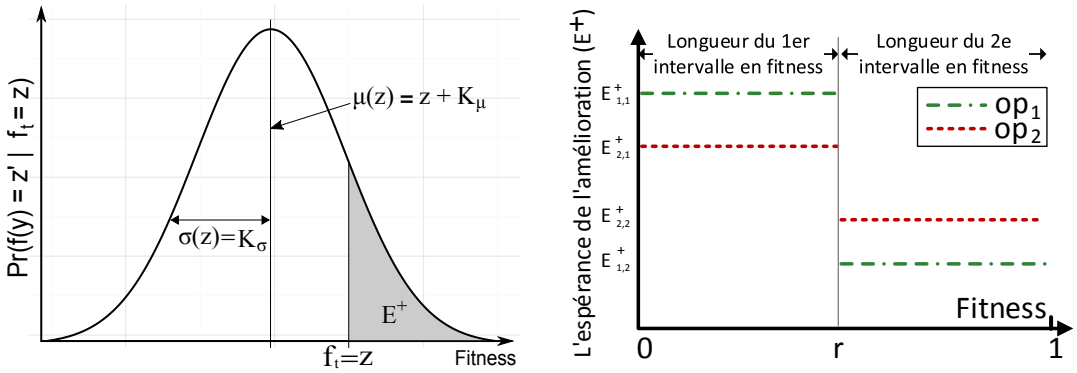


FIGURE 24 – Modèle Fitness Cloud : scénario avec deux opérateurs et deux intervalles de fitness.

deux intervalles. Comme avec le modèle Fitness Cloud, la distribution de fitness après application de l'opérateur est modélisée par une loi normale définie par une moyenne et une variance (cf. équation 7). Sur chacun des deux intervalles et pour chaque opérateur, les moyennes et variances sont définies par : $\mu_i(z) = z + K_{\mu_i}$ et $\sigma_i^2(z) = K_{\sigma_i}$ où pour chaque opérateur op_i, $i \in \{1, 2\}$, les paramètres K_{μ_i} et K_{σ_i} sont des nombres constants différents à chaque intervalle de qualité de la solution. Ainsi, 9 paramètres sont à définir dans ce scénario : le paramètre $r \in]0, 1[$, et les 8 paramètres pour les distributions normales pour chaque opérateur et à chaque intervalle de fitness. Cependant, comme il sera montré dans la suite, le temps moyen de résolution en itération optimal à atteindre est de valeur 1 dépend de l'espérance de l'amélioration de chaque opérateur sur chaque intervalle. Par conséquent, seulement 5 paramètres sont libres comme illustré par la figure 24 ; où $E^+_{i,j}$ désigne l'espérance de l'amélioration de la fitness pour un opérateur op_i, $i \in \{1, 2\}$, pour l'intervalle de fitness $j \in \{1, 2\}$. En outre, nous supposons que le meilleure opérateur pour le premier intervalle de fitness est op_1, tandis qu'il passe à op_2 dans le second intervalle de fitness, c'est-à-dire $E^+_{2,1} < E^+_{1,1}$, et $E^+_{1,2} < E^+_{2,2}$. Enfin, comme dans de nombreux problèmes d'optimisation, nous supposons que l'espérance de l'amélioration diminue lorsque la l'intervalle de fitness augmente : $E^+_{1,2} < E^+_{1,1}$, et $E^+_{2,2} < E^+_{2,1}$. Il est important de noter que la performance relative des algorithmes dans le portefeuille ne dépend pas du temps (nombre d'itérations), ni du nombre de fois où un opérateur est appliquée ; mais uniquement de l'état de la recherche qui est supposée être donnée par la valeur de fitness actuelle.

ÉTUDE ANALYTIQUE DU TEMPS DE RÉSOLUTION. Dans ce paragraphe, nous analysons formellement la stratégie de sélection optimale, appelée oracle, qui sélectionne le meilleur opérateur sur chaque intervalle et la stratégie constante qui sélectionne toujours le même opérateur. L'analyse consiste à

donner les bornes inférieure et supérieure du temps moyen de résolution (Expected Running Time, ERT, voir section 4.1.2).

Nous supposons qu'une stratégie de sélection constante choisit un opérateur arbitraire, notée op, du portefeuille et l'exécute dans le cadre du scénario précédemment décrit jusqu'à atteindre la valeur de fitness cible 1. Nous supposerons que l'opérateur considéré suit le modèle de l'algorithme 3. L'algorithme est initialisé avec une solution ayant une valeur fitness 0 et réalise une sélection élitiste, c'est-à-dire que la meilleure solution est retenue pour la prochaine itération. L'espérance de l'amélioration de l'opérateur op sur chaque itération est par définition constante. Nous notons par E_1^+ (respectivement E_2^+), l'espérance de l'amélioration sur le premier (respectivement second) intervalle de fitness. Considérons le temps d'optimisation, c'est-à-dire, le temps (mesuré en nombre d'itérations) pour atteindre la valeur de fitness finale : $T = \min\{t \ : \ F_t \geqslant 1\}$ où F_t est la variable aléatoire qui donne la valeur de fitness de la meilleure solution trouvée à l'itération t. Notez que le nombre total d'évaluations dépend de λ_t et le nombre d'évaluations pour chaque opérateur. Nous pouvons ensuite prouver ce qui suit :

Thèorème 1 *Le temps moyen de résolution,* $E[T]$*, (expected running time) est borné par :* $T_{up} - \frac{\delta}{E_2^+} \leqslant E[T] \leqslant T_{up}$ *avec*

$$T_{up} = \frac{r}{E_1^+} + \frac{1-r}{E_2^+} \quad et \quad \delta = \begin{cases} 1-r & si\ 1-r \leqslant E_1^+, \\ E_1^+ & si\ E_1^+ < 1-r \end{cases}$$

Preuve 1 *Soit* $T_1 = \min\{t \ : \ F_t \geqslant r\}$ *et* $T_2 = \min\{t - T_1 \ : \ F_t \geqslant 1\}$*. Par définition et par linéarité de l'espérance, nous avons* $E[T] = E[T_1] + E[T_2]$*. Maintenant, prouvons le lemme suivant pour obtenir le résultat :*

Lemme 1 *Soit deux réels* $(k, \ell) \in [0,1]^2$ *tels que* $r \leqslant k \leqslant \ell$ *ou* $k \leqslant \ell \leqslant r$*. Soit le temps* $T' = \min\{t \ : \ F_t \geqslant \ell\}$ *avec* $F_0 = k$*. Posons* E^+ *l'espérance de l'amélioration de fitness sur l'intervalle* $[k, \ell]$*. Alors,* $E[T'] = (\ell - k)/E^+$*.*

*La preuve de ce lemme est une application du théorème 1 de l'article de Lehre et Witt [***Lehre:2013***] dont le principe est celui du drift additif : lorsque l'espérance de l'amélioration est bornée par une constante, alors le temps moyen pour atteindre la cible est borné par la valeur de la cible divisée par la constante. Plus précisément, en considérant la variable aléatoire* $X_t = \ell - F_t$*, nous avons par définition du modèle fitness cloud* $E[X_t - X_{t+1} \ : \ X_t] = E^+$ *ce qui donne la condition nécessaire du drift additif du théorème et la conclusion du lemme est alors immédiate.*

Avec ce lemme, le reste de la preuve peut s'établir. Comme l'espérance de l'amélioration sur le premier intervalle de fitness est E_1^+*, l'application du lemme précédent avec* $F_0 = 0$ *fournit :* $E[T_1] = r/E_1^+$*. De même, soit* $k \geqslant r$ *la valeur de fitness de*

la (meilleur) solution juste après avoir trouvée pour la première fois une solution x_t *de fitness plus grand que* r. *Comme l'espérance de l'amélioration sur le second intervalle de fitness est* E_2^+, *l'application du lemme précédent avec* $F_0 = k \geqslant r$ *donne :* $E[T_2] = (1-k)/E_2^+ \leqslant (1-r)/E_2^+$. *Ce qui donne la borne supérieure énoncée par le théorème. Pour obtenir la borne inférieure, définissons maintenant* $Y_{t'} = 1 - F_{t'}$ *où* $t' = t - T_1$. *Par conséquent,* $E[T_2 \ : \ Y_0] = Y_0/E_2^+$. *En appliquant le théorème de l'espérance totale, nous avons* $E[T_2] = E[Y_0]/E_2^+$. *Soit* $F_{T_1} = F_{T_1-1} + \Delta_{T_1-1}$ *où* Δ_{T_1-1} *est la variable aléatoire de la différence de fitness entre les itérations* $T_1 - 1$ *et* T_1. *Par définition de* T_1, $F_{T_1-1} < r$, *et selon les modalités du scénario, nous avons* $E[\Delta_{t'-1}] = E_1^+ > 0$. *Il s'ensuit que :* $E[Y_0] \geqslant 1 - r - E_1^+$. *Lorsque* $1 - r - E_1^+ \leqslant 0$, *l'algorithme est capable d'atteindre fitness final sans aucune itération dans le second intervalle de fitness. Dans le cas contraire, pour* $1 - r - E_1^+ > 0$, *l'algorithme dépense au moins* $(1 - r - E_1^+)/E_2^+$ *itérations dans le second intervalle de fitness ce qui conclue la preuve du théorme* □

Dans le cas d'un algorithme $(1+1)$-EA, l'espérance de l'amélioration $E^+(z)$ a la valeur fitness z donnée par :

$$E^+(z) = (K_\mu - z).(1 - \Phi(\frac{-(K_\mu - z)}{K_\sigma})) + \frac{K_\sigma}{\sqrt{2\pi}} \exp(\frac{-(K_\mu - z)^2}{2K_\sigma^2}) \qquad (11)$$

où Φ est la fonction de distribution de probabilité cumulative de la loi normale centrée réduite. En conséquence, le temps d'exécution moyen pour les deux opérateurs possibles op_1 et op_2 en fonction de la longueur r du premier intervalle de fitness, est illustrée sur la figure 24 à droite.

4.2.3 *Scénario ld2 : non-statique avec une récompense décroissante*

Dans ce scénario, la fonction donnant l'espérance de l'amélioration des deux opérateurs a été modélisée pour s'approcher encore plus de ce que l'on peut constater pour des problèmes d'optimisation comme les problèmes OneMax et paysage NK. Ici, l'espérance de l'amélioration décroit de façon continue à mesure que la performance des solutions s'améliore. Dans ce scénario, nous avons également un portefeuille composé de deux opérateurs dont la distribution de fitness après application de l'opérateur i suit une loi normale de moyenne $\mu_i(z)$ et variance $\sigma_i^2(z)$ où z est la valeur de fitness de la solution considérée. La variance est encore supposée constante $\sigma_i^2(z) = K_{i,\sigma^2}$. Par contre, la moyenne décroit linéairement avec la fitness z : $\mu_i(z) = a_i \times z$ avec a_i un nombre réel négatif et z une valeur de fitness comprise entre 0 et 1. Pour une solution initiale avec $z = 0$, en moyenne, l'opérateur n'améliore pas la qualité de la solution de la même manière que classiquement un opérateur permet d'obtenir une autre solution aléatoire initialement. La négativité du coefficient a_i modélise le fait que plus la recherche avance vers les bonnes solutions, plus il est difficile d'obtenir de meilleure solution.

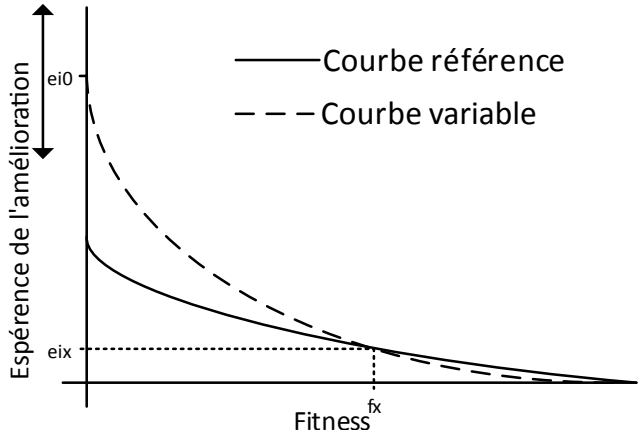
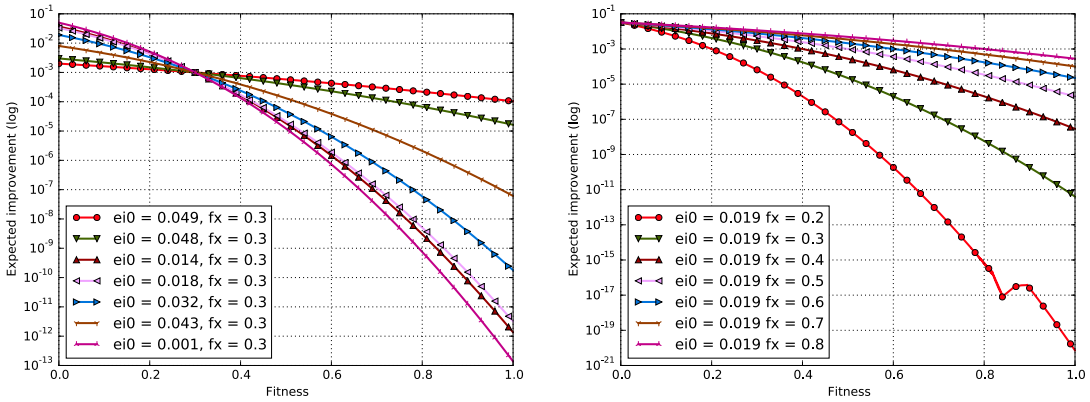


FIGURE 25 – Espérance de l'amélioration en fonction de la valeur de fitness pour deux opérateurs op_1 et op_2 avec un point intersection en (fx, eix) et une amélioration moyenne initiale $ei0$.

Pour définir complètement le profil d'un opérateur, nous avons donc besoin de définir la pente a_i et l'écart-type σ_i. De nouveau, il est important de comparer l'espérance de l'amélioration des deux opérateurs (cf. figure 25). L'élément important du scénario est l'intersection des espérances d'amélioration des deux opérateurs, c'est-à-dire la valeur de performance pour laquelle la qualité relative entre les deux opérateurs s'inverse. Le point d'intersection est repéré par les coordonnées (fx, eix). Une autre point remarquable de la fonction donnant l'espérance de l'amélioration de chaque opérateur est l'espérance de l'amélioration initiale $ei0$ définie en $z = 0$. Nous déterminons les paramètres a_i et σ_i^2 de l'opérateur i à l'aide de ces points remarquables choisis. Ainsi, nous fixons exactement la différence de performance entre les opérateurs initialement et quand la qualité des opérateurs s'inverse. Les paramètres a_i sont déterminés par recherche dichotomique en utilisant l'espérance de l'amélioration (voir équation 10) et $\sigma = ei0 * \sqrt{2 \times \pi}$.

La figure 26 présente plusieurs exemples l'espérance de l'amélioration d'opérateurs en fonction de la valeur de fitness pour différentes valeurs de paramètres $ei0$, fx, et eix. Le point d'intersection est pour (a) $(fx, eix) = (0.3, 10^{-3})$ et pour (b) le point d'intersection varie en fonction de fx.

Le temps de résolution pour atteinte l'optimum $fitness_max = 1$ d'un opérateur op est la somme des temps de résolution avant et après le point d'inversion (fx, eix). Soient deux opérateurs op_0 et op_1, où la performance de l'opérateur 0 est meilleur que l'opérateur 1 avant l'intersection et inversement après le l'intersection. Si T_i^j est le temps (nombre d'itération) de l'opé-



(a) (b)

FIGURE 26 – Espérance de l'amélioration en échelle logarithmique en fonction de la valeur de fitness. Le paramètre eix est égale à 10^{-3}. À gauche (a), différents opérateurs ayant les mêmes $ei0 = 1.9 * 10^{-2}$ et $eix = 10^{-3}$ et différentes valeur de fx. À droite (b), différents opérateurs ayant les mêmes $fx = 0.3$ et $eix = 10^{-3}$ et différente valeur $ei0$.

rateur op_i sur l'intervalle j, alors le temps de résolution total T_i de chaque opérateur i est donné par :

$$\begin{cases} T^0 = T_0^0 + T_1^0 \\ T^1 = T_0^1 + T_1^1 \end{cases} \tag{12}$$

Nous pouvons en déduire le temps de résolution (en nombre d'itérations) pour une stratégie appelée *oracle* qui sélectionne l'opérateur optimal à chaque itération : $T_{oracle} = T_0^0 + T_1^1$.

4.3 ANALYSE EXPÉRIMENTALE

Dans cette section, nous étudions expérimentalement les trois scénarios : c2 (a), cpm2 (b), et ld2 (c). Dans le scénario *c2*, nous étudions la convergence des stratégies de sélection à deux opérateurs, c'est-à-dire le temps de convergence vers un opérateur en fonction des caractéristiques des opérateurs et de la taille du portefeuille d'opérateurs. Dans le scénario *cpm2*, nous étudions les stratégies de sélection adaptative et les stratégies de référence face à un changement de performance relative des opérateurs. Le scénario *ld2*, le plus prospectif et le plus proche des problèmes académiques comme les problèmes OneMax et NK, permet d'étudier une forte décroissante continue des performances des opérateurs. Nous effectuerons d'ailleurs un parallèle avec les problèmes OneMax et NK. L'étude de l'ensemble des scénarios est réaliser dans le cadre séquentiel avec un nœud de calcul et la valeur du para-

mètre λ égale à 64. Chaque résultat est la moyenne calculée à partir de cents exécutions indépendantes.

4.3.1 *Scénario c2 : convergence vers le meilleur opérateur*

Dans cette section, nous étudions tout d'abord la situation avec deux opérateurs op_1 et op_2 en fonction de l'espérance de l'amélioration relative de ceux-ci. Ensuite, nous étudions la convergence dans une situation avec plus de deux opérateurs.

Pour toute la suite, nous appelons les stratégies de sélection de référence (ou encore naïve), la stratégie *oracle* qui sélectionne le meilleur opérateur en fonction de l'état de la recherche, et la stratégie *aléatoire* qui sélectionne l'un des opérateurs aléatoirement de manière équiprobable à chaque itération de l'algorithme. Trois autres familles de stratégies sont analysées : les stratégies UCB, SBM et ϵ-greedy (voir section 2.5.2).

PERFORMANCE AVEC UN PORTEFEUILLE À DEUX OPÉRATEURS Nous étudions d'une part, le temps moyen de résolution en fonction de la performance relative entre les deux opérateurs et d'autre part, la fréquence de sélection des opérateurs (voir figure 27). Les deux fréquences de sélection des deux opérateurs étant naturellement complémentaire. Pour chaque stratégie, les méta-paramètres sont choisis parmi l'ensemble des valeurs décrites au tableau 1. Pour chaque valeur de l'espérance d'amélioration de l'opérateur 2, chaque valeur des méta-paramètres est testée. Les méta-paramètres qui obtiennent le meilleur rang moyen sur cet ensemble est alors choisi.

Dans cette série d'expériences l'opérateur op_1 a une espérance de l'amélioration de 10^{-3} et l'opérateur op_2 a une espérance de l'amélioration variable indiquée en abscisse de la figure 27. Lorsque l'espérance de l'amélioration op_2 est à égal à 10^{-3}, la même valeur que l'espérance de l'amélioration de op_1, les performances des deux opérateurs sont identiques, et toutes les stratégies de sélection sont équivalentes et optimales.

Globalement, pour toutes les stratégies, lorsque les espérances de l'amélioration les deux opérateurs sont proches (mais non nulles), c'est-à-dire pour une espérance de op_2 autour de 0.002, la sélection de l'opérateur ayant la meilleure performance est plus difficile. En effet, dans ce cas, à la fois de temps de résolution moyen et la fréquence de sélection de l'opérateur le moins performant sont important.

La stratégie de sélection UCB est la plus performante des stratégies ($c = 5 \times 10^{-4}$). Cette stratégie sélectionne le moins souvent l'opérateur le moins intéressant et le temps de résolution est pratiquement similaire à celui d'une stratégie oracle. La stratégie de sélection SBM, la version indépendante utilisant une fenêtre glissant, obtient les moins bonnes performances. Cette stratégie utilise l'opérateur le moins performant en moyenne 20% du temps

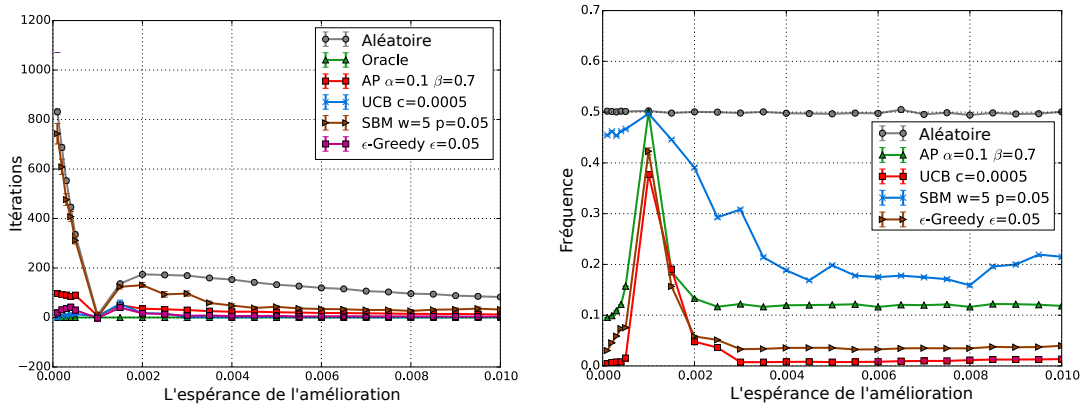


FIGURE 27 – À gauche, nombre d'itérations moyen en fonction de l'espérance de l'amélioration de l'opérateur op_2 pour chaque stratégie de sélection. À droite, fréquence d'utilisation de l'opérateur ayant de mauvaises performances en fonction de l'espérance de l'amélioration de l'opérateur op_2. L'espérance de l'amélioration de l'opérateur op_1 est de 10^{-3}.

lorsque l'espérance de l'amélioration de l'opérateur 2 est supérieure à 4×10^{-3}.

Dans ce scénario où les opérateurs ne changent pas de performance en fonction de l'état de la recherche, et où il s'agit seulement de détecter le même meilleur opérateur tout au long de la recherche, la vitesse de convergence de la stratégie de sélection UCB est très rapide, proche d'une stratégie oracle. Toutefois, la proximité de performance des opérateurs a tendance à réduire le temps de convergence.

DYNAMIQUE DE SÉLECTION POUR LA STRATÉGIE UCB Les paramètres de la stratégie de sélection influencent la qualité de l'apprentissage de l'opérateur le plus pertinent. Nous étudions pour la stratégie de sélection UCB qui est la meilleure stratégie de sélection dans ce contexte, comment le paramètre C influence l'apprentissage. La figure 28 montre la proportion d'utilisation de l'opérateur le plus pertinent à choisir à savoir l'opérateur op_1 avec une espérance de l'amélioration 10^{-3}. Quand la valeur C est égale à 5×10^{-4}, l'opérateur le plus pertinent est utilisé la plupart du temps sauf au début de l'optimisation car il y a un temps d'apprentissage. La convergence de l'opérateur est très rapide en une cinquantaine d'itérations en moyenne. Plus la valeur C est dégradée, plus la stratégie de sélection choisira l'opérateur le moins pertinent. Autrement dit, le facteur d'exploration est plus élevé.

PERFORMANCE AVEC UN PORTEFEUILLE À PLUSIEURS OPÉRATEURS L'une des sources de difficulté dans la sélection du meilleur opérateur est le nombre d'opérateurs du portefeuille. Dans ce scénario, nous considérons toujours

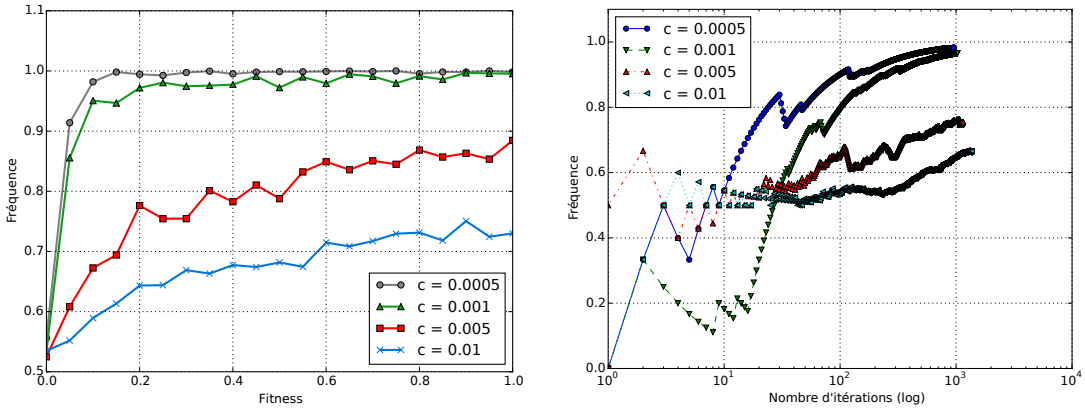


FIGURE 28 – (a) Fréquence d'utilisation du meilleur opérateur op^* en fonction de la fitness (avec une taille de fenêtre de 0.05) suivant les différents paramètres de C. op_1 avec $EI(op_1) = 0.001$ $EI(op_2) = 0.0004$. (b) Proportion de sélection depuis le début de l'optimisation de l'opérateur op_1 en fonction du nombre d'itération en échelle logarithmique avec $EI(op_1) = 0.001$ $EI(op_2) = 0.0004$.

des opérateurs ne changeant pas de performance, mais le portefeuille est composé d'un seul opérateur de bonne performance (espérance de l'amélioration de 2×10^{-3}), noté op_g pour « good », et d'opérateurs identiques avec de moins bonnes performances (espérance de l'amélioration de 4×10^{-4}), noté op_b pour « bad ». Nous analysons les performances des stratégies de sélection en fonction du nombre d'opérateurs composant le portefeuille.

La figure 29 présente le temps moyen de résolution (ERT) nécessaire pour atteindre la fitness maximale en fonction du nombre d'opérateurs. Les performances des stratégies de sélection se dégradent à mesure que le nombre d'opérateurs de mauvaise performance augmente dans le portefeuille. Plus il y a d'opérateurs, plus le temps pour trouver le meilleur parmi les autres est important. Les stratégies de sélection doivent tester/explorer plus d'opérateurs avant de détecter le meilleur opérateur. La stratégie Select-Best-Mutate (SBM) obtient de mauvaises performances similaires à la stratégie de sélection aléatoire. Par contre, la stratégie de sélection UCB obtient les meilleurs performances. Le temps de résolution moyen se dégrade avec le nombre d'opérateurs linéairement, d'équation $y = 6.50x + 999.91$ avec R^2 de 0.995. Le temps de résolution moyen pour les stratégies de sélection AP et ϵ-Greedy croît de manière significative et bien plus importante que la stratégie UCB. La stratégie UCB détecte efficacement le meilleur opérateur dans un contexte de plusieurs opérateurs aux performances constantes. Le scénario suivant va permettre de contraster ce résultat lorsque les performances des opérateurs varient.

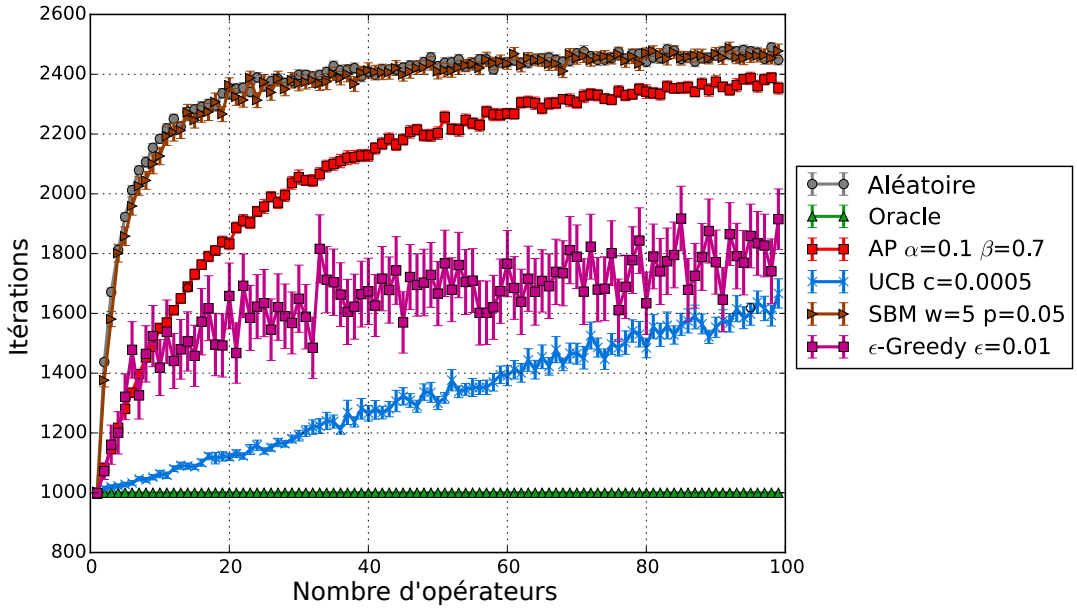


FIGURE 29 – Temps moyen de résolution (ERT) en fonction du nombre d'opérateurs que compose le portefeuille. Un seul opérateur est de bonne performance op_g et les autres op_b sont identiques de qualité plus faible : $EI(op_g) = 10^{-3}$ et $EI(op_b) = 4 \cdot 10^{-4}$.

4.3.2 *Scénario c2pm2 : comparaison des stratégies de sélection lors d'un changement*

Le scénario (b) est décrit à la section 4.2.2. Pour chaque opérateur, l'espérance de l'amélioration est constante sur deux intervalles de fitness, l'espérance étant plus petite pour chacun des opérateurs sur le second intervalle. Sur le premier intervalle, l'opérateur 1 est meilleur que l'opérateur 2 et inversement sur le second intervalle. Ainsi, nous pouvons comparer les capacités d'adaptation des méthodes de sélection lors d'un changement d'opérateur.

3 cas d'étude particuliers sont retenus dans ce scénario. Les paramètres de ces 3 cas qui précise l'espérance de l'amélioration (et les paramètres de la loi normale relatifs) sont donnés dans le tableau 30. Pour les 3 cas, les opérateurs sont identiques sur le premier intervalle de fitness. L'espérance de l'amélioration (EI) de l'opérateur 1 est deux fois supérieures à celle de l'opérateur 2 (resp. 6.10^{-3} et 3.10^{-3}). Dans le cas 1, l'EI sur le second intervalle est du même ordre de grandeur que sur le premier intervalle, et l'opérateur 1 a une performance inférieure mais qualité proche de l'opérateur 2 (resp. $1,8.10^{-3}$ et 2.10^{-3}). Il sera donc difficile dans ce cas d'identifier l'opérateur optimale sur le second intervalle. Dans les cas 2 et 3, l'EI de l'opérateur 2 est deux fois supérieur à celui de l'opérateur 1 sur le second intervalle. Dans le cas 2, l'EI est du même order de grandeur que sur l'intervalle 1 (resp. 10^{-3} et 2.10^{-3}), alors que dans le cas 3, un facteur d'ordre 10 a été choisi (resp. $0,2.10^{-3}$

TABLE 4 – Paramètres des 3 cas : espérance de l'amélioration ($E^+_{i,j}$), moyenne (K_{μ_i}), et écart-type (K_{σ_i}). Les valeurs sont données avec un facteur de 10^{-3}.

Cas	Op.	Intervalle de fitness 1			Intervalle de fitness 2		
		$E^+_{i,1}$	K_{μ_i}	K_{σ_i}	$E^+_{i,2}$	K_{μ_i}	K_{σ_i}
Cas 1	op_1	6	−1	16.27	1.8	−2	6.72
	op_2	3	−1	8.72	2	−2	7.24
Cas 2	op_1	6	−1	16.27	1	−2	4.59
	op_2	3	−1	8.72	2	−2	7.25
Cas 3	op_1	6	−1	16.27	0.2	−2	2.14
	op_2	3	−1	8.72	0.4	−2	2.84

et $0, 4.10^{-3}$) pour modéliser une optimisation difficile en fin d'optimisation. Dans tous les cas, la longueur du premier intervalle r est un paramètre libre entre 0 et 1 qui sert à contraster les performances des stratégies de sélection.

Les paramètres expérimentaux des stratégies de sélection sont définies par rapport au chapitre précédant 3.2.2. Pour la stratégie UCB, l'ensemble des paramètres C étudiés est $\{0.0008, 0.01, 0.1, 0.75, 2, 4, 10, 20, 25, 50\}$, et pour la stratégie AP, les paramètres de l'adaptation α et du taux d'apprentissage β sont $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Nous incluons trois autres stratégies dans notre analyse. La stratégie *oracle* sélectionne le meilleur opérateur dans chaque intervalle de fitness. La *stratégie aléatoire* sélectionne à chaque itération un opérateur au hasard de manière équiprobable. Notez que, dans ce cas, l'espérance de l'amélioration est la moyenne des espérances de l'amélioration des deux opérateurs. La *stratégie uniforme* sélectionne toujours le même opérateur, soit op_1 ou op_2. Cet opérateur est choisi avant que l'exécution soit commencé. Tous les résultats sont calculés en moyenne sur 100 exécutions indépendantes. La figure 30 représente la borne supérieure du nombre moyen d'itérations pour les stratégies oracle, uniforme et aléatoire en fonction de la longueur du premier intervalle de fitness (r).

ANALYSE DES RÉSULTATS Nous analysons la performance relative et le comportement des différentes stratégies de sélection. Les stratégies adaptatives sont comparées aux stratégies uniformes et aléatoire, et les stratégies UCB et d'AP sont également comparées individuellement. La mesure de performance est le nombre moyen d'itérations pour atteindre la valeur de fitness 1. Pour chaque stratégie, nous considérons les meilleurs méta-paramètres de la manière suivante. Pour chaque cas, nous calculons le rang moyen (la performance étant le nombre d'itérations) d'un paramètre sur toutes les valeurs possibles r de la longueur du premier intervalle et le paramètre le mieux

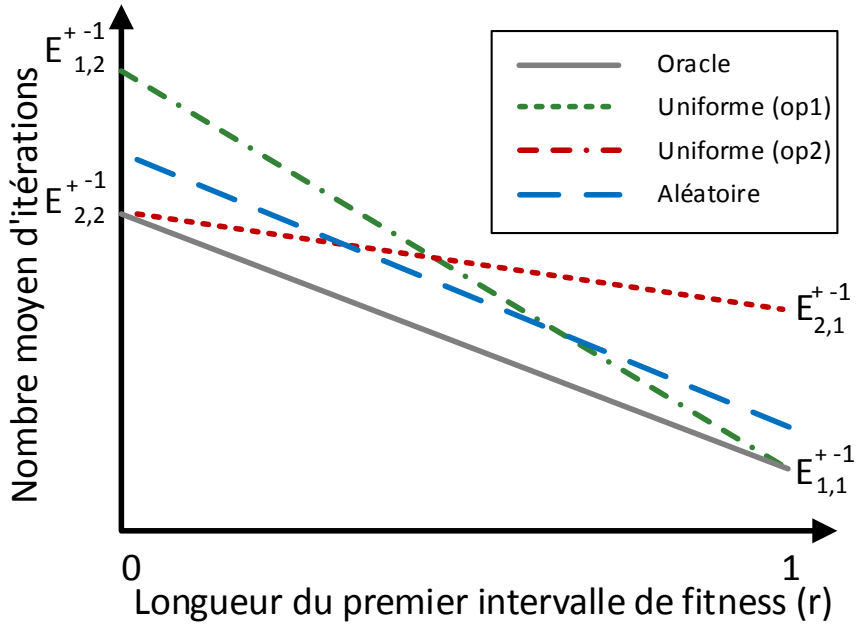


FIGURE 30 – Représentation de la borne supérieure du nombre moyen d'itérations (expected running time) en fonction de la longueur du premier intervalle de fitness (r) pour la stratégie oracle, uniforme et aléatoire.

classé est sélectionné. La comparaison de performance pour calculer le rang est basée sur le test de Wilcoxon-Mann-Whitney à un niveau de confiance de 0.05.

La figure 31 montre la performance obtenue dans les trois cas d'essai en fonction de la longueur r du premier intervalle. Notez que la différence moyenne sur l'ensemble des valeurs r du nombre d'itérations entre l'oracle et la meilleure stratégie uniforme (soit avec op_1 ou op_2) est de 20, 57 et 100 évaluations respectivement pour les cas 1, 2 et 3. Ces différences sont les différences maximales de performance entre une méthode adaptative optimale et une méthode optimale de réglage automatique avant l'optimisation.

Stratégies adaptative *vs.* stratégies aléatoire. En conséquence du résultat du théorème 1, la performance moyenne des stratégies de sélection aléatoire, oracle et uniforme diminue linéairement avec la longueur r du premier intervalle de fitness. Nous observons aussi à partir de nos données empiriques que la performance des stratégies UCB et AP décroit également linéairement. Pour tous les cas de test et pour toute longueur r de premier intervalle, les stratégies UCB et AP obtiennent des performances nettement supérieure à la stratégie aléatoire. Il est intéressant de noter que l'écart moyen entre UCB et la stratégie aléatoire dans les cas 1, 2 et 3 est approximativement respec-

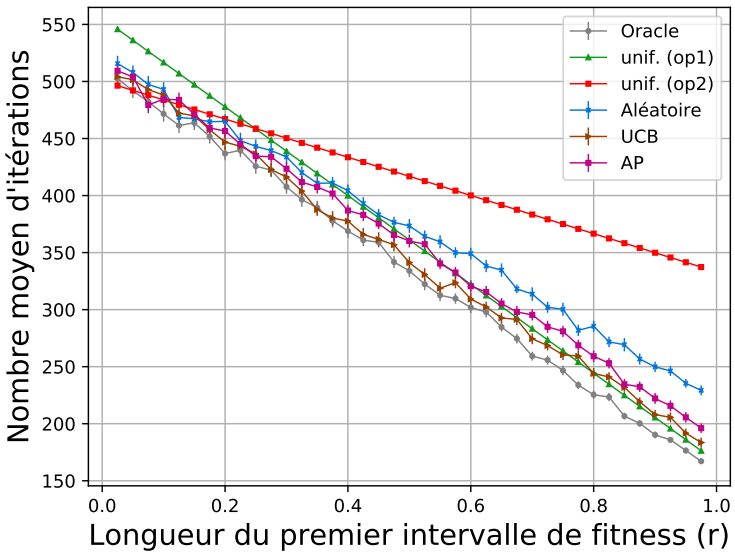
tivement 26, 87, and 263 évaluations. Beaucoup plus élevé que la différence entre l'oracle et la meilleure stratégie uniforme.

Stratégies adaptative *vs.* stratégies uniforme. La performance des stratégies adaptatives peut être pire qu'une stratégie uniforme. Par exemple, dans le cas 2, la stratégie UCB est meilleure que toute stratégie uniforme pour $r \in [0.11, 0.99]$. La longueur de l'intervalle de fitness où l'espérance de l'amélioration de op_2 est le meilleur est court ($r < 0.11$). Sur cet intervalle, la stratégie uniforme choisissant constamment op_2 est meilleure que la stratégie adaptative UCB. De même, pour $r > 0.99$, la stratégie uniforme choisissant op_1 est meilleur que la stratégie UCB. La performance des stratégies de sélection adaptative dépend de la longueur de l'intervalle où un opérateur domine l'autre, mais également des espérances de l'amélioration dans le second intervalle. Par exemple, dans le cas 3, lorsque les espérances de l'amélioration des algorithmes dans le portefeuille dans le second intervalle sont très faibles par rapport au premier, une stratégie uniforme est préférée. En effet, la stratégie UCB exige des 71 évaluations supplémentaires en moyenne par rapport à la stratégie uniforme. Dans ce cas, il n'y a que peu d'intérêt à changer d'opérateur, les performances résultantes principalement dans la sélection correcte sur le second intervalle exigeant un plus grand nombre d'évaluations.

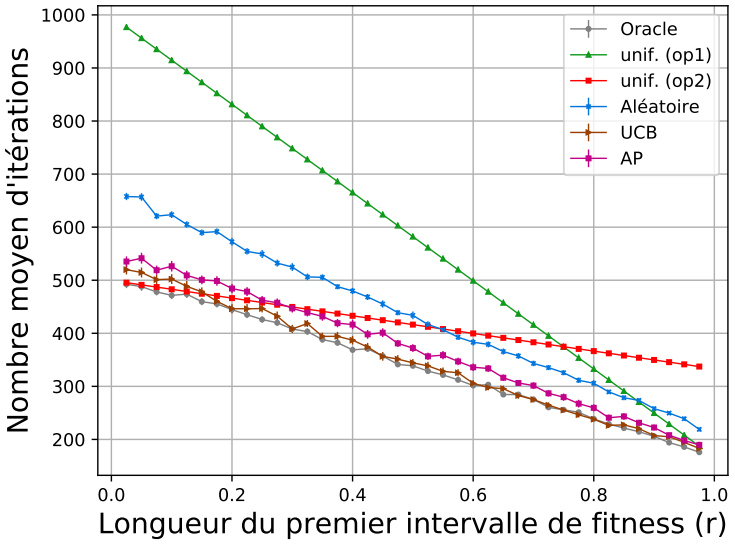
Stratégies adaptative *vs.* oracle. La performance des stratégies UCB et AP suit la tendance de la performance d'une stratégie oracle qui sélectionne le meilleur opérateur à chaque itération. L'écart moyen entre les stratégies oracle et UCB est de seulement 14, et de 30 entre les stratégies oracle et AP. Les pentes de la régression linéaire de performance en fonction de r des stratégies UCB et AP sont respectivement -0.36 et -0.37. Elles sont plus petites, mais dans le même ordre que la pente de la régression de performance de la stratégie oracle qui est égale à -0.33. La différence de l'espérance de l'amélioration entre opérateurs a plus d'impact sur la performance des méthodes adaptatives que la longueur r de l'intervalle de fitness où un opérateur est meilleur que l'autre.

Stratégies UCB *vs.* AP. Dans l'ensemble, les performances de la stratégie AP ne dépassent jamais celles de la stratégie UCB sauf dans 3 exceptions mineures pour les valeurs les plus faibles de r dans le cas 3. Dans le cas 1, la stratégie UCB est meilleure que la stratégie AP pour $r > 0.35$ et la différence entre ces stratégies est de 10 évaluations en moyenne, ce qui est la moitié de la différence entre la meilleure stratégie uniforme et la stratégie oracle. Dans le cas 2, la stratégie UCB est meilleure que AP, à l'exception des 3 plus grandes valeurs de r, tout en étant très proche de l'oracle : la différence moyenne avec la stratégie oracle pour la stratégie UCB est seulement de 8 évaluations par rapport aux 32 évaluations pour la stratégie AP, et 57 évaluations pour une stratégie uniforme. Dans le cas 3, la différence de performance entre les stratégies UCB et AP est beaucoup plus petite (seulement

Cas 1



Cas 2



Cas 3

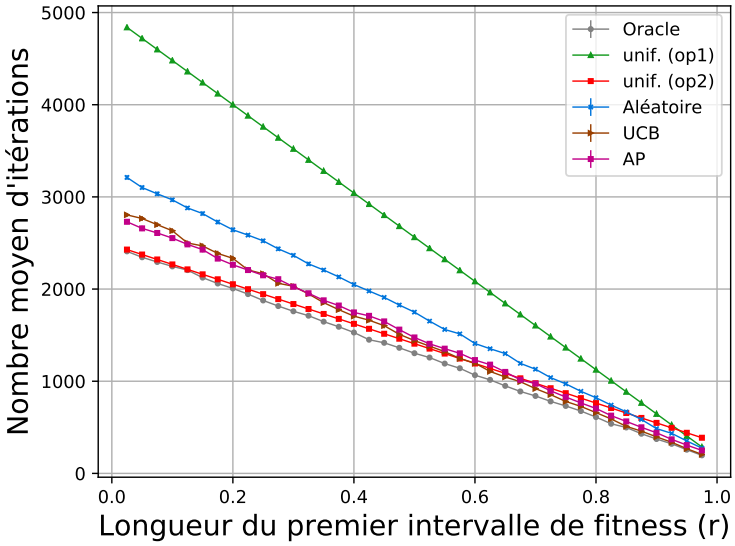


FIGURE 31 – Comparaison des stratégies de sélection avec les meilleurs paramètres pour UCB $C = 4$, and for AP $\alpha = 0.1$, $\beta = 0.1$. De haut en bas : cas 1, 2, et 3.

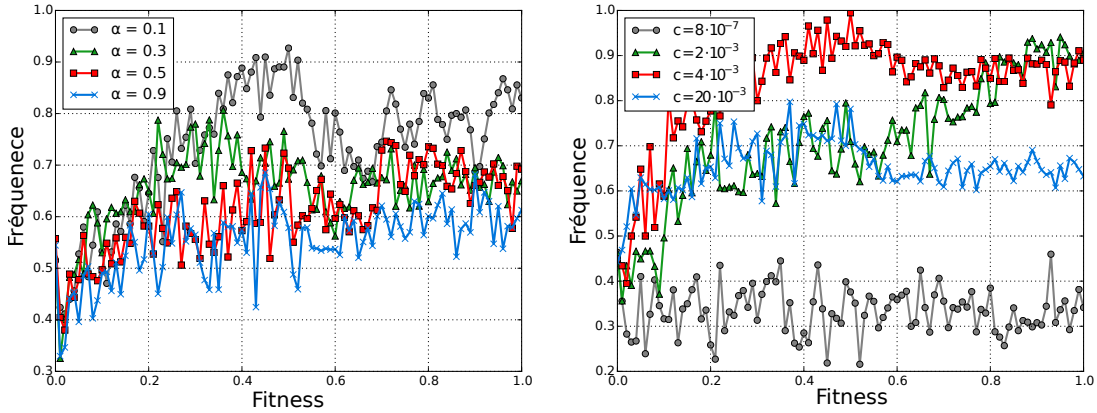


FIGURE 32 – Fréquence de la sélection du meilleur opérateur selon la valeur de fitness pour différents paramètres. Cas 2 avec $r = 0.5$ AP avec $\beta = 1$ (gauche) et UCB (droite).

15 évaluations en moyenne). Bien que UCB surpasse AP pour les valeurs r plus grandes que 0.37, les deux stratégies sont en moyenne pire que la stratégie uniforme optimale d'un facteur de 1.8 sur le nombre d'évaluations.

La figure 32 montre la fréquence de sélection du meilleur opérateur selon la valeur de fitness pour les stratégies UCB et AP dans le cas 2 et lorsque l'espérance de l'amélioration des opérateurs est modifiée à la valeur de fitness $r = 0.5$. La sélection d'opérateurs pour la stratégie UCB avec $C = 0.004$ converge vers la fitness 0.3, et pour AP avec $\alpha = 0.1$ autour de la fitness 0.5. Lorsque le meilleur opérateur change à la valeur de fitness $r = 0.5$, la stratégie UCB détecte plus rapidement le nouveau meilleur opérateur que la stratégie AP. UCB est capable de détecter plus efficacement le meilleur opérateur que AP même dans un environnement simple et non stationnaire.

Stratégies aléatoire *vs.* uniforme. À l'aide de ce scénario, nous avons montré plusieurs fois que la sélection aléatoire peut surpasser une méthode uniforme avec une sélection d'opérateur constante. Ce résultat rejoint des observation des travaux [TF13] et [Gar+14]. Ce scénario *c2pm2* avec deux intervalles de fitness nous aide à mieux comprendre pourquoi la sélection aléatoire peut être avantageuse. En effet, la stratégie aléatoire est meilleure que la meilleure stratégie uniforme lorsque la longueur r appartient aux intervalles $[0.14, 0.4]$, $[0.56, 0.88]$, et $[0.6, 1]$ respectivement pour les cas 1, 2, and 3. En résumé, une stratégie aléatoire est meilleure qu'une stratégie uniforme non adaptative lorsque les performances de chaque opérateur dans le portefeuille sont proches. Alors qu'une stratégie uniforme est plus efficace lorsque la qualité de l'un des deux opérateurs dépasse nettement l'autre, c'est-à-dire lorsque la longueur l'intervalle est courte.

Discussion. Ce scénario à deux intervalles de fitness offre la possibilité de régler finement les performances de chaque opérateur à deux étapes succes-

sives de la recherche. La comparaison des cas 2 et 3 montre que lorsque les espérances de l'amélioration à la seconde étape sont beaucoup plus faibles qu'à la première étape, une méthode adaptative devient moins efficace sauf lorsque la longueur de la première étape est très importante. En effet, le temps qui peut être gagné dans la première étape devient négligeable par rapport à la durée de la seconde étape car la principale difficulté se révèle alors être la convergence finale vers la valeur optimale. Lorsque l'échelle des espérances de l'amélioration entre les deux étapes est modérée comme dans le cas 2, une méthode adaptative comme la stratégie UCB est très efficace. Cependant, lorsque la différence de performance entre opérateurs à un stade de la recherche devient faible, comme dans le cas 1, le problème est tout aussi difficile pour tous les opérateurs du portefeuille, et l'intérêt d'une sélection adaptative devient faible, en plus du fait qu'il devient difficile de détecter l'opérateur le plus performant.

4.3.3 *Scénario ld2 : comparaison avec les problèmes académiques*

Dans ce scénario, nous avons deux opérateurs op_1 et op_2 dont la moyenne de la distribution de fitness après application d'un opérateur décroit linéairement avec la fitness. Pour rappel, avant le point d'intersection de coordonnées (fx, eix) dans le plan fitness *vs.* amélioration moyenne, lorsque l'espérance de l'amélioration initiale de l'opérateur $\mathrm{ei0}_{\mathrm{op2}}$ de l'opérateur op_2 est inférieur à celui $\mathrm{ei0}_{\mathrm{op}_1}$ de l'opérateur 1. L'opérateur 1 a une meilleure espérance de l'amélioration que l'opérateur 2 avant la valeur de fitness fx et inversement après cette valeur de fitness fx. Dans le cas inverse où $\mathrm{ei0}_{\mathrm{op}_1}$ est inférieur à $\mathrm{ei0}_{\mathrm{op}_2}$, les rôles des opérateurs 1 et 2 sont inversés.

TEMPS DE RÉSOLUTION DE LA STRATÉGIE AP ET LA STRATÉGIE DE RÉFÉRENCE. Dans ce scénario, nous définissons l'espérance de l'amélioration initiale de l'opérateur op_1 à $\mathrm{ei0}_{\mathrm{op}_1} = 2 \times 10^{-3}$ et le point d'intersection à $(\mathrm{fx} = 0.5, \mathrm{eix} = 0.001)$ (voir la figure 33 pour mémoire des paramètres). L'espérance de l'amélioration de l'opérateur op_2 est variable : $\mathrm{ei0}_{\mathrm{op}_2} \in [1 \times 10^{-3}; 5 \times 10^{-2}]$.

La figure 34 montre le temps de résolution moyen (ERT) en fonction de $\mathrm{ei0}_{\mathrm{op}_2}$. Au cours de l'optimisation, le temps passé avant l'intersection d'abscisse $\mathrm{fx} = 0.5$ est plus court qu'après l'intersection. En effet, pour les valeurs de fitness plus petite que fx, les espérances d'amélioration sont plus grandes. L'opérateur ayant la meilleur performance après fx sera celui qu'il faudra privilégier pour avoir une meilleure performance. Ainsi, lorsque $\mathrm{ei0}_{\mathrm{op}_2}$ est inférieur à $\mathrm{ei0}_{\mathrm{op}_1}$, la stratégie uniforme opérateur op_2 a un temps d'optimisation plus court que la stratégie uniforme opérateur op_1 ; et inversement, lorsque $\mathrm{ei0}_{\mathrm{op}_2}$ est supérieur à $\mathrm{ei0}_{\mathrm{op}_1}$, le temps d'optimisation avec l'opérateur 1 est plus court.

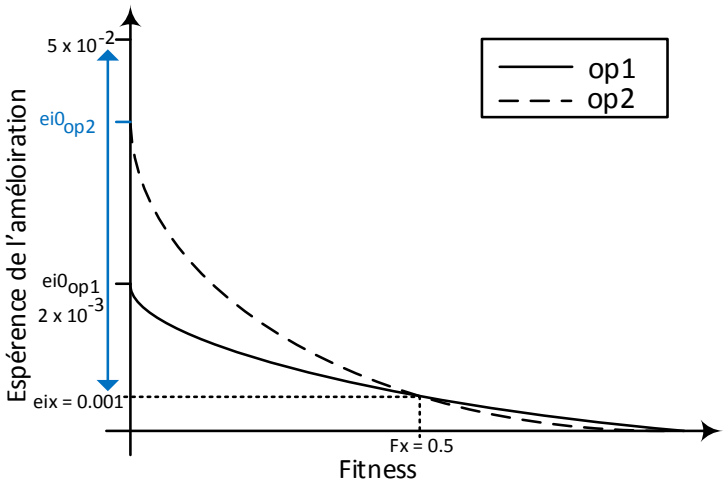


FIGURE 33 – Paramètre des deux opérateurs utilisé pour le scénario à deux opérateurs.

Dans cette étude qui étudie de nombreuses situation différente, la stratégie AP a été préférée à la stratégie UCB car elle offre de meilleure performance moyenne en utilisant la même valeur de méta-paramètre. En effet, les méta-paramètres de la stratégie AP sont plus robuste de ce point de vue que ceux de la stratégie UCB. La stratégie de sélection adaptative AP de méta-paramètres $\alpha = 0.4$ et $\beta = 0.5$ a de bonnes performances pour les valeurs de $ei0_{op_2}$ comprise entre 0.01 et 0.05 puisque le temps de résolution moyen est plus court que celui des deux stratégies de sélection uniforme. Par contre, lorsque les performances des deux opérateurs sont proches (autour des valeur de $ei0_{op_2}$ de 10^{-3}), les performances de la stratégie de sélection adaptative sont dégradées. Cette observation rejoint celle du cas 1 du scénario *c2pm2*.

CORRESPONDANCE ENTRE PROBLÈME D'OPTIMISATION ET LE MODÈLE

De façon prospective, nous proposons de faire une correspondance des problèmes d'optimisation académique et le scénario *ld2* du modèle Fitness Cloud. En effet, comme nous avons pu l'observer pour les problèmes onMax et NK (voir figure 2 et 16). Les moyennes de la distribution de fitness ainsi que l'espérance de l'amélioration s'approche de la forme des courbes obtenues avec le scénario *ld2*. L'idée est donc double. D'une part, de montrer qu'il est possible d'expliquer les performances des méthodes adaptatives sur les problèmes d'optimisation académiques à l'aide du scénario *ld2* du modèle FC, et d'autres part, de montrer qu'il serait possible de choisir une méthode adaptative ainsi que ces paramètres en « projetant » le problème académique dans le plan fitness *vs* espérance du gain, et d'en déduire les méta-paramètres des adaptatives à l'aide du scénario *ld2*.

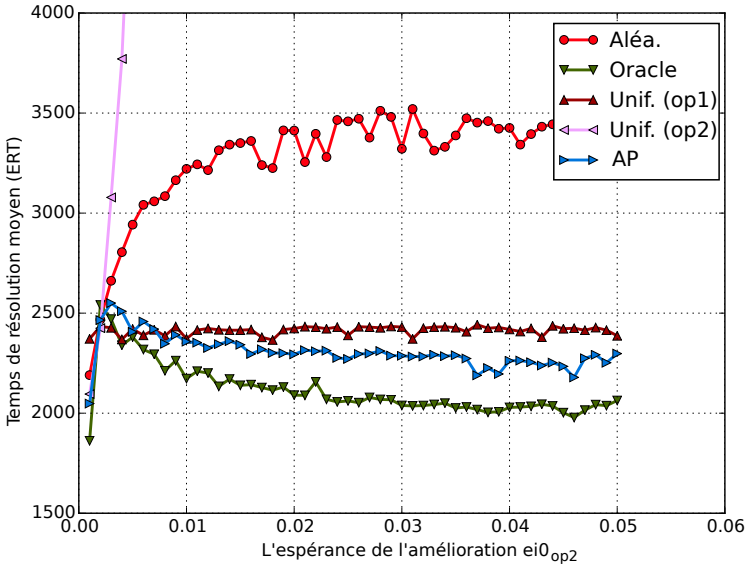


FIGURE 34 – Temps de résolution moyen (ERT) en fonction de l'espérance de l'amélioration initiale $ei0$ de l'opérateur 2. Comparaisons de la stratégie de sélection AP de paramètre $\alpha = 0.4$ et $\beta = 0.5$ et des stratégies de référence aléatoire (Aléa.), oracle, uniforme pour l'opérateur 1 (unif. op1), uniforme pour l'opérateur 2 (unif. op2).

Nous proposons donc de réaliser cette correspondance entre un scénario *ld2* du modèle FC et les deux problèmes étudiés auparavant, OneMax et le paysage NK, pour les opérateurs de mutation bit-flip de taux c/N avec différentes valeurs de c. Pour ce faire, nous nous sommes appuyé sur le fitness cloud empirique des opérateurs bit-flip (cf. figure 2 pour le OneMax et cf. figure 16 pour le paysage NK). L'opérateur 1 du modèle modélise l'opérateur bit-flip avec le taux $c/N = 1/N$. Les différents opérateurs 2 possibles sont modélise les autres taux de mutation c/N avec $c \in \{2, 4, 8, 16\}$. Le point d'intersection expérimental entre les opérateur sont normalisés de manière à ce que la fitness initiale soit 0 et la fitness optimale soit 1. Le tableau 5 résume les paramètres du modèle établissant la correspondance.

Tout d'abord, nous avons calculé la moyenne du temps de résolution pour les méthodes non-adaptatives uniforme avec l'opérateur 1 et avec l'opérateur 2, pour la stratégie de sélection adaptative AP pour différentes valeurs fx et $ei0_{op_2}$. Puis, nous avons situé dans ce plan fx *vs.* $ei0_{op_2}$, l'estimation des paramètres pour les problèmes OneMax et NK. La figure 35 montre l'ensemble des résultats obtenus. La couleur indique la différence du nombre d'itérations entre la meilleure stratégie uniforme et la stratégie AP.

Nous pouvons faire un parallèle entre les figures 34 et 35 par rapport aux paramètres du modèle étudié. Les deux figures portent le même type d'information représentée différemment. La figure 34 correspondant à la valeur

TABLE 5 – Valeur de d'intersection entre l'opérateur 1 (taux $c = 1$) et l'opérateur 2, et valeur initiale d'espérance d'amélioration de l'opérateur 2 pour différents couples d'opérateurs pour le problèmes OneMax et paysage NK utilisés pour comparer au modèle du scénario *ld2*.

Opérateur 2 bit-flip c/n	OneMax		NK	
	fx	eio	fx	eio
$c = 2$	0.7	0.00112	0.63	0.00102
$c = 4$	0.5	0.0016	0.5	0.0012
$c = 8$	0.4	0.0022	0.3	0.0013
$c = 16$	0.3	0.0032	0.25	0.00136

$fx = 0.5$ et la coloration correspondant à la différence de temps moyen d'optimisation rentre la meilleure stratégie uniforme et la stratégie AP. Quand la coloration est supérieure à 0, l'avantage va à la stratégie adaptative AP ; quand la coloration est inférieure à 0, une stratégie uniforme obtient de meilleure performance que la stratégie AP. Une stratégie adaptative AP tend à être une performant lorsque l'intersection entre espérances d'amélioration des opérateurs augmente (plus grande que $fx = 0.5$) et lorsque l'espérance d'amélioration initiale de l'opérateur 2 augmente. En effet, dans ces cas, l'opérateur 2 devient prépondérant sur l'opérateur 1 tardivement dans le processus d'optimisation et il est aussi facile de détecter la différence de performance entre opérateurs. Toutefois, pour une espérance d'amélioration de l'opérateur 2 faible proche de 0.001, quelque soit la position de l'intersection fx, une stratégie adaptative AP est moins performante que une stratégie uniforme optimale. De façon prospective pour les problèmes OneMax et NK, nous avons placé les caractéristiques des opérateurs bit-flip avec un taux de mutation c/n avec $c \in \{2, 4, 8, 16\}$ dans le plan des paramètres du modèle $(fx, ei0_2)$. Pour établir cette correspondance, nous avons normalisé linéairement les valeurs maximum et minimum des espérances d'amélioration entre 0 et 1 (voir tableau 5). Les tendances générales données par le modèle sont respectées. Lorsque l'opérateur 2 a un taux de mutation faible et plus proche de celui de l'opérateur 1, une stratégie adaptative semble profitable selon le modèle. Par ailleurs, toujours selon la position observée des opérateur dans ce plan, les portefeuilles pour le problème NK semblent moins favorables à l'utilisation d'une stratégie adaptative par rapport à une stratégie uniforme. Cette observation est en accord avec les expériences des chapitres précédents. Le modèle nous donne des pistes de recherches futures pour la compréhension et le choix d'une stratégie de réglage des paramètres avant ou après optimisation en fonction des caractéristiques des opérateurs du portefeuille.

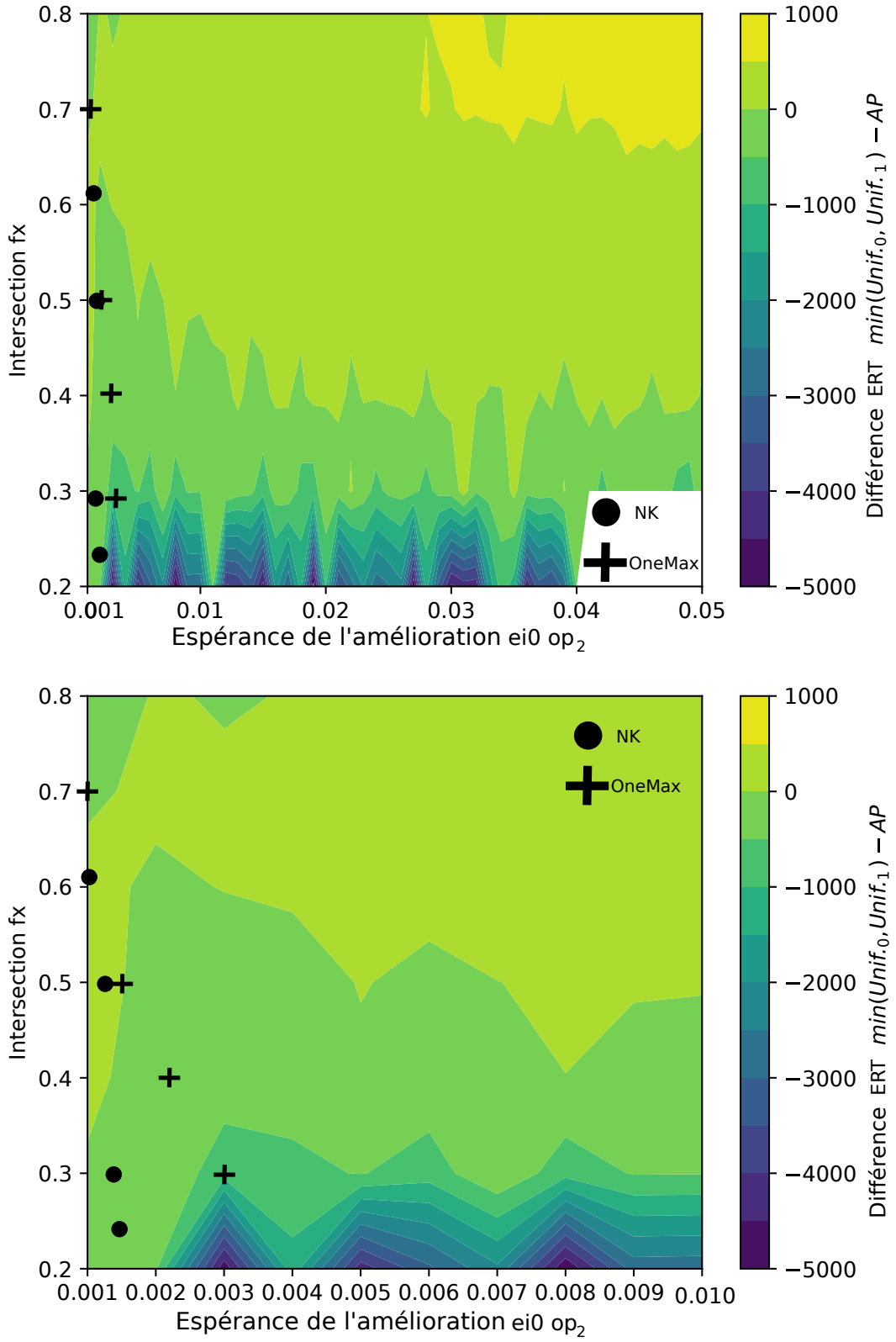


FIGURE 35 – Différence du temps d'optimisation entre la meilleure stratégie uniforme (qui utilise constamment le même opérateur) et la stratégie de sélection AP en fonction des paramètres fx, fitness du point d'intersection, et $ei_{0,op2}$, espérance de l'amélioration initiale de l'opérateur 2. Une valeur positive indique que la stratégie adaptative AP a un temps de résolution plus court que la meilleure stratégie uniforme. Différent couples d'opérateur bit-flip pour les problèmes OneMax et Nk ont aussi été placés dans ce plan (voir Tab. 5).

4.4 SYNTHÈSE

Le modèle fitness cloud détermine de façon précise l'espérance de l'amélioration d'un opérateur en fonction de l'état de la recherche. Malgré certaines limites, en particulier relatives à notion d'optima locaux, ce modèle permet de capturer un grand nombre de propriétés des espaces de recherche. Il donne un cadre formel d'expérimentation et d'analyse des stratégies de sélection d'opérateurs dans différents contextes. Ainsi, il permet d'identifier des contextes où le réglage des paramètres pendant optimisation (on-line) et plus pertinent que le réglage avant optimisation (off-line).

Plusieurs scénarios du modèle fitness cloud ont été établi en s'inspirant du cadre d'étude du chapitre3 qui étudie les stratégies de sélection sur les problèmes académiques OneMax et NK. Chaque scénario construit s'attache à étudier une caractéristique particulière qui progressivement permet de compléter la compréhension des stratégies de sélection.

Dans le scénario *c2*, le gain des opérateurs est invariant, quel que soit l'état de la recherche. Ce scénario permet d'étudier les capacités d'apprentissage des stratégies de sélection à sélectionner un opérateur quand celui-ci domine tous les autres pendant toute la recherche. Ce scénario a permis de comparer les performances de différentes stratégies, en particulier la stratégie basée sur l'UCB qui se révèle être très performante. De plus, nous avons pu montrer que toutes les méthodes sont sensibles à la différence de gain moyen entre opérateurs. Par ailleurs, le scénario *c2* nous a permis de mettre en évidence l'impact du nombre d'opérateurs sur le temps de convergence de la sélection adaptative vers l'opérateur optimal. Il est possible pour les stratégies adaptatives telle celle basée sur UCB de sélectionner un opérateur optimal parmi un grand nombre d'opérateurs. Le temps de convergence se dégrade alors linéaire avec le nombre d'opérateurs. Le scénario *c2pm2* ajoute la possibilité de changement de gain des opérateurs au cours de l'optimisation. Dans ce scénario, l'espace de recherche est divisé en deux intervalles de fitness sur lesquels le gain des opérateurs est constant, et pour lesquels l'ordre de dominance des opérateurs s'inverse entre les deux intervalles. Ce scénario montre qu'il est plus important que la sélection correcte s'effectue en fin d'optimisation lorsque la progression est lente. De plus, de nouveau, lorsque les différences de gain entre opérateurs sont notables, la stratégie basée sur UCB est performante. Enfin dans le scénario *ld2*, la décroissance de gain des opérateurs à sélectionner est continue et s'approche plus des problèmes académiques étudiés auparavant. Ce scénario permet de mieux comprendre pourquoi la sélection adaptative d'opérateurs en particulier dans les problèmes NK n'est que très efficace. En effet, une différence faible entre opérateurs et combinée à un gain faible en fin d'optimisation ne permet pas à une méthode adaptative de tirer parti d'un changement d'opérateurs.

Le modèle Fitness Cloud est un benchmark pertinent pour comprendre les méthodes adaptatives. L'analyse réalisée dans ce chapitre et le modèle fitness cloud peuvent être étendus. Premièrement, nous pouvons étendre cette analyse à un contexte d'optimisation distribuée. Il sera également intéressant d'étendre le modèle Fitness Cloud à l'optimisation multi-objectifs qui nécessite alors de définir la covariance des distributions de qualité de solutions selon toutes les paires de fonction objectif.

5

MODÈLE DE CALCUL MAÎTRE-ESCLAVE

Le modèle de calcul maître-esclave est caractérisé par une organisation différente par rapport au modèle en île. Le nœud maître prend l'ensemble des décisions pour les nœuds esclaves ; les nœuds esclaves exécutent à la suite les instructions. Autrement dit, le nœud maître a la possibilité d'avoir une vision globale de l'ensemble des tâches exécutées et en cours de réalisation. Les nœuds esclaves ont une vision individuelle, chacun possédant uniquement une vision locale sur la tâche qui lui est affecté. L'intérêt du modèle maître-esclave par rapport au modèle en île réside dans la facilité de l'implémentation et dans l'organisation simple des communications puisque seule une communication entre le maître et des esclaves sont permises. En outre, cette organisation des communications, bien que pouvant être relativement restrictives par rapport à un modèle en îles quant aux différentes possibilités que l'on peut tirer d'un système distribué en général. Elle donne un cadre alternatif pour penser les stratégies de sélection de façon simple tout en profitant du parallélisme offert par les ressources de calcul.

En effet, la résolution d'un problème d'optimisation à l'aide de ce modèle relativement simple de calcul, et dans le cadre de stratégie de sélection à portefeuille d'opérateurs, implique des choix de conception multiples qui peuvent s'avérer critiques pour la performance globale du système distribué ainsi considéré. Dans ce chapitre, nous proposons de concevoir et d'étudier en particulier les composants suivants : la proportion de nœuds devant être affectés par itération à chaque opérateur du portefeuille, la possibilité d'un traitement par lot des tâches (c'est-à-dire l'exécutions d'une liste opérateurs sélectionnés) par les esclaves, et l'adéquation du calcul de la récompense et l'impact du choix de la fonction d'agrégation. Nous discuterons en détail de ces composants et des défis sous-jacents par la suite (cf. section 5.1.1).

Nous utiliserons dans notre étude le modèle fitness cloud présenté dans le chapitre précédent. Nous mettons l'accent sur une sous-classe de notre modèle dans laquelle nous considérons un scénario à deux opérateurs avec une espérance d'amélioration fixe (cf. section 4.2.1). En particulier, ceci nous permettra de mieux étudier la pertinence des choix de conception effectués pour répondre aux questions suivantes :

1. Comment construire une stratégie de sélection performante au sein d'un environnement parallèle maître-esclave en fonction des caractéristiques du problème et du nombre de nœuds de calcul ?

2. La façon de considérer la récompense ayant une influence sur estimation du paramétrage ; comment calculer une récompense pour chaque paramétrage du portefeuille de façon pertinente ?
3. Comment réduire de façon effective les temps de communication ?
4. Quelle est la proportion optimale des opérateurs sur les nœuds de calcul ? En effet, avec un portefeuille de paramètres, le modèle maître esclave permet d'explorer le paramètre le plus approprié de façon plus explicite et/ou d'allouer des nœuds de calcul proportionnellement par rapport à l'intérêt du paramètre de façon assez flexible.

Dans la suite, nous commençons par décrire les différents composants algorithmiques nécessaires dans un tel modèle en discutant en détail les actions effectués par le nœud maître et les nœud esclaves, ainsi que les différents choix qui peuvent se présenter à nous. Dans un second temps, nous présentons une analyse expérimentale détaillée de l'impact de ces choix. D'abord, nous analysons l'impact du taux d'utilisation des opérateurs sur les nœuds esclaves. Ensuite, nous étudions l'impact du traitement par lot des tâches sur la capacité d'adaptation des nœuds esclaves et sur temps de résolution global.

5.1 CONCEPTION DES COMPOSANTS ALGORITHMIQUES POUR LE MAÎTRE ET LES ESCLAVES

5.1.1 *Description générale de l'algorithme adaptatif pour le modèle maître-esclave*

De façon générale, il est bien connu que le modèle maître-esclave est adapté pour la résolution de problèmes d'optimisation lorsque le temps nécessaire pour le calcul de la fonction de fitness est non-négligeable. À cet effet, le nœud maître affecte un ensemble de tâche aux nœuds esclaves qui consistent souvent à générer un nombre de nouvelles solutions et à calculer leurs fonctions de fitness. Dans le cadre de notre étude, ces solutions sont générées par différents opérateurs qui seront décidés par le maître. Nous devons donc décider de la stratégie de sélection au niveau du maître ainsi que de la gestion des récompenses relatives aux différents opérateurs exécutés par les esclaves. Le traitement de la récompense des opérateurs prend en particulier une place importante puisqu'elle permettra de guider le maître dans le choix des opérateurs à exécuter par les esclaves dans la prochaine itération à l'aide de la stratégie de sélection. Ce principe général est détaillé dans l'algorithme 4 s'appuyant sur une architecture maître-esclave synchrone et donnant une vue globale de son fonctionnement. L'algorithme 5 décrit quant à lui le code de haut niveau exécuté par le maître pour le choix des opérateurs ; et l'algorithme 6 décrit le code de haut niveau exécuté en parallèle par chaque nœud esclave.

Algorithm 4 Algorithme adaptatif maître-esclave pour le nœud maître

$(\Theta^1, \Theta^2, ..., \Theta^n) \leftarrow$Selection_Strategy_Initialization()
$x^\star \leftarrow$ Solution_Initialization()
$f^\star \leftarrow f(x^\star)$
repeat
  for each worker i **do**
    Send $\mathsf{Msg}(\Theta^i, x^\star, f^\star)$ to worker i
  end for
  Wait until all messages are received from all workers
  for each worker i **do**
    $(r^i, x^i, f^i) \leftarrow$ Receive $\mathsf{Msg}()$ from worker i
  end for
  $x^\star \leftarrow x^i$
  $f^\star \leftarrow f^i$ s.t. $f^i = \max\{f^\star, f^1, f^2, \ldots, f^n\}$
  $(\Theta^1, \Theta^2, ..., \Theta^n) \leftarrow$ Selection_Strategy$(r^1, ..., r^n)$
until stopping criterion is true

Algorithm 5 Algorithme de stratégie de sélection adaptative

function SELECTION_STRATEGY$(r^1, ..., r^n)$
  for each operator j **do**
    $R_j \leftarrow$ Global_Reward_Aggregation$(r^1, ..., r^n)$
  end for
  $(\Theta^1, \Theta^2, ..., \Theta^n) \leftarrow$ Decision_Strategy$(R_1, R_2, ..., R_k)$
  return $(\Theta^1, \Theta^2, ..., \Theta^n)$
end function

Algorithm 6 Algorithme adaptative maître-esclave pour le chaque nœud esclave

$(\Theta, x^\star, f^\star) \leftarrow$ Receive Msg() from master
for each index b of operators batch Θ **do**
  $x' \leftarrow$ Apply operator Θ_b on $x^\star$
  $f' \leftarrow$ Evaluate fitness of x'
  $\delta_b \leftarrow \max(0, f' - f^\star)$
  if $f(x^\star) < f(x')$ **then**
    $x^\star \leftarrow x'$
    $f^\star \leftarrow f'$
  end if
end for
for each operator j **do**
  $r_j \leftarrow$ Local_Reward_Aggregation(δ)
end for
Send $\mathsf{Msg}((r_1, r_2, \ldots, r_k), x^*, f^*)$ to master

L'algorithme 4 global fonctionne de façon itérative. À chaque itération, le nœud maître envoie la meilleure solution $x^\star$ et un lot d'identificateurs d'opé-

rateurs Θ^i à exécuter par chaque nœud esclave i. Nous reviendrons plus précisément sur l'utilité de cette notion de lot plus tard. Pour le moment, un lot est simplement une suite d'opérateurs représentant la séquence de tâches à exécuter par un esclave. Ainsi, à la réception d'un lot, un esclave exécute itérativement les opérateurs qui lui sont donc assignés en partant de $x^\star$ comme solution initiale. En exécutant le lot, chaque esclave calcule une nouvelle meilleure solution localement et de façon parallèle. Le nœud maître attend (de façon synchrone) la réception de l'ensemble des meilleures solutions locales calculées en parallèle par les esclaves, met à jour $x^\star$, recommence le processus d'envoi de nouveaux lots, et ainsi de suite jusqu'à une condition de terminaison.

Il est important de remarquer que chacun des nœuds esclaves envoie également au nœud maître la performance observée lors de l'exécution d'un lot d'opérateurs. En effet, dans le cadre de notre étude, les paramètres de sélection d'algorithmes, un portefeuille de k opérateurs, sont supposés être donnés et aucune connaissance *a priori* n'est prise en compte sur le comportement de ces opérateurs. L'aspect adaptatif est donc principalement traitée au niveau du nœud maître. Après avoir recueilli les récompenses calculées localement pour chaque nœud esclave, le nœud maître exécute la fonction principale *Selection_Strategy* (ligne 14 de l'algorithme 4). Cela lui permet de calculer le nouveau jeu de lots Θ^i renvoyé à chaque nœud esclave i. Comme mentionnée précédemment, le traitement par lot Θ^i est simplement une liste ordonnée d'opérateurs à exécuter séquentiellement par le nœud esclave i. Plus précisément, nous adoptons le cadre général d'un $(1+1)$-EA (voir l'algorithme 6). Par exemple, dans le cas où la taille du lot des opérateurs est de un, chaque nœud esclave exécute une seule évaluation de fitness ; ce qui est fondamentalement similaire au flot d'exécution d'un algorithme $(1+\lambda)$-EA parallèle synchrone où λ équivaut au nombre de esclaves n. Ainsi, le nombre d'itérations exécutées par chaque nœud esclave correspond à la taille du lot et l'opérateur de recherche appliquée à chaque itération correspond à l'ordre donné dans ce lot.

L'exécution d'un lot par un esclave fonctionne donc de façon itérative comme illustré dans l'algorithme 6. À chaque itération exécutée par un esclave, l'amélioration de la fitness δ_b de chaque opérateur appartenant au lot assigné est calculée. L'amélioration de la fitness [Fia+09] permet à chaque nœud esclave de calculer une récompense locale pour chaque opérateur exécuté. Ceci est codé dans la fonction *Local_Reward_Aggregation* (ligne 12 de l'algorithme 6) permettant au nœud esclave de calculer une récompense locale r_j pour chaque opérateur selon les améliorations observées δ. La récompense locale est déterminée pour tous les opérateurs exécutés au moins une fois par le nœud esclave. L'agrégation locale concerne donc tout opérateur appartenant au portefeuille et présent au moins une fois dans le lot reçu par l'esclave. Il est à noter que lorsque la taille du lot est de un élément, la ré-

compense locale est simplement égale à l'amélioration δ_1 pour l'opérateur considéré.

Pour résumer, à la fin du calcul correspondant à un lot, le nœud esclave renvoie au nœud maître un triplet contenant : (i) la meilleure solution trouvée par le nœud esclave, (ii) la valeur de fitness correspondante et (iii) les vecteurs de récompenses locaux r. La fonction principale *Selection_Strategy* exécutée sur le maître (ligne 14 de l'algorithme 4 et algorithme 5) est alors responsable de : (i) regrouper/agréger les récompenses locales envoyées par les esclaves et (ii) sélectionner le nouvel ensemble de lots d'opérateurs. Dans la suite nous commençons par décrire les stratégies de sélection adopté par le maître, puis nous revenons sur l'agrégation des récompenses par le maître. Pour plus de clarté, et sauf indication contraire, nous considérons le cas de lots unitaires, de taille 1, comme base à notre raisonnement ; c'est-à-dire qu'un esclave donnée exécute un seul opérateur. Ceci nous permet de détailler les stratégies de sélection adaptative de façon compréhensive et simple. Le contexte plus riche de lots non-unitaires est traité dans un second temps.

5.1.2 *Sélection adaptative homogène et hétérogène*

Le choix d'un lot d'opérateurs à affecter pour chaque nœud esclave relève exclusivement de la responsabilité du maître. Nous considérons deux classes de stratégies (voir figure 36) : (i) les stratégies adaptatives homogènes où le nœud maître calcule à chaque itération un opérateur afin de l'assigner à l'ensemble des nœuds esclaves, et (ii) les stratégies adaptatives hétérogènes où les opérateurs affectés aux nœuds esclaves à chaque itération peuvent être différents d'un esclave à un autre.

La stratégie homogène fonctionne de la manière suivante : dans une itération, le même opérateur est exécuté en parallèle par tous les nœuds esclaves. L'idée sous-jacente à cette stratégie est que à chaque itération un même opérateur permet d'avoir le maximum de bénéfice à la recherche. Ceci peut donc être vu comme une stratégie guidée par l'exploitation qui vise à éviter de perdre des évaluations en évitant de faire exécuter des opérateurs peu performants à une proportion d'esclave. En revanche, l'exploration d'opérateurs à priori peu efficace dans le passé est aussi nécessaire. Ceci est donc effectué de façon tout aussi homogène par tous les esclaves. Ainsi, la recherche peut être vue comme agissant par cycles entiers, où les esclaves exécutent exclusivement ou bien le même opérateur, dont la performance relative est la meilleure, ou bien l'opérateur le plus sous-exploré.

La stratégie hétérogène fonctionne de la manière suivante : à chaque itération seule une proportion de nœuds est affectée à un opérateur. L'idée sous-jacentes consiste à envisager que plus un opérateur est pertinent plus le nombre de nœuds utilisant l'opérateur doit être élevé. Ceci revient à dire qu'un mélange d'opérateurs différents devrait mieux servir la recherche

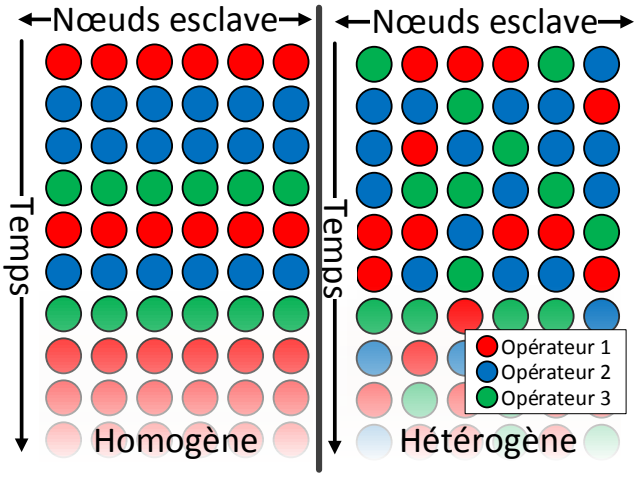


FIGURE 36 – À gauche, une stratégie de sélection de type homogène. L'ensemble des nœuds pour une itération exécute le même opérateur. À droite, une stratégie de sélection de type hétérogène, chaque nœud exécute un opérateur pouvant être différent.

qu'un ensemble contenant le même opérateur : la probabilité d'obtenir une meilleure solution lors de l'exécution de différents opérateurs à chaque itération est plus grande. Notons que les études antérieures dans le contexte du paramétrage ont considéré un modèle d'île hétérogène [TF13 ; Gar+14].

Dans le cadre d'une politique de choix homogène, nous considérons les trois stratégies suivantes : ϵ-greedy, AP et UCB. Le même opérateur calculé par l'une de ces stratégies est attribué par le nœud maître aux nœuds esclaves indépendamment du nombre de nœuds de calcul ou de la taille du lot. Le lot est traité de manière séquentielle, chaque opérateur est exécuté de manière synchrone sur une solution et une récompenses est calculée. Pour plus de détail sur le calcul de la récompense, voir la sous-section 5.1.3 ci-dessous.

Dans le cadre d'une politique de choix hétérogène, nous considérons les deux stratégies de sélection : ϵ-greedy et AP. En effet, nous rappelons que contrairement à UCB qui est déterministe, ces deux stratégies sont aléatoires. Pour ϵ-greedy, le meilleur opérateur est sélectionné avec un taux $1 - \epsilon$ et les autres avec un taux ϵ/k. Pour AP, chaque opérateur est sélectionné proportionnellement à un taux p_i. Ainsi, en exécutant itérativement une stratégie, les opérateurs sélectionnés par le maître sont susceptibles d'être différents à chaque exécution et le nœud maître peut donc affecter différents opérateurs aux nœuds esclaves, tout en se basant sur la même historique de récompenses.

Nous avons également conçu une nouvelle stratégie de sélection hétérogène appelée Δ-greedy. À la première itération, les opérateurs sont attribués au hasard aux nœuds esclaves comme pour les autres stratégies de sélection. Aux itérations suivantes, le nombre de nœuds esclaves associés au meilleur

opérateur est augmenté de Δ, où Δ est un paramètre entier. Le nombre de nœuds esclaves associés au plus mauvais opérateur est quant à lui diminué de Δ. Le meilleur (respectivement le pire) opérateur est calculé comme celui obtenant la plus haute (respectivement le plus basse) récompense locale à l'itération précédente. Afin d'éviter des effets de bord, le nombre de nœuds esclaves associé à chaque opérateur est limité à un nombre minimal n_{min}, constituant ainsi le second paramètre de cette stratégie. Dans le cadre du traitement par lots, pour chaque nœud esclave, la liste des opérateurs reste cependant homogène pour cette stratégie. Autrement dit, un nœud esclave exécute le même opérateur qui peut cependant être différent de celui exécuté par un autre nœud esclave.

5.1.3 *Agrégation des mesures*

L'agrégation des mesures de récompenses découle du fait qu'un opérateur peut être exécuté plusieurs fois par les nœuds de calcul esclaves. Aussi bien dans les cas d'une politique homogène ou hétérogène, chaque nœud à une récompense à envoyer au nœud maître après exécution d'un opérateur déterminé par la stratégie de sélection du nœud maître. La stratégie de sélection du nœud maître procède à l'agrégation globale des récompenses reçues. Quand la taille du lot est non-unitaire, le nœud local exécute de façon séquentielle chaque opérateur du lot envoyer par le nœud maître. Afin de réduire les communications, les nœuds esclaves n'envoient pas l'ensemble des récompenses calcul localement. Il procède à une agrégation locale des récompenses et envoie au nœud maître les récompenses pour chaque opérateur du portefeuille. Deux types d'agrégation sont étudiés dans la suite : la moyenne des espérances de l'amélioration pour chaque opérateur et le maximum des espérances de l'amélioration pour chaque opérateur.

Notons que l'espérance de l'amélioration de la fitness après application d'un opérateur est donnée par une distribution de probabilité. Cette distribution est bien entendu inconnus à priori à l'algorithme de recherche. Dans ce contexte, la moyenne des valeurs des récompenses calculées par les n nœuds esclaves permet d'estimer l'espérance de cette distribution avec une grande précision, alors que le maximum donne des informations sur ses extrêmes [Fia+09]. De la même façon que dans les chapitres précédents, nous considérons une fenêtre glissante de taille W pour estimer la récompense attendue $\hat{\mu}_i$ dans les stratégies ϵ-Greedy et UCB et pour estimer la moyenne pondérée exponentielle récente dans la stratégie AP.

5.2 ANALYSE EXPÉRIMENTALE : HOMOGÈNE-HÉTÉROGÈNE

Globalement, nous considérons donc les quatre stratégies de sélections adaptatives : ϵ-greedy, UCB, AP, et Δ-greedy. En se basant sur le modèle

TABLE 6 – Paramétrage des stratégies de sélection

Stratégies de sélection	Valeur des paramètres	
UCB Max.	$C = 0.005$	$W = 700$
UCB Mean	$C = 0.05$	$W = 5000$
AP	$\alpha = 0.2$	$\beta = 0.2$
ϵ-greedy He. Max.	$\epsilon = 0.5$	$W = 400$
ϵ-greedy He. Mean	$\epsilon = 0.5$	$W = 400$
ϵ-greedy Ho. Max	$\epsilon = 0.05$	$W = 4500$
ϵ-greedy Ho. Mean	$\epsilon = 0.05$	$W = 4500$
Δ-greedy	$\Delta = 10$	$n_{min} = 1$

fitness cloud, ces stratégies sont déclinées dans différentes situations aussi bien avec des politiques homogène que hétérogène, et en utilisant les deux fonctions d'agrégation des récompenses (moyenne et maximum).

En outre, nous considérons deux autres stratégies servant de référence : la stratégie aléatoire et la stratégie uniforme. La stratégie aléatoire sélectionne un opérateur de manière aléatoire alors que la stratégie uniforme sélectionne le même opérateur à chaque itération. Dans le cadre homogène avec une stratégie uniforme, l'ensemble des nœuds esclave et pendant l'ensemble de l'optimisation, l'opérateur utilisé ne change pas. Pour la stratégie aléatoire dans le cadre hétérogène, la stratégie sélectionne aléatoirement indépendamment un opérateur pour chaque nœud à chaque itération. Dans le cadre homogène, l'ensemble des nœuds de calcul exécute un opérateur sélectionné aléatoirement et la stratégie uniforme utilise le même opérateur, quels que soient le nœud et l'itération. Dans ce qui suit, nous discutons de la performance relative globale, puis fournissons une analyse plus ciblée pour mieux comprendre le comportement et la dynamique des différentes variantes ainsi étudiées. Sauf indication contraire, un lot de taille 1 sera considéré. Le cas d'un lot de plus grande taille est étudié plus en détail par la suite.

5.2.1 *Performance relative globale*

Nous adoptons une approche basée sur la simulation où nous comptons le nombre d'itérations effectuées par le nœud maître pour atteindre la valeur de fitness optimale. Cela nous permet de faire abstraction des problèmes de communication et d'évaluer avec précision le pouvoir d'adaptation des stratégie considérées. Il est cependant à noter que des considérations liées aux aspects communications seront discutées plus tard dans ce chapitre. Suivant le modèle Fitness Cloud (cf chaptire 4), nous considérons un portefeuille avec deux opérateurs où le premier suit la distribution normale $\mathcal{N}(-10^{-4}, 10^{-4})$,

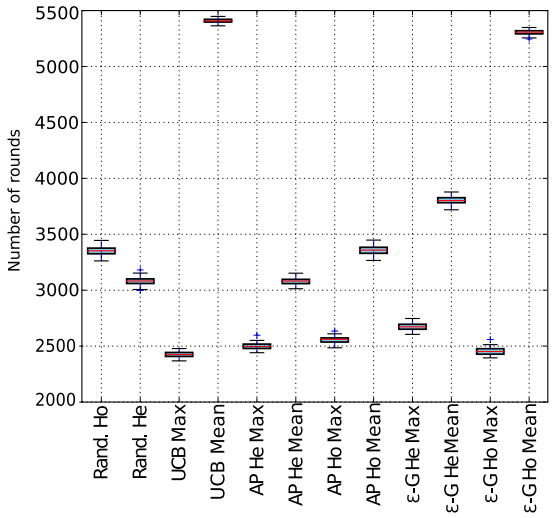


FIGURE 37 – Nombre d'itérations pour atteindre la fitness optimale en utilisant l'opérateur $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$, l'opérateur $2 \sim \mathcal{N}(-10^{-3}, 5 \times 10^{-4})$ avec $n = 256$ nœuds esclaves.

et le second $\mathcal{N}(-10^{-3}, 5 \times 10^{-4})$. Le choix d'une telle instantiation sera discuté dans ce qui suit. L'ensemble des paramètres des différentes stratégies de sélection est donné dans le tableau 6. Ces réglages choisis grace à l'étude des chapitres précédents se révèlent en effet robuste. Chaque variante est exécutée 100 fois et un aperçu de la performance relative en terme du nombre d'itérations est donné dans la figure 37.

Trois observations principales peuvent être relevées à partir de la figure 37. Premièrement, l'utilisation du maximum comme fonction d'agrégation des récompenses est clairement meilleure que l'utilisation de la moyenne. Deuxièmement, la différence entre un réglage homogène et hétérogène est atténuée et dépend de la stratégie de sélection elle-même. Troisièmement, selon le test statistique de Wilcoxon-Mann-Whitney avec un niveau de confiance de 5%, et en comparant le meilleur réglage des variantes de sélection, la stratégie UCB est la meilleure, suivie par la stratégie Δ-greedy et ϵ-greedy qui partagent la même performance et sont meilleures que la stratégie AP. Il est cependant à noter que toutes les stratégies d'adaptation partagent une performance relativement bonne lorsque les autres composants de conception, que sont le choix de la récompense et le type d'hétérogénéité, sont bien réglés.

5.2.2 *Analyse des fonctions d'agrégation de la récompense*

Pour comprendre la différence fondamentale entre utiliser le maximum ou la moyenne comme fonction d'agrégation de récompenses, ainsi que son importance lors de la conception d'une stratégie d'adaptation, nous étudions

cette propriété à l'aide du modèle Fitness Cloud dans un environnement étendu. Plus précisément, nous fixons les paramètres de la distribution normale correspondant au premier opérateur du portefeuille à $\mu_1 = -10^{-4}$ et $\sigma_1^2 = 10^{-4}$. Fixons également la moyenne de la distribution normale du second opérateur à $\mu_2 = -10^{-3}$ et étudions le comportement du système pour différentes valeurs de la variance σ_2^2. Pour un nombre fixe de nœuds esclave, et puisque les paramètres de la loi normale ne changent pas au cours de l'optimisation, le gain attendu en appliquant l'opérateur 1 est toujours le même (par exemple, en moyenne 8.33×10^{-2}, cf. section 4.2.1), et correspond à la récompense locale. Nous examinons maintenant comment la valeur relative de la récompense serait pour opérateur 2 si sa variance était définie pour prendre des valeurs différentes que celles de notre configuration initiale. Ceci est résumé dans la figure 38 qui montre les avantages attendus des deux opérateurs lors de l'utilisation d'une fonction d'agrégation moyenne (à droite) et d'une fonction d'agrégation maximum (à gauche), pour $n = 256$ nœuds esclaves en fonction de la variance σ^2 pour l'opérateur 2.

Dans la figure 38, nous étudions l'évolution relative de la récompense locale moyenne et de la récompense locale maximum. Selon la fonction d'agrégation, les courbes d'évolutions de la récompense pour l'opérateur 1 et l'opérateur 2 s'intersectent au point a pour la valeur $\sigma^2 = 4.17 \times 10^{-4}$ lorsque la moyenne est utilisée et au point b pour la valeur $\sigma^2 = 5.6 \times 10^{-4}$ lorsque le maximum est utilisé. Il est à noter que la valeur de la récompenses donne une indication sur quel opérateur sera à priori préféré lors de l'exécution effective de la stratégie de sélection adaptative. Trois scénarios sont donc à examiner de plus près. Avant le point a les deux fonctions d'agrégation donnent l'opérateur 1 gagnant de façon consistante. Après le point d'intersection b, l'opérateur 2 étant donnée gagnant par les deux fonctions d'agrégation. Dans ces deux situations, et même si la différence des valeurs des gains relatifs des les opérateurs sont différents, les rangs respectifs des deux opérateurs selon la fonction d'agrégation maximum ou moyenne sont similaires. Autrement dit, une stratégie de sélection d'opérateurs élitiste qui sélectionne l'opérateur selon la valeur de récompense la plus élevé sélectionnerait le même opérateur, quelle que soit la fonction d'agrégation utilisée. Cependant, la situation change lorsque la valeur de la variance σ_2 se situent dans l'intervale $[a, b]$. En effet, dans ce cas, les opérateurs sont tour à tour donnés gagnant et perdant selon la fonction d'aggregation utilisée.

De plus, la valeur moyenne de la récompense ne dépend pas du nombre de nœuds esclave. En effet, l'augmentation du nombre de récompenses locales calculées par les nœuds esclave réduit simplement l'intervalle de confiance autour de la valeur globale de la récompense moyenne. En revanche, la valeur maximale de la récompense augmente logarithmiquement avec le nombre de nœuds esclaves. Ceci est montré dans la figure 39 où trois exemples représentatifs sont considérés ($\sigma^2 = 3 \times 10^{-4} < a$, $\sigma^2 = 5 \times 10^{-4} \in [a, b]$, et

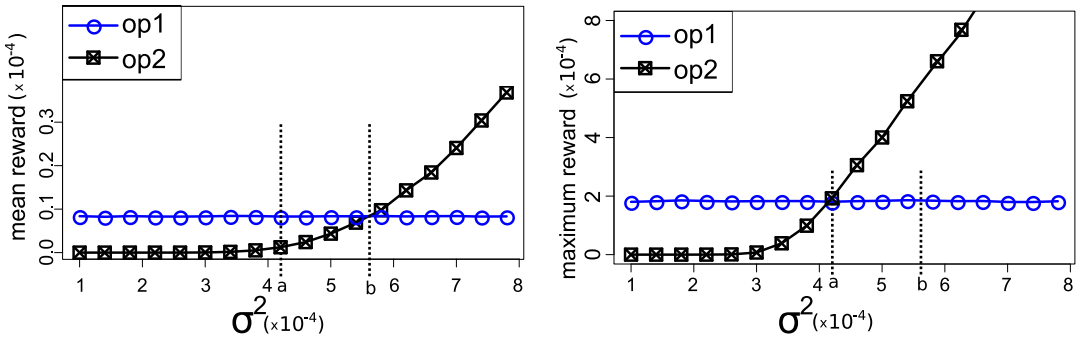
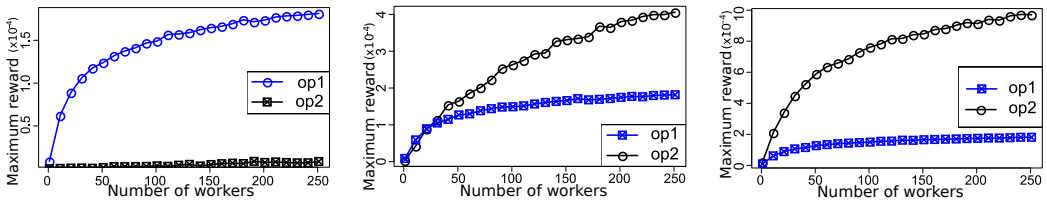


FIGURE 38 – À gauche la récompense moyenne, et à droite la récompense maximum obtenues avec les valeurs pour l'opérateur $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$ et l'opérateur $2 \sim \mathcal{N}(-10^{-3}, \sigma^2)$ et $n = 256$ de nœuds en fonction de la variance σ^2.



(a) (b) (c)

FIGURE 39 – La récompense maximum en fonction du nombre de nœuds pour l'opérateur $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$ et d'écart-type différent pour l'opérateur $2 \sim \mathcal{N}(-10^{-3}, \sigma^2)$: $\sigma^2 = 3 \times 10^{-4}$ (a), $\sigma^2 = 5 \times 10^{-4}$ (b), et $\sigma^2 = 7 \times 10^{-4}$ (c).

$\sigma^2 = 7 \times 10^{-4} > b$). Lorsque la variance est petite ou grande, le nombre de nœuds esclave ne change pas le rang de chaque opérateur par rapport à la valeur maximale de la récompense. Cependant, lorsque la variance σ^2 est entre a et b, le meilleur opérateur selon la récompense maximum change : pour un faible nombre de nœuds esclaves, l'opérateur 1 a une récompense plus élevée, alors que l'opérateur 2 est préféré lorsque $n \geqslant 30$.

Ces observations empiriques expliquent pourquoi la fonction d'agrégation qui considère le maximum des récompenses locale a permis d'avoir de meilleurs résultats par rapport à l'utilisation de la moyenne dans notre configuration initiale du modèle fitness cloud (cf. figure 37). En effet, nous avons fixé la variance de l'opérateur 2 à la valeur $5 \times 10^{-4} \in [a, b]$ correspondant ainsi à un scénario difficile pour la sélection adaptative.

Notons aussi que la récompense moyenne ne peut mesurer que l'espérance de l'amélioration d'un opérateur lorsqu'il est exécuté localement et indépendamment par chaque esclave individuel. La récompense maximale quant à elle mesure l'espérance de l'amélioration de la prochaine solution qui serait obtenue de façon coopérative par le système coopératif qui ne re-

tient en effet que la meilleure solution trouvée à travers tous les esclaves. À cet effet, une stratégie de sélection distribuée adéquate doit être en mesure d'acquérir et de prendre en compte les informations sur la qualité d'un opérateur lorsqu'il est exécuté en coopération par toutes les entités du système, et non seulement sur la qualité d'un opérateur pris indépendamment de l'environnement distribué et coopératif dans lequel il peut être exécuté. Par conséquent, l'agrégation de la récompense maximale doit être préférée lorsque l'objectif de l'algorithme adaptatif en mode maître-esclave est d'augmenter autant que possible la valeur de fitness dans chaque cycle de calcul, ce qui est généralement le cas d'un algorithme de type $(1+\lambda)$-EA. Plus généralement, nous soutenons qu'il devrait être possible d'étendre ce genre de résultats dans d'autres contexte d'algorithmes d'optimisation adaptative, où par exemple, une récompense globale adéquate doit tenir compte explicitement d'une mesure de diversité supplémentaire.

5.2.3 *Impact de l'hétérogénéité*

L'impact d'une politique homogène ou hétérogène lors du choix des opérateurs à exécuter par les esclaves peut elle aussi être étudié à la lumière de la différences entre la variance des opérateurs du portefeuille. Considérons par exemple un scénario où la stratégie de récompense maximale est utilisée ; puisque car elle s'est avérée être la meilleure. Considérons alors le modèle Fitness Cloud où l'opérateur 1 suit $\mathcal{N}(-10^{-4}, 10^{-4})$ et l'opérateur 2 suit $\mathcal{N}(-10^{-3}, \sigma^2)$. En faisant varier σ^2, et pour un nombre fixé de 256 nœuds esclaves, nous allons calculer et examiner de la même façon que précédemment la récompense globale maximale, c'est-à-dire l'amélioration attendue d'une itération de l'algorithme maître-esclave. Cependant, nous distinguons cette fois le contexte d'une exécution hétérogène dans laquelle les esclaves sont divisés en deux catégories ; ceux exécutant l'opérateur 1 et ceux exécutant l'opérateur 2. En faisant ainsi varier la proportion de nœuds esclaves hétérogènes, on peut alors calculer le nombre optimal n_1 d'esclaves qui devraient exécuter l'opérateur 1 ($n - n_1$ exécutent l'opérateur 2), en fonction de la variance σ^2 de l'opérateur 2.

Plus précisément, pour chaque valeur $n_1 \in [0, n]$, la moyenne de la récompense maximale sur 1000 itérations indépendantes est calculée et la valeur n_1 avec la récompense maximale la plus élevée est sélectionnée. Ceci est résumé dans la figure 40. Clairement, pour un large intervalle de valeur de σ^2, la valeur optimale n_1 est soit 256 soit 0. Ceci indique qu'un réglage homogène (avec seulement l'opérateur 1 ou l'opérateur 2) est optimal, sauf pour un petit intervalle de valeurs de la variance (entre 3.4×10^{-4} et 4.4×10^{-4}). De plus, lorsqu'une stratégie hétérogène est avérée être optimale, le gain en terme de récompense maximum en comparaison à une stratégie homogène est très faible (cf. figure 40 à droite). Compte tenu de ces observations, nous

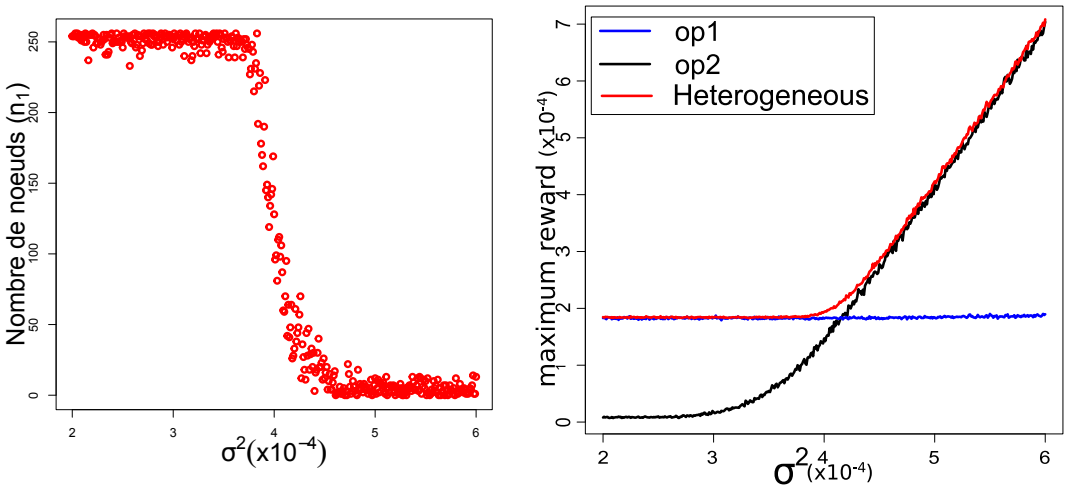


FIGURE 40 – Nombre optimal de nœuds n_1 avec l'opérateur 1 tel qu'il maximise la valeur de récompense maximum pour une stratégie de sélection hétérogène en fonction du paramètre d'écart-type σ^2. L'opérateur 1 suit la loi normale $\mathcal{N}(-10^{-4}, 10^{-4})$ et l'opérateur 2 suit la loi normale $\mathcal{N}(-10^{-3}, \sigma^2)$ pour 256 nœuds. À gauche le nombre optimal n_1 de nœuds. À droite, la valeur de la récompense maximale pour une stratégie de sélection homogène avec l'opérateur 1 (courbe bleue), l'opérateur 2 (courbe noire) et la stratégie de sélection hétérogène optimale (courbe rouge).

pouvons mieux comprendre l'impact de l'hétérogénéité sur la performance relative observée pour les différentes stratégies de la figure 37 pour laquelle la variance a été fixé à $\sigma = 5 \times 10^{-4}$.

5.3 ANALYSE EXPÉRIMENTALE : TRAITEMENT PAR LOTS

Jusqu'à maintenant, nous avons considéré un lot de taille 1. Une technique de planification par lots de taille plus grande peut cependant être utilisée. Ceci peut en effet s'avérer primordial lorsque le coût relatif aux communications entre maître et esclaves est significativement supérieur par rapport aux coûts des calculs effectués localement par les esclaves. Ceci est l'objet de l'analyse menée dans la suite de ce chapitre.

Toujours suivant le modèle Fitness Cloud, nous considérons un portefeuille avec deux opérateurs dont la distribution de fitness suit une distribution de loi normale. Comme précédemment, la valeur moyenne pour le premier opérateur op_1 est choisie pour être supérieure à celle du second opérateur op_2, mais la variance de op_1 est plus petite que celle de op_2. Rappelons que dans ce cas, lorsqu'une solution est créée par un opérateur, l'es-

TABLE 7 – Paramètres utilisés pour les stratégies de sélection

Stratégies de sélection	Méta-paramètre
UCB	$c = 0.005$, $w = 700$
AP	$\alpha = 0.2$, $\beta = 0.2$
ϵ-Greedy	$\epsilon = 0.05$, $w = 4500$
Δ-Greedy	$\Delta = 10$

Opérateurs	μ	σ
op_1	-1×10^{-4}	1×10^{-4}
op_2	-10×10^{-4}	5×10^{-4}

pérance de l'amélioration du premier opérateur op_1 est supérieure à celui de l'opérateur op_2. De cette façon, le premier opérateur serait préféré pour un algorithme séquentiel produisant une nouvelle solution itérative. Cependant, lorsqu'un nombre relativement grand de solutions sont créées simultanément en parallèle, le gain du second opérateur op_2 exécuté par l'ensemble des nœuds devient progressivement plus grand. Ce scénario nous permet donc d'analyser avec pertinence et précision l'impact du nombre de nœuds esclaves et la taille du lot envoyés à chaque nœud esclave. Le tableau 7 donne les valeurs des paramètres de la distribution de la loi normale utilisée dans ce travail.

Nous rappelons que dans le cadre d'un traitement par lot, nous utilisons les stratégies de sélection précédemment ennoncées AP, UCB, ϵ-Greedy et Δ-Greedy, avec deux adaptations majeurs. D'abord, les fonctions locales et globales d'agrégation (voir l'algorithme 4 et 6) sont utilisées pour calculer le maximum de l'amélioration de fitness obtenu, respectivement, par le lot d'opérateurs et par l'ensemble des nœuds esclaves. Deuxièmement, la stratégie de sélection choisit un ensemble d'opérateurs au lieu d'un seul opérateur à chaque itération, constituant ainsi un lot à communiquer à chaque esclave. La stratégie UCB choisit de façon déterministe l'opérateur ayant le score le plus élevé. L'ensemble des opérateurs sélectionnés est donc homogène ; à la fois à l'intérieur de chaque lot et entre les différents lots envoyés aux nœuds esclaves. Les autres stratégies de sélection (AP, ϵ-Greedy, et Δ-Greedy) définissent une probabilité de sélection pour chaque opérateur. Ainsi, pour tout opérateur et tout lot correspondant à un nœud esclave, l'opérateur est sélectionné en fonction de cette probabilité. Les lots d'opérateurs ainsi calculés par le maître sont hétérogènes au niveau des nœuds esclaves et au niveau du lot.

L'ensemble des méta-paramètres des différentes stratégies de sélection est donné dans le tableau 7. Pour toutes les expériences, les algorithmes sont

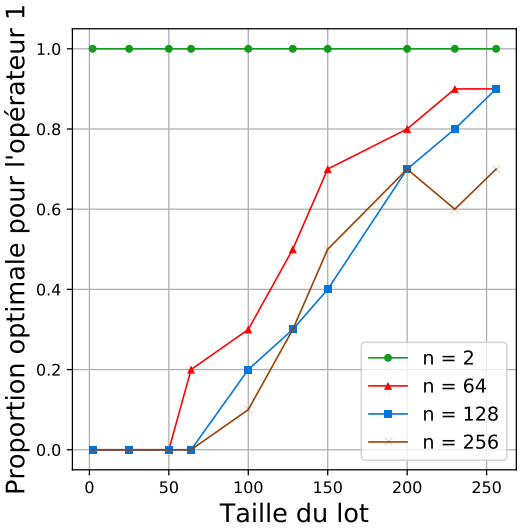


FIGURE 41 – Valeur de $p^\star$ de la probabilité optimale de sélection de l'opérateur 1 dans une stratégie aléatoire biaisée en fonction de la taille du lot et du nombre n de nœuds esclaves.

répétés indépendamment 32 fois et les valeurs moyennes de performance (le nombre d'itération ou effort de calcul) sont calculées.

5.3.1 *Stratégies de sélection de référence*

Les opérateurs op_1 et op_2 sont stationnaires dans le sens où les paramètres de la distribution de fitness des opérateurs sont constants tout au long le processus d'optimisation. Dans ce cas, en suivant les résultats illustrés à la figure 40, nous introduisons une stratégie de sélection aléatoire biaisée avec un taux p compris entre 0 et 1, l'opérateur op_1 est sélectionné, et avec un taux $1-p$ l'opérateur op_2 est sélectionné. Cette stratégie naïve aide à comprendre la proportion de nœuds affectée à un opérateur en fonction de la taille des lots et du nombre des nœuds esclaves. Nous étudions toujours également les stratégies de sélection uniformes qui choisissent l'un des deux opérateurs tout au long du processus.

Dans la figure 41, nous montrons la valeur optimale de $p^\star$ de sélection aléatoire de l'opérateur 1 donnant la meilleure performance en fonction de la taille du lot et du nombre de nœuds de calcul. Cette valeur est calculée avant l'optimisation en testant les valeur de p dans l'ensemble $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Nous pouvons noter que lorsque la taille du lot est faible, le taux optimal $p^\star = 0$ et le second opérateur est toujours sélectionné. Nous remarquons aussi que le taux optimal $p^\star$ augmente la part d'utilisation de l'opérateur 1 avec la taille du lot. Pour rappel, nous avons choisi une situation où l'opérateur 1 a une meilleure récompense avec l'agrégation par la moyenne alors

que l'opérateur 2 à une meilleure récompense avec l'agrégation par le maximum. Lorsque le nombre de nœuds esclaves est égal à la taille du lot (128), la sélection aléatoire biaisée optimale assigne 70% des nœuds de calcul à l'opérateur 2 (30% à l'opérateur 1). Enfin, la valeur optimum $p^\star$ est plus grande que 80% lorsque la taille du lot est importante par rapport au nombre de nœuds esclaves (la taille du lot est le double du nombre de nœuds esclaves).

5.3.2 *Performance adaptative avec une technique de planification par lot*

De toute évidence, plus la taille d'un lot est grande plus le rapport entre le temps de communication et le coût nécessaire au traitement d'un lot par un esclave est réduit. Ainsi, une technique de planification par lot permet de réduire le coût relatif de communication si le contenu des lots est pertinent. Dans cette section, l'objectif n'est pas d'analyser la performance d'une telle technique dans une perspective purement parallèle, mais plutôt de mettre en évidence l'exactitude d'une stratégie d'adaptation pour sélectionner les opérateurs contenus dans un lot. Ainsi, nous nous appuyons comme précédemment sur une étude empirique à base de simulation. Cependant, au lieu d'analyser le nombre d'itérations jusqu'à atteindre l'optimum de fitness, nous étudions le nombre d'évaluations exécutées en parallèle par tous les nœuds esclaves. À cet effet, le nombre d'évaluations peut être étudié en fonction du nombre de nœud esclaves et en fonction de la taille d'un lot. Cependant, nous présentons les résultats en normalisant par le nombre d'évaluations de la stratégie de sélection aléatoire non-biaisée avec $p = 0.5$. Cela facile la présentation des résultats pour des raisons d'échelle de valeurs, mais surtout cela permet d'avoir le gain de chaque stratégie par rapport à une stratégie de sélection naïve. Il est à noter qu'une stratégie avec une valeur normalisée inférieure à 1 est meilleure que la stratégie aléatoire non-biaisée, et inversement. Par soucis d'exhaustivité, nous incluons les stratégies uniforme qui sélectionnent toujours le même opérateur à chaque itération.

La figure 42 donne les performances relatives des différentes stratégies de sélection adaptative considérées dans nos expérimentations. La performance relative de toutes les stratégies adaptatives diminue avec la taille des lots. Les stratégies d'adaptation sont en fait meilleures que la stratégie aléatoire non-biaisée pour une taille relativement petite des lots, mais elles deviennent moins performantes pour des lots de grande taille. Pour une taille de lot fixée, la performance des stratégies d'adaptation semble cependant augmenter avec le nombre de nœuds esclaves utilisés. Les stratégies UCB et ϵ-greedy ne sont pas en mesure de sélectionner l'opérateur optimal à chaque itération lorsque le nombre de nœuds esclave est inférieur à 50 (et la taille du lot 128). Dans ce cas, la stratégie AP hétérogène est meilleure que les autres stratégies et peut être mieux qu'une stratégie homogène optimale. Lorsque la taille du lot est grande par rapport au nombre de nœuds esclave, les stratégies adapta-

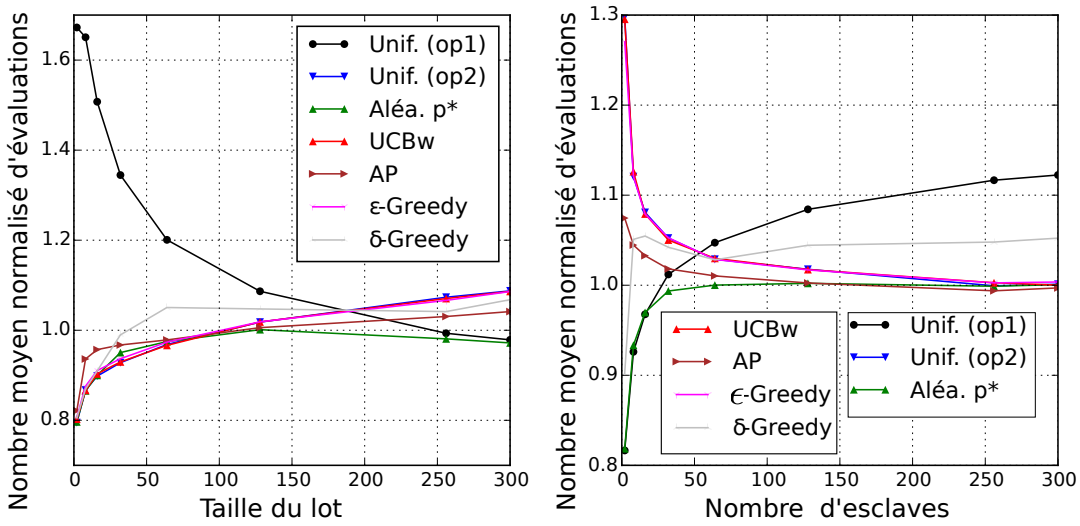


FIGURE 42 – Nombre moyen normalisé d'évaluations en fonction de la taille du lot en utilisant 128 esclaves (à gauche), et en fonction du nombre de esclaves utilisant une taille de lot 128 (à droite). Les valeurs sont normalisées par le nombre moyen d'évaluations de la stratégie aléatoire non-biaisée avec $p = 0.5$. Un nombre d'évaluations est faible indique une meilleure performance. Une stratégie avec une valeur normalisée inférieure à 1 est meilleure que la stratégie aléatoire non-biaisée, et inversement.

tives échouent à détecter l'opérateur le plus approprié. Nous émettons à cet effet l'hypothèse que d'autres mécanismes de sélection basés sur la récompense locale de chaque nœud esclave seraient meilleures qu'une sélection basée sur l'estimation de la récompense globale.

5.3.3 *Temps de communication*

Nous considérons maintenant l'impact de l'utilisation d'une stratégie par lot sur le temps de communication et ainsi sur l'efficacité parallèle. En adoptant le modèle analytique de Hadka *et al.* [HMR13], nous pouvons analyser l'impact de la taille des lots et du nombre de nœuds esclaves sur le temps total d'optimisation. En effet, dans [HMR13], le temps de d'optimisation est donné par :

$$T_S = \frac{N}{n}.(T_F + 2n.T_C + T_A)$$

où N est le nombre total d'évaluations, n le nombre de nœuds esclaves, T_F le temps d'évaluation de la fitness, T_C le temps de communication, et T_A le temps de génération de la nouvelle population (descendants). Dans leur modèle maître-esclave, Hadka *et al.* [HMR13] ont également introduit un

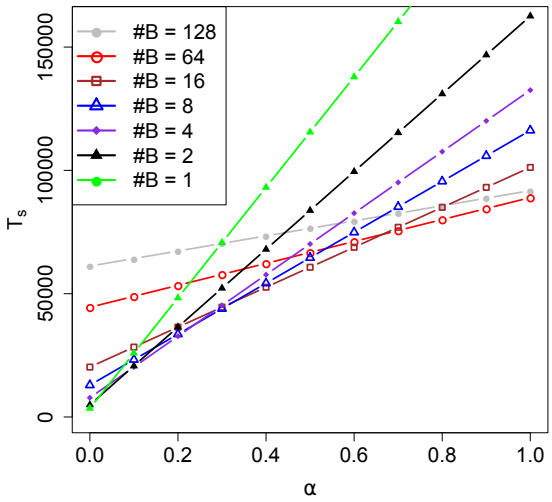


FIGURE 43 – Temps pour atteindre l'optimum global pour la stratégie adaptative poursuit (AP), en unité de fitness $T_F = 1$, en fonction du paramètre α pour la taille de lot différé $\sharp B$ et pour $n = 128$ nœuds esclaves.

ratio entre les coûts de communication et de calcul : $\alpha = \frac{T_C}{T_F + T_A}$. Dans le cadre de notre étude, le temps T_A pour produire la nouvelle population peut être omis. En conséquence, en réarrangeant le nombre d'évaluations par rapport aux nombres d'itérations R de notre algorithme globale adaptatif et en prenant en compte la taille du lot $\sharp B$, il est facile d'obtenir la relation suivante :

$$T_S = R \cdot (2\, n \cdot \alpha + \sharp B) \cdot T_F$$

Nous remarquons que dans l'équation précédente, α et T_F peuvent être considérés comme des paramètres qui dépendent de l'architecture physique de l'environnement parallèle considéré et de la fonction fitness du problème d'optimisation. À l'inverse, la valeur de R dépend de l'efficacité de l'algorithme distribué de sélection adaptative pour atteindre l'optimum global. Plus précisément, R est une fonction du choix des paramètres n et $\sharp B$ ainsi que de la stratégie de sélection elle même. Ainsi, le temps de calcul global peut être étudié d'un point de vue purement parallèle étant donnée que la valeur de R ne dépend pas de l'architecture parallèle et peut être déterminé par simulation.

En utilisant le nombre expérimental d'itérations R de la stratégie AP extrait à partir de nos expériences précédentes (voir Sect. 5.3.2), nous illustrons dans la Figure 43 le temps moyen pour atteindre la fitness optimale en fonction du rapport du coût de la communication α, en utilisant 128 nœuds esclaves et la valeur unitaire de $T_F = 1$. Premièrement, lorsque le coût de la communication est négligeable ($\alpha = 0$), une stratégie de lot avec $\sharp B > 1$ est inutile pour réduire le temps de communication, rendant le temps d'optimisation

plus élevé. Inversement, lorsque le coût de la communication est très élevé par rapport au coût d'évaluation de la fitness ($\alpha = 0.9$), la taille de lot optimale est grande ($\sharp B = 64$), mais pas la plus grande valeur de la taille du lot. En effet, pour une valeur intermédiaire du rapport de communication α, un compromis de la taille du lot est nécessaire pour pouvoir réduire le coût de communication sans diminuer la performance de la stratégie de sélection adaptative. Par exemple, pour $\alpha = 0.3$, la taille de lot optimale est autour des valeurs intermédiaires 4, 8, ou 16 ; mais au-delà de la taille 16, la performance de la méthode d'adaptation diminue significativement sans pouvoir être compensée par le gain espéré en terme de temps de communication.

5.4 SYNTHÈSE

Dans ce chapitre, nous avons étudié le modèle maitre-esclave pour la sélection adaptative d'opérateurs. Le nœud maître a une vision d'ensemble des récompenses des opérateurs exécutés localement par les différents nœuds esclaves. Le maître a ainsi un contrôle global de l'affectation des opérateurs alors que les nœuds esclaves n'ont qu'une vision locale puisqu'ils se contentent d'exécuter les opérateurs et renvoyer leur observations au maître de façon indépendante. Dans ce cadre, nous avons étudié les aspects critiques suivants : l'homogénéité des opérateurs, l'agrégation des mesures de récompense et le traitement par lot des opérateurs du portefeuille. Nous avons aussi considéré le modèle Fitness Cloud avec un scénario à deux opérateurs ayant une espérance d'amélioration constante.

Dans ce contexte, la sélection adaptative d'un opérateur pour chaque nœud de calcul peut être vu comme un problème d'allocation de ressource de calcul. Autrement dit, une fraction des ressources de calcul est allouée à chaque opérateur du portefeuille en sachant que l'ensemble des ressources est fini. Nous avons étudié dans ce chapitre deux cas extrêmes : la situation homogène et hétérogène où l'ensemble des nœuds de calcul appliquent soit exactement le même opérateur ou éventuellement des opérateurs différents. D'un point de vue abstrait, nous avons pu souligner le faible avantage de la situation hétérogène. La situation homogène est préférable quand la sélection de l'opérateur considère le meilleur opérateur, en particulier dans le cas d'un algorithme de type $(1 + \lambda)$-EA, où la probabilité d'amélioration instantanée est corrélée au nombre de nœuds. En effet, plus il y a de nœuds de calcul exécutant un opérateur en une itération, plus il y a une chance d'obtenir une meilleure amélioration instantanée. Cependant ce propos est à nuancer par rapport à la stratégie de sélection (UCB, AP, ϵ-Greedy, ...), car la plupart des stratégies sont intrinsèquement pensées pour avoir un compromis entre exploitation et exploitation. Intuitivement, réaliser une stratégie hétérogène permet d'améliorer la composante exploratoire de la stratégie de sélection. Nous pensons qu'allouer un faible nombre de nœuds de calcul aux opéra-

teurs du portefeuille inutilisé peut améliorer l'optimisation en permettant notamment de mieux anticiper les changements des performances des opérateurs.

À chaque itération, chaque nœud esclave génère des mesures de récompense pour les opérateurs exécutés. Cependant, l'agrégation de ces mesures peut être réalisée de différentes manières. Dans notre travail, nous avons comparé deux modalités qui consistent à considérer la moyenne et le maximum des récompenses estimées par les nœuds esclaves. Il s'avère que l'agrégation utilisant le maximum est la plus intéressante à utiliser dans le cadre du modèle Fitness Cloud étudié. En effet, ce type d'agrégation permet de mieux capturer le meilleur fonctionnement coopératif global lorsque un processus d'optimisation élitiste de type $(1+\lambda)$-EA est considéré.

En outre, réduire le nombre de communications soulève des questions algorithmiques difficiles. Nous avons considéré une approche par échange de lots d'opérateurs entre les esclaves et le maître. La manière de générer le lot basée sur l'historique des récompenses est importante pour avoir une bonne amélioration de la solution sur chaque nœud esclave. Cependant, une taille trop importante limite l'amélioration de la solution. Après exécution du lot, l'agrégation de l'ensemble des récompenses obtenues pour chaque exécution d'un opérateur du lot au niveau des nœuds esclaves provoque une perte d'information. De façon générale, plus le lot contient d'opérateurs, plus le temps d'optimisation est élevé, mais le nombre de messages échanger s'en trouve réduit.

Cette étude peut être étendue de plusieurs façons. Une première perspective est de considérer un environnement réellement parallèle et d'attaquer un problème d'optimisation « réel » ayant une fonction d'évaluation coûteuse. Un exemple de choix dont nous avons pu commencer l'étude, est le problème de mobilité urbaine traité par Armas *et al.* [Arm+16] ou encore l'optimisation des paramètres du pilotage d'un réacteur nucléaire [Mun+16]. Dans ce type d'applications, le calcul d'une valeur de fitness s'appuie en effet sur des calculs et des simulations coûteuses en terme de temps de calcul. À titre d'exemple, ceux-ci peuvent prendre plusieurs minutes pour le problème de mobilité urbaine et plusieurs dizaines de minutes dans le cas du réacteur nucléaire. Un autre perspective est l'étude d'un modèle asynchrone ; qui semble en effet être plus adapté à des environnements au temps de calcul hétérogènes où la variabilité dans le temps de calcul de la valeur de fitness est typiquement très variable d'une solution candidate à une autre. Une difficulté principale est de pouvoir agréger les récompenses de façon effective sachant que certaines récompenses peuvent ne plus être à jour au regard d'autres récompenses obtenues par des nœuds de calcul plus rapides.

6

CONCLUSION

Nous proposons dans cette thèse une étude approfondie des stratégies de sélection d'opérateurs/algorithmes/paramètres parmi un portefeuille dans un contexte de calcul distribué. Le paramétrage des algorithmes évolutionnaires, ou plus généralement des métaheuristiques est contemporain de l'invention de ces algorithmes, et toujours de circonstance de par son importance d'un point de vue fondamental et pratique. Le réglage de paramètres pendant l'exécution de l'algorithme d'optimisation permet de changer la valeur d'un paramètre ou plus généralement un opérateur, idéalement en choisissant la/le plus approprié, au cours de l'optimisation contrairement à un réglage de paramètres avant l'exécution de l'algorithme d'optimisation. L'étude de ce type de paramètre, dit apprentissage durant l'optimisation, vise à deux objectifs : avoir un paramétrage ad hoc pour une classe d'algorithmes et de problèmes donnés et acquérir de la connaissance sur le lien entre un opérateur et un problème. Auparavant principalement étudiés dans le contexte séquentiel, nous proposons dans ce manuscrit de l'étudier dans un cadre parallèle.

Eiben *et al* en 2000 [BFMoo, p. 173] soulignent deux difficultés importantes pour le réglage des paramètres basé dans un contexte d'optimisation « boîte noire » par essai/erreur : le coût de calcul et les incertitudes des paramètres choisis avec une stratégie de sélection. Le calcul distribué ne réduit pas le coût de calcul, mais le temps d'optimisation en augmentant la puissance de calcul par unité de temps. Il permet aussi pour un même temps d'explorer plus de solutions du problème. Ces techniques parallèles rendent possible l'optimisation en un temps raisonnable des problèmes très coûteux en ressource de calcul, par lorsque la fonction à optimiser est issue d'une simulation numérique. La recherche d'un opérateur de manière adaptative vise à réduire le coût en ressource de calcul en recherchant les paramètres optimaux à chaque itération du processus d'optimisation du problème. Pour un modèle de calcul parallèle, la sélection d'opérateurs s'appuie sur la récolte de plus d'information provenant des différents nœuds de calcul afin d'améliorer le choix. Pour un modèle de calcul séquentiel, augmenter le recueil d'information pour la sélection d'opérateurs revient mécaniquement à augmenter le temps d'optimisation global. Dans le cadre parallèle, l'information supplémentaire est utile à une meilleure sélection d'opérateurs à condition que les stratégies de sélection parallèle, comme nous avons proposé dans ce manuscrit, prennent en compte le nouveau compromis entre exploration et exploitation qui implique ce partage d'information entre pairs.

Dans la suite, nous discutons de la contribution apportée par cette thèse ainsi que des perspectives.

6.1 CONTRIBUTIONS

La figure 44 synthétise les principaux éléments étudiés dans la thèse. Décomposé en trois parties : architecture du modèle distribué (A), les échanges de solution entre les nœuds de calcul (B) et la conception de stratégie de sélection (C). Le chapitre 3 étudie les stratégies de sélection dans le cadre du modèle en île, le chapitre 4 étudie les stratégies de sélection dans différant contexte et le chapitre 5 s'est porté en particulier pour le cadre du modèle maître-esclave.

LE MODÈLE EN ÎLE Nous avons analysé en particulier le choix adaptatif d'un opérateur pour le problème OneMax et le problème NK. Ces travaux font suite aux travaux initier par **[derbel:2011]** introduisant le cadre DAMS. L'un des défis est le partage de la récompense entre les nœuds de calcul afin d'améliorer la sélection de l'opérateur d'un portefeuille. Nous avons ainsi comparé les stratégies de sélection indépendante, qui ne partage pas d'information avec les voisins, et les stratégies de sélection collective qui partage des informations avec les nœuds voisins.

Le partage d'information améliore le choix de la sélection puisque plus d'information instantanée est disponible pour prendre une décision. Nous avons montré que la prise en compte de l'historique des récompenses sans aucune remise en cause pénalise le choix de la stratégie de sélection, car dans un environnement non stationnaire l'opérateur le plus pertinent est susceptible de changer. L'oublie des récompenses passées permet de considérer les récompenses les plus importantes pour réaliser le choix.

À partir du problème OneMax, il en ressort deux éléments principaux : (i) la version collective d'une stratégie de sélection améliore l'optimisation et (ii) la version collective à besoin de moins explorer. Notons tout de même qu'une stratégie de sélection indépendante comme ϵ-Greedy-W_i avec les meilleurs méta-paramètres détecte très bien le bon opérateur.

Dans le cas du problème NK, les stratégies de sélections sont peu efficaces. Cela est dû à la faible différence entre les opérateurs. Les stratégies de sélection explorent les opérateurs du portefeuille pour trouver l'opérateur qui convient le mieux. Cependant, comme il est difficile de détecter l'opérateur pertinent, il passe beaucoup de temps à explorer (comme le UCB) et finalement peu de temps à exploiter. C'est pourquoi les stratégies naïves comme uniforme sont bien meilleures. Elles se concentrent sur le meilleur opérateur à utiliser.

LE MODÈLE FITNESS CLOUD Afin de mieux saisir le comportement des stratégies de sélection, nous avons développé le modèle fitness cloud. Il modélise le rapport entre certaines caractéristiques du problème d'optimisation et les opérateurs pouvant explorer celui-ci. Ce modèle utilisé dans différents scénarios avec des portefeuilles d'opérateurs dont les performances peuvent dépendre de l'état de la recherche. De nombreux scénarios sont possibles, c'est-à-dire des modèles de problème à résoudre, nous en avons retenu trois dont la complexité est croissante et dont les caractéristiques sont inspirées des problèmes OneMax et NK. Les deux principales caractéristiques sont : (i) la différence de l'espérance d'amélioration entre les opérateurs et (ii) la dégradation de l'espérance de l'amélioration à mesure que la fitness est élevée.

Ce modèle nous permet de mieux saisir le choix du type de paramétrage (avant l'optimisation ou durant l'optimisation) suivant les caractéristiques des paramètres du modèle. Nous avons étudié l'impact de la taille d'un portefeuille d'opérateurs et de chaque stratégie de sélection. Nous avons étudié le temps de convergence des stratégies de sélection à choisir l'opérateur supposé pertinent au début de l'optimisation comme en cours d'optimisation. Nous avons vu que la différence d'espérance d'amélioration entre les opérateurs a un fort impact sur le choix de la stratégie de sélection ainsi que la période au cours de laquelle un opérateur reste pertinent. Lorsque la période est courte et que la stratégie de sélection explore peu, il est fort probable que la stratégie de sélection change tardivement l'opérateur précédemment appris.

LE MODÈLE MAÎTRE-ESCLAVE Nous avons abordé modèle de calcul maître-esclave, à partir de scénario du modèle fitness cloud. Nous avons étudié en profondeur les composants algorithmiques du modèle de calcul : (i) le type de regroupement/agrégation des récompenses (ii) l'affectation d'une proportion de nœuds de calcul à chaque opérateur du portefeuille et (iii) le traitement par les nœuds esclaves d'un lot d'opérateurs sélectionnés par le nœud maître.

Nous avons étudié deux types d'agrégations : l'agrégation moyenne et l'agrégation par le maximum. L'agrégation par le maximum est le meilleur choix. Il permet aux stratégies de sélection de considérer l'opérateur avec les récompenses extrêmes qui correspond à la performance globale du système parallèle. Notons, dans des cas particuliers, que l'opérateur le meilleur opérateur change en fonction du type d'agrégation.

Nous avons adapté les stratégies de sélection stochastique telle que ϵ-Greedy, AP afin de proposer une proportion de nœuds, et donc de ressources de calcul à chaque opérateur du portefeuille. Nous avons proposé une nouvelle stratégie de sélection δ-Greedy qui fait varier la proportion des opérateurs en la faveur du meilleur opérateur.

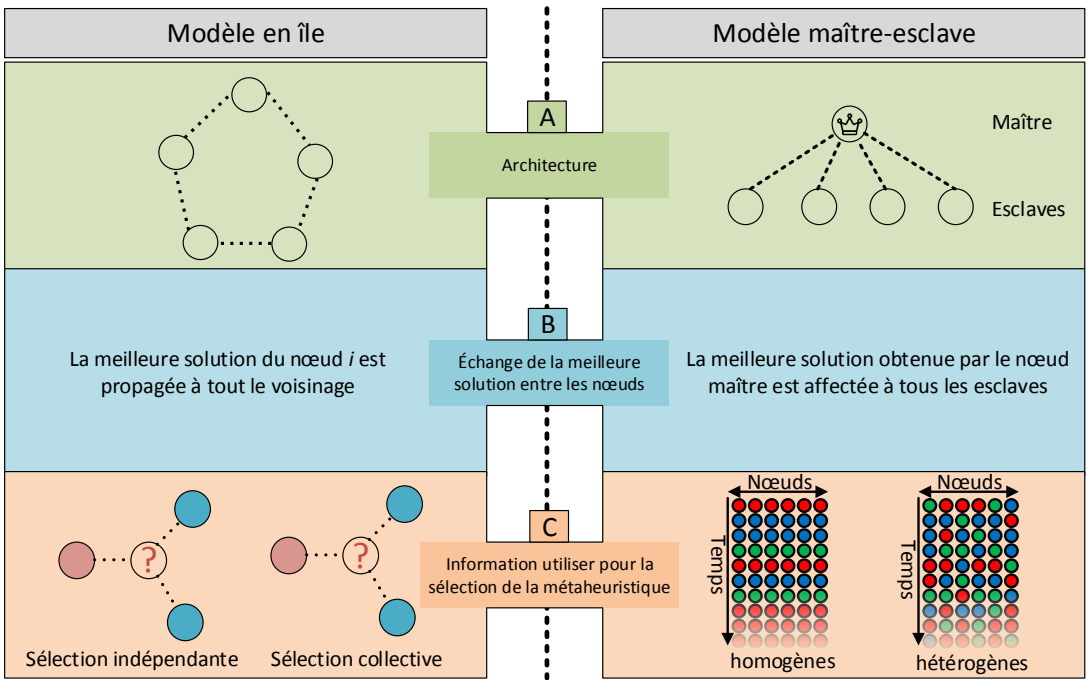


FIGURE 44 – Synthèse comparative entre le modèle en île et le modèle maître-esclave dans le cadre de la stratégie de sélection adaptative à portefeuille d'opérateurs.

Le traitement d'opérateurs par lot permet de réduire le nombre de messages échangés entre le nœud maître et les nœuds esclaves ; cependant cela augmente le temps d'optimisation. Plus la taille du lot à exécuter est grande, plus le temps d'optimisation est augmenté en conséquence.

6.2 PERSPECTIVES

D'un point vu de l'application de ces travaux, une des perspectives naturelles est de résoudre des problèmes issus des problématiques du monde réel coûteux en ressource de calcul par exemple la simulation multi-agents des déplacements d'agents au sein de la ville de Quito – capitale de l'Équateur – dont l'objet est l'amélioration du trafique et la diminution de la pollution émise par les véhicules à l'aide de la recherche d'une meilleure synchronisation des feux de circulation de la ville.

Nous avons analysé les stratégies de sélection adaptative parallèle utilisant des communications synchrones que se soit pour le modèle en île que pour le modèle maître-esclaves. Les communications synchrones sont pertinentes lorsque les temps d'exécution de l'évaluation d'une solution sont du même ordre, quel que soit le nœud de calcul. Cependant le temps d'exécution peut varier fortement selon la solution à évaluer et l'hétérogénéité des ressources de calcul. De plus, les pics de communication entre nœuds sont réduits dans

une communication asynchrone. Dans ce cas, il est alors plus approprié d'utiliser une communication asynchrone. De nouvelles problématiques se dégagent. En particulier, les informations échangées entre les nœuds de calcul peuvent être de différents moments correspondant à différents états de la recherche, et il s'agit alors de prendre en compte cet asynchronisme dans la sélection des paramètres : comment agréger les récompenses, comment décider des récompenses utiles, etc.

Dans le modèle de calcul maître-esclave proposé, le nœud coordonne complètement les nœuds esclaves en sélectionnant la suite de paramètres à exécuter sur chaque nœud esclave. Une perspective d'amélioration est d'intégrer sur chaque nœud esclave une stratégie de sélection propre qui permet de mettre à jour la sélection au fur et à mesure de la recherche. Plus généralement, que ce soit pour le modèle distribué ou maître-esclave, nous avons considéré seulement une même stratégie de sélection de paramètres, il serait possible d'imaginer des stratégies différentes sur chaque nœud de calcul. Ainsi, cette voie permet de concevoir une stratégie de sélection elle-même distribuée hétérogène sur l'environnement de calcul.

Les techniques de paramétrage automatique font appel elles-mêmes à des méta-paramètres. Bien que les méta-paramètres des méthodes utilisés dans ce manuscrit soient robustes dans de nombreuses situations, il serait intéressant d'étendre l'étude à d'autres problèmes académiques (problèmes UBQP, SAT, etc.). Des techniques spécifiques pourraient alors être mises en place pour régler, si besoin, ces méta-paramètres qui utilisent les propriétés du problème pour choisir ces méta-paramètres.

Lorsque la ressource de calcul est partagée par différents utilisateurs, typiquement les environnements *cloud computing*, ou plus généralement, lorsque le prix du calcul est pris en compte, réduire le nombre de ressources de calcul pour l'optimisation peut être un objectif. Dans le contexte de la sélection adaptative, le nombre de ressources nécessaires au cours du calcul n'est pas le même. Lorsque la différence entre opérateurs est importante, moins de tests d'opérateurs sont nécessaires pour les différencier ; au contraire, dans les phases d'optimisation avec des opérateurs de qualités proches, il peut être utile d'augmenter la quantité d'information sur ces opérateurs, et donc les ressources de calcul. De nouvelles stratégies de sélection adaptative qui contrôlent également le nombre de ressources de calcul sont à concevoir dans le futur.

BIBLIOGRAPHIE

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

[Jan+17b] Christopher Jankee, Sébastien Verel, Bilel Derbel et Cyril Fonlupt. "On the Design of a Master-Worker Adaptive Algorithm Selection Framework". In : *In 13th International Conference on Artificial Evolution*. Paris, France, 2017.

[Arm+16] Rolando Armas, Hernán Aguirre, Saúl Zapotecas-Martínez et Kiyoshi Tanaka. "Traffic Signal Optimization : Minimizing Travel Time and Fuel Consumption". In : *Artificial Evolution : 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*. Springer International Publishing, 2016, p. 29-43.

[GLS16] Adrien Goëffon, Frédéric Lardeux et Frédéric Saubion. "Simulating non-stationary operators in search algorithms". In : *Appl. Soft Comput.* 38 (2016), p. 257-268.

[Jan+16] Christopher Jankee, Sébastien Verel, Bilel Derbel et Cyril Fonlupt. "A Fitness Cloud Model for Adaptive Metaheuristic Selection Methods". In : *PPSN 2016*. 2016.

[Mun+16] Mathieu Muniglia, Jean-Michel Do, Le Pallec Jean-Charles, Hubert Grard, Sébastien Verel et S. David. "A Multi-Physics PWR Model for the Load Following". In : *Int. Cong. on Advances in Nuclear Power Plants (ICAPP)*. 2016.

[RF16] Denis Robilliard et Cyril Fonlupt. "Towards Human-Competitive Game Playing for Complex Board Games with Genetic Programming". In : *Artificial Evolution : 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*. Cham : Springer International Publishing, 2016, p. 123-135.

[Ver16] Sébastien Verel. "Contributions to fitness landscapes analysis for single- and multi-objective optimization". Habilitation à diriger des recherches. Université du Littoral Côte d'Opale, 2016.

[WRM16] Simon Wessing, Günter Rudolph et Dino A. Menges. "Comparing Asynchronous and Synchronous Parallelization of the SMS-EMOA". In : *Parallel Problem Solving from Nature – PPSN XIV : 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. Cham : Springer International Publishing, 2016, p. 558-567.

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

[Jan+15a] Christopher Jankee, Sébastien Verel, Bilel Derbel et Cyril Fonlupt. "New Adaptive Selection Strategies for Distributed Adaptive Metaheuristic Selection". In : *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO Companion '15. Madrid, Spain : ACM, 2015, p. 1405-1406.

[Jan+15b] Christopher Jankee, Sébastien Verel, Bilel Derbel et Cyril Fonlupt. "Distributed Adaptive Metaheuristic Selection : Comparisons of Selection Strategies". In : *EA 2015*. 2015, p. 83-96.

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

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

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

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

[Goë14] Adrien Goëffon. "Modèles d'abstraction pour la résolution de problèmes combinatoires". 175 pages. Habilitation à diriger des recherches en informatique. Université d'Angers, 2014.

[Sia14] P. SIARRY. *Métaheuristiques : Recuits simulé, recherche avec tabous, recherche à voisinages variables, méthodes GRASP, algorithmes évolutionnaires, fourmis artificielles, essaims particulaires et autres méthodes d'optimisation*. Algorithmes. Eyrolles, 2014.

[Can+13] Caner CANDAN, Adrien GOËFFON, Frédéric LARDEUX et Frédéric SAUBION. "Non stationary operator selection with island models". In : *GECCO*. 2013, p. 1509-1516.

[HMR13] D. HADKA, K. MADDURI et P. REED. "Scalability Analysis of the Asynchronous, Master-Slave Borg Multiobjective Evolutionary Algorithm". In : *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*. 2013, p. 425-434.

[TF13] R. TANABE et A. FUKUNAGA. "Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms". In : *Evolutionary Computation (CEC), 2013 IEEE Congress on*. 2013, p. 1263-1270.

[Can+12] Caner CANDAN, Adrien GOEFFON, Frédéric LARDEUX et Frédéric SAUBION. "A Dynamic Island Model for Adaptive Operator Selection". In : *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. GECCO '12. Philadelphia, Pennsylvania, USA : ACM, 2012, p. 1253-1260.

[Kot12] Lars KOTTHOFF. "Algorithm Selection for Combinatorial Search Problems : A Survey". In : *arXiv :1210.7959 [cs]* (2012), p. 48-60. arXiv : 1210.7959.

[DV11] Bilel DERBEL et Sébastien VEREL. "DAMS : distributed adaptive metaheuristic selection". In : *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. ACM Press, 2011, p. 1955-1962.

[GL11] A. GOËFFON et F. LARDEUX. "Optimal One-Max Strategy with Dynamic Island Models". In : *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*. 2011, p. 485-488.

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

[Lóp+11] Manuel LÓPEZ-IBÁÑEZ, Jérémie DUBOIS-LACOSTE, Thomas STÜTZLE et Mauro BIRATTARI. *The Rpackageirace package, Iterated Race for Automatic Algorithm Configuration*. Rapp. tech. TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[Mar+11] Marie-Eléonore Marmion, Clarisse Dhaenens, Laetitia Jourdan, Arnaud Liefooghe et Sébastien Verel. "On the Neutrality of Flowshop Scheduling Fitness Landscapes". In : *Learning and Intelligent Optimization : 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 238-252.

[Bur+10] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan et John R. Woodward. "A classification of hyper-heuristic approaches". In : *Handbook of metaheuristics*. Springer, 2010, p. 449-468.

[Fia10] Álvaro Fialho. "Selection Adaptative d'Operateurs pour l'Optimisation". Thèse de doct. Université Paris-Sud 11, 2010. 237 p.

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

[Fia+10b] Álvaro Fialho, Raymond Ros, Marc Schoenauer et Michèle Sebag. "Comparison-based adaptive strategy selection with bandits in differential evolution". In : *Parallel Problem Solving from Nature, PPSN XI*. Springer, 2010, p. 194-203.

[OVT10] Gabriela Ochoa, Sébastien Verel et Marco Tomassini. "First-Improvement vs. Best-Improvement Local Optima Networks of NK Landscapes". In : *Parallel Problem Solving from Nature, PPSN XI : 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 104-113.

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

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

[Tal09] El-Ghazali Talbi. *Metaheuristics : From Design to Implementation*. Wiley Publishing, 2009.

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

[Xu+08] Lin Xu, Frank Hutter, Holger H. Hoos et Kevin Leyton-Brown. "SATzilla : Portfolio-based Algorithm Selection for SAT". In : *J. Artif. Int. Res.* 32.1 (2008), p. 565-606.

[BBS07] Prasanna Balaprakash, Mauro Birattari et Thomas Stützle. "Improvement Strategies for the F-Race Algorithm : Sampling Design and Iterative Refinement". In : *Hybrid Metaheuristics : 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 108-122.

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

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

[Ver+07] Sébastien Verel, Philippe Collard, Marco Tomassini et Leonardo Vanneschi. "Fitness landscape of the cellular automata majority problem : View from the Olympus". In : *Theoretical Computer Science* 378.1 (2007), p. 54-77.

[AL06] Belarmino Adenso-Díaz et Manuel Laguna. "Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search". In : *Operations Research* 54.1 (2006), p. 99-114.

[AO06] Charles Audet et Dominique Orban. "Finding Optimal Algorithmic Parameters Using Derivative Free Optimization". In : *SIAM Journal on Optimization* 17.3 (2006), p. 642-664.

[DGP06] Marc Dubreuil, Christian Gagne et Marc Parizeau. "Analysis of a Master-Slave Architecture for Distributed Evolutionary Computations". In : *IEEE T. on Systems, Man, and Cybernetics : Part B* 36 (2006), p. 229-235.

[Van+06] Leonardo Vanneschi, Marco Tomassini, Philippe Collard et Sébastien Vérel. "Negative Slope Coefficient : A Measure to Characterize Genetic Programming Fitness Landscapes". In : *Genetic Programming : 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 178-189.

[AH05] A. Auger et N. Hansen. "Performance evaluation of an advanced local search evolutionary algorithm". In : *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. T. 2. IEEE. 2005, p. 1777-1784.

[Thi05] Dirk Thierens. "An adaptive pursuit strategy for allocating operator probabilities". In : *GECCO'05*. 2005, p. 1539-1546.

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

[CVC04] Philippe Collard, Sébastien Verel et Manuel Clergue. "Local search heuristics : Fitness Cloud versus Fitness Landscape". In : *the 2004 European Conference on Artificial Intelligence (ECAI04)*. Valencia, Spain : IOS Press, 2004, p. 973-974.

[Bar03] Lionel Barnett. "Evolutionary search on fitness landscapes with neutral networks". Thèse de doct. University of Sussex, UK, 2003.

[LMS03] Helena R. Lourenço, Olivier C. Martin et Thomas Stützle. "Iterated Local Search". In : *Handbook of Metaheuristics*. Boston, MA : Springer US, 2003, p. 320-353.

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

[AT02] E. Alba et M. Tomassini. "Parallelism and Evolutionary Algorithms". In : *Trans. Evol. Comp* 6.5 (2002), p. 443-462.

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

[Bir+02] Mauro Birattari, Thomas Stützle, Luis Paquete et Klaus Varrentrapp. "A Racing Algorithm for Configuring Metaheuristics". In : *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '02. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2002, p. 11-18.

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

[BFM00] Thomas Bäck, David B. Fogel et Zbigniew Michalewicz, éd. *Evolutionary Computation 2 : Advanced Algorithms and Operators*. 1st. Bristol, UK, UK : IOP Publishing Ltd., 2000.

[EFM00] Thomas Evolutionary Computation 1 : Bäck, David B. Fogel et Zbigniew Michalewicz, éd. *Basic Algorithms and Operators*. 1st. Bristol, UK, UK : IOP Publishing Ltd., 2000.

[FR00] Cyril FONLUPT et Denis ROBILLIARD. "Genetic Programming with Dynamic Fitness for a Remote Sensing Application". In : *Parallel Problem Solving from Nature PPSN VI : 6th International Conference Paris, France, September 18–20, 2000 Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000, p. 191-200.

[Sko00] Zbigniew Maciej SKOLICKI. "An Analysis of Island Models in Evolutionary Computation". Thèse de doct. Fairfax, VA, USA, 2000.

[Can98] Erick CANTÚ-PAZ. "A Survey of Parallel Genetic Algorithms". In : *CALCULATEURS PARALLELES, RESEAUX ET SYSTEMS REPARTIS* 10 (1998).

[SB98] Richard S. SUTTON et Andrew G. BARTO. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA : MIT Press, 1998.

[LLG97] Fernando LOBO, O G. LOBO et David E. GOLDBERG. *Decision Making in a Hybrid Genetic Algorithm*. 1997.

[WM97] David H WOLPERT et William G MACREADY. "No free lunch theorems for optimization". In : *IEEE T. Evolutionary Computation* 1.1 (1997), p. 67-82.

[DMC96] M. DORIGO, V. MANIEZZO et A. COLORNI. "Ant system : optimization by a colony of cooperating agents". In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), p. 29-41.

[Kau93] S. A. KAUFFMAN. *The Origins of Order*. Oxford University Press, 1993.

[KK93] S.A. KAUFFMAN et S. KAUFFMAN. *The Origins of Order : Self-organization and Selection in Evolution*. Oxford University Press, 1993.

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

[Koz92] John R. KOZA. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA : MIT Press, 1992.

[CDM91] Alberto COLORNI, Marco DORIGO et Vittorio MANIEZZO. "Distributed Optimization by Ant Colonies". In : *European Conference on Artificial Life*. 1991, p. 134-142.

[Ing89] L. INGBER. "Very fast simulated re-annealing". In : *Mathematical and Computer Modelling* 12.8 (1989), p. 967-973.

[Glo86] Fred GLOVER. "Future Paths for Integer Programming and Links to Artificial Intelligence". In : *Comput. Oper. Res.* 13.5 (1986), p. 533-549.

[Gre86] John J Grefenstette. "Optimization of control parameters for genetic algorithms". In : *Systems, Man and Cybernetics, IEEE Transactions on* 16.1 (1986), p. 122-128.

[Čer85] V. Černý. "Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm". In : *Journal of Optimization Theory and Applications* 45.1 (1985), p. 41-51.

[KGV83] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi. "Optimization by simulated annealing". In : *SCIENCE* 220.4598 (1983), p. 671-680.

[Ric76] John R. Rice. "The Algorithm Selection Problem." In : *Advances in Computers* 15 (1976), p. 65-118.

[Wri32] S. Wright. "The roles of mutation, inbreeding, crossbreeding and selection in evolution". In : 1932, p. 355-366.