UNIVERSITÉ DE MONTRÉAL

ÉTUDE D'HEURISTIQUES DISTRIBUÉES DE RECHERCHE SANS CONTRÔLE GLOBAL EXPLICITE

MICHEL TOULOUSE
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)
(GÉNIE ÉLECTRIQUE)
NOVEMBRE 1996

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

ÉTUDE D'HEURISTIQUES DISTRIBUÉES DE RECHERCHE SANS CONTRÔLE GLOBAL EXPLICITE

présentée par: TOULOUSE Michel
en vue de l'obtention du diplôme de: Philosophiae Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. BERNARD Jean-Charles, Ph.D., président
Mme SANSO Brunilde, Ph.D., membre et directeur de recherche
M. CRAINIC Théodor G., Ph.D., membre et co-directeur de recherche
M. GENDREAU Michel, Ph.D., membre
Mme ROUCAIROL Catherine, Ph.D., membre externe

À mes parents

REMERCIEMENTS

Je voudrais tout d'abord remercier Théodor Crainic pour avoir accepté de superviser cette recherche pendant toutes ces années. Sa grande disponibilité, son ouverture d'esprit et sa franchise ont permis de développer et de maintenir une relation fort enrichissante sur le plan scientifique et humain. Je ne crois pas que j'aurais pu espérer un meilleur encadrement, il m'a toujours donné toute la latitude désirée dans mon projet tout en me faisant bénéficier de son immense expérience de chercheur. Je tiens également à exprimer ma plus grande gratitude à Brunilde Sanso et Claude Évequoz pour avoir accepté de s'impliquer dans la codirection de cette recherche. Leur apport fut indispensable au succès de ce projet.

La plus grande partie de ma recherche au doctorat s'est déroulé au Centre de Recherche sur les Transports (CRT). Tous les résultats expérimentaux de cette thèse ont été obtenu à partir du réseau de stations de travail du CRT et du GERAD. Avant que le réseau dédié au parallélisme n'existe, les professeurs, chercheurs et programmeurs du CRT et du GERAD ont gracieusement accepté de m'accorder du temps machine pour que je puisse réaliser mes tests sur le traitement parallèle, je leur en suis grandement reconnaissant. Je tiens également à souligner le professionnalisme tout à fait exceptionnel de l'équipe responsable du maintien de ce réseau de stations de travail au CRT. Malgré les mises à jour constantes que ce réseau a dû subir au cours des années, le programmeur de bas niveau qu'il m'arrive d'être, n'a jamais eu à constater la moindre conséquence pour ses applications (si ce n'est une amélioration des performances du réseau). J'ai eu à travailler avec plusieurs personnes au CRT, Patrick Soriano m'a assisté de son esprit méthodologique lors de la conception des premiers programmes sur la méthode tabou. L'énigmatique Michel Gendreau, je lui suis redevable de plusieurs petits conseils qui je crois vont pouvoir s'appliquer à plusieurs situations, ils en sont d'autant plus précieux. Un gros merci à tous les autres collègues du centre, ils m'ont permis de travailler dans une ambiance de

recherche formidable et de vivre une expérience enrichissante et inoubliable.

J'ai obtenu ma formation de base en traitement parallèle de mes cours à l'Université McGill. À ce titre je suis particulièrement redevable au Professeur Guang et son groupe de recherche (Advanced Compilers, Architecture, and Parallel Systems). J'ai pu suivre l'évolution rapide des systèmes parallèles en partie grâce à l'apport financier du Centre de Recherche en Informatique Distribué et d'Alex Informatique. Ils m'ont permis de participer à plusieurs congrès internationaux et de suivre des cours donnés par les meilleurs spécialistes mondiaux en traitement parallèle.

Enfin je tiens à remercier toute la famille, proche et lointaine, et les amis qui ont dû subir trop souvent mes désistements aux repas, réunions de famille et rencontres, mais qui ont tout de même continué de me témoigner toute leur affection et un soutien indéfectible tout au long de ce processus.

RÉSUMÉ

La théorie du calcul constitue l'un des aboutissements du formalisme en philosophie des sciences. On peut situer l'origine de ce courant philosophique dans les travaux de Leibniz visant à définir un langage mathématique universel dans lequel toutes les propositions d'un langage quelconque pourraient être traduites et calculées. Appartiennent également à ce courant, des mathématiciens et logiciens comme Fredge, Russell, Hilbert et Gödel. L'objectif principal de ce courant fut de réaliser l'axiomatisation des sciences, c'est-à-dire la déduction à partir d'un certain nombre d'axiomes, de toute la connaissance scientifique sous forme de propositions mathématiques. Pour atteindre ce but, il fallait pouvoir exprimer le savoir scientifique sous forme d'un système formel cohérent, c'est-à-dire tel qu'aucun énoncé ne puisse être à la fois vrai et faux. Cependant Gödel devait montré qu'aucun système d'axiomes arithmétiques ne peut être à la fois cohérent et complet. Dans sa preuve, Gödel faisait appel la notion d'algorithme, qui n'avait alors qu'une valeur intuitive. Turing a donné une valeur formelle à cette notion en la rendant équivalente à celle d'une application mécanique en termes des opérations d'une machine qui simule le comportement d'un humain résolvant un problème mathématique. Cette formalisation de la notion d'algorithme est la base sur laquelle repose toute la théorie actuelle du calcul.

Von Neumann a fait remarquer que la Machine de Turing ne pouvait pas décrire le fonctionnement de certains systèmes de traitement de l'information, par exemple celui de la vision chez l'être humain. Se basant sur les travaux portant sur les réseaux de neurones de Warren McCulloch et Walter Pitts et sur les résultats de Turing sur la théorie du calcul, von Neumann a voulu développer un modèle de calcul qui puisse tenir compte de tous les modes (artificiels ou naturels) de traitement de l'information. C'est sous la forme des automates cellulaires que la théorie des automates de von Neumann s'est finalement cristallisée. Le modèle de calcul

de l'automate cellulaire représente la totalité des états et des dynamiques de changement dans un seul champs, au lieu de séparer ces facteurs selon des données symboliques et une unité de contrôle. L'évolution dans le temps d'un automate cellulaire se compare à celui d'un système dynamique autonome. Il s'agit d'un modèle de calcul où le parallélisme est inhérent contrairement à la Machine de Turing qui est un modèle du calcul séquentiel.

Dans cette thèse nous allons étudier des systèmes parallèles de calcul dont une partie de la dynamique peut être modélisée à l'intérieur du cadre formel de la théorie des automates cellulaires. Nous allons nous intéresser à des stratégies de parallélisation où chaque tâche concurrente exécute un algorithme particulier (en fait une heuristique dans le cadre de la présente recherche), mais où l'interaction entre les tâches est définie par une dynamique complexe dont plusieurs paramètres échappent au contrôle des algorithmes séquentiels de chaque tâche. Chaque noeud du système parallèle exécute une métaheuristique tabou, une méthode de résolution approchée pour des problèmes d'optimisation combinatoire. Chaque tâche coopère avec les autres par le biais d'échange d'information portant sur l'espace de solutions du problème à résoudre. Nous identifierons par "recherches multiples coopérantes" ce type de stratégies de parallélisation et par "connaissances acquises" sur l'espace de solutions, l'information échangée entre les tâches concurrentes. Nous allons montrer que l'échange de connaissances acquises engendre une dynamique d'interaction fort complexe entre les tâches concurrentes. Cette dynamique d'interaction tend à minimiser l'entropie de la recherche parallèle au détriment de la logique d'optimisation de l'algorithme exécuté au niveau de chaque noeud du système.

Cette argumentation sera étoffée d'expérimentations portant sur plusieurs parallélisations de la méthode tabou et d'une analyse théorique développée à partir de la théorie des automates cellulaires, et de modèles empruntés à la mécanique statistique et à la théorie des systèmes dynamiques. Nous avons en effet développé un modèle qualitatif de la dynamique d'interaction entre les tâches concurrentes

en terme d'une interaction entre les variables d'un système dynamique. Dans ce modèle, chaque tâche concurrente exécute la même heuristique que celle utilisée pour la parallélisation par recherches multiples coopérantes. Cependant, au niveau du modèle, nous avons remplacé l'échange de connaissances acquises entre les tâches concurrentes par des messages dont le contenu est généré par l'évolution dans le temps d'un système dynamique vers une réduction de son entropie (accroissement de son niveau d'organisation). En conséquence, les tâches concurrentes ne s'échangent plus de connaissances acquises, lesquelles connaissances proviennent de la logique d'optimisation de la méthode de recherche. À la place, nous avons implanté des procédures parallèles où les tâches concurrentes s'échangent de l'information visant à diriger l'exploration dans l'espace de solutions de la procédure parallèle vers un attracteur du système dynamique, c'est-à-dire à accroître le niveau d'organisation (au sens de la mécanique statistique) de l'exploration parallèle .

Nos résultats sont encore préliminaires, mais ils indiquent que les procédures parallèles basées sur l'échange de connaissances acquises et celles basées sur des modèles qualitatifs d'interaction ont le même comportement, c'est-à-dire que les deux types de procédures tendent à long terme à minimiser l'entropie de l'exploration de l'espace de solutions. Si ce résultat se confirme, il impliquerait, contrairement à ce que l'on croyait auparavant, que l'échange de connaissances acquises entre les tâches des heuristiques parallèles obéit à une logique différente de celle des méthodes d'optimisation (laquelle vise à orienter l'exploration de l'espace de solutions vers des régions intéressantes au sens de la fonction économique). En particulier, comme nos résultats semblent l'indiquer, la logique d'interaction entre les tâches concurrentes pourrait faire l'objet d'une approximation par des modèles provenant de théories comme la mécanique statistique, la théorie des systèmes dynamiques et la théorie de l'information. Cette conclusion n'est pas sans intérêt car elle implique que nous pourrions utiliser le vaste ensemble de résultats mathématiques provenant de ces différents champs pour concevoir et analyser les versions parallèles de plusieurs

heuristiques connues. Dans cette thèse, nous avons fait un pas dans cette direction. En plus du résultat montrant le type de comportement à long terme des recherches multiples coopérantes, l'analyse des procédures parallèles dont la coopération a été conçue à partir de modèles décrivant l'évolution d'un système dynamique, nous a permis d'expliquer les causes des anomalies d'accélération parallèles des recherches multiples coopérantes. C'est également à partir de cette approche que les procédures parallèles par recherches multiples coopérantes les plus performantes ont été conçues, trouvant la solution optimale pour tous les problèmes testés.

ABSTRACT

This research concerns the parallelization of search heuristics applied to combinatorial optimization problems. Specifically we focus on the tabu search method, and parallelization strategies based on the concurrent execution of many sequential searches. Each sequential search cooperates with the others by exchanging information on the problem's solution space. We identify by "cooperating multiple searches" these parallelization strategies and by "knowledge" the information exchanged on the solution space among the sequential searches. The parallelization of heuristics using cooperating multiple searches faces two different kinds of problems which we usually don't meet when the parallelization strategies are based on the problem decomposition:

- The data dependencies of the sequential program can not be used to define how the cooperation will be performed among the parallel tasks. Consequently, the design of a parallel procedure based on cooperating multiple searches involves specifying how the cooperation will be executed.
- An increase in the number of the sequential searches may result in worse solutions to the optimization problem.

Our investigation of these two problems go along the following five points:

- The above computation performed by heuristics is based on speculative steps (non-deterministic computation) as oppose to deterministic computation.
- The main source of parallelism of heuristics is based on parallel execution of alternative speculative steps, which is basically the case for the cooperating multiple searches.
- One of the side effects of the information exchange among sequential searches is that chains of interactions among exploration strategies of the sequential

searches will be triggered (that is, an initial exchange of knowledge might trigger many other message exchanges among sequential searches, which would have not happened without the initial exchange).

- The chains of interactions interfere with the exploration strategies and modify the way speculative steps will be computed by the parallel procedure.
- The interference lead to an entropy reduction of the parallel search at the detriment of the optimization logic of the search method.

The above arguments will be supported by experimental results provided by a large set of computational tests on the parallelization of the tabu search method, and by a theoretical analysis provided by models from statistical mechanics and the dynamical systems theory. All those tests performed in this research, lead us to propose a structured design approach to define how cooperation should be performed among parallel tabu search tasks based on multiple searches. Beside the chains of interactions, there are other side effects of the sharing of information among sequential searches, such as the uniformisation of the knowledge across the tabu search memories of many sequential searches and premature convergence of the parallel procedure. Based on generalizations from our experimental data and observations reported by other researchers on genetic algorithms and simulated annealing, we have defined a set of criteria that will help to decide which information should be exchanged among sequential searches, when it should be exchanged and among which sequential searches. These criteria should help to define the cooperation among sequential searches, and to control the uniformisation of the knowledge and premature convergence. Our theoretical analysis aims to explain some behaviors of parallel procedures based on cooperating multiple searches. This theoretical analysis is used to model the interference of the interaction among sequential searches and its impact on the exploration of the solution space in terms of the evolution of a dynamical system through its different attractors. We have replaced the knowledge exchange among sequential searches by qualitative models of the interaction among

variables of a dynamical system. More specifically, we have chosen models in which the dynamical system evolves to a reduction of its entropy (which result in an increase of the system organization). In this new design, the sequential searches do not exchange knowledge on the solution space, they exchange information which aims to lead the exploration of the solution space performed by the parallel procedure to attractors of the dynamical system, that is to increase (from a statistical mechanics point of view) the level of organization of the parallel exploration. Our results are still preliminaries, but they show that parallel procedures based on knowledge exchange, and those based on qualitative models of interaction, have both the same behavior from a statistical mechanics point of view, that is, both procedures tend to minimize the entropy of the solutions space exploration. If these preliminary results were to be ratified, they will imply, contrary to what was generally admit, that the exchange of knowledge among tasks of parallel heuristics follows a logic different from the one used by optimization methods (which aims to direct the search in solution space to interesting regions according to the economic function). Specially, as our results seem to indicate, if the logic of interaction among sequential searches can be approximated by models from theories like the statistical mechanics, the dynamical systems theory and the information theory, we will be in the position to use the mathematical results of these fields to design and analyse the parallel versions of many known heuristics. Beside these results on the investigation of the long term behavior of cooperating multiple searches, the theoretical analysis used in this thesis shows why the performances of parallel procedure based on cooperating multiple searches can be degraded as the number of sequential searches increases. Finally, based on this analysis, we have designed parallel procedures using cooperating multiple searches which give us the best results of all the parallel procedures tested in this research, finding the optimal solution for all the problems.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

Chapitre 1
Introduction

Nous allons nous intéresser à des procédures heuristiques dont le caractère intrinsèquement séquentiel pose certaines limitations à l'utilisation de stratégies de parallélisation traditionnelles basées sur une décomposition fonctionnelle ou du domaine de la procédure séquentielle. C'est le cas pour les recherches itératives comme la méthode tabou et le recuit simulé. Pour ces méthodes de résolution, il existe cependant une autre source de parallélisme qui n'est pas limitée par le caractère séquentiel des recherches itératives. Cette source de parallélisme provient de plusieurs exécutions de la même recherche séquentielle en utilisant des paramètres de recherche différents. On identifie par "recherches multiples" les stratégies de parallélisation basées sur cette source de parallélisme.

Il n'est pas nécessaire de disposer de plusieurs processeurs pour implanter une stratégie par recherches multiples, on peut simplement exécuter chaque recherche séquentielle l'une à la suite de l'autre sur le même processeur. Taillard [93], ainsi que d'autres chercheurs, ont montré que dans certains cas p recherches multiples exécutant chacune p itérations de la méthode de recherche pouvaient obtenir de meilleurs résultats qu'une seule recherche séquentielle exécutant $t \times p$ itérations. Clearwater, Hogg & Huberman [24], Huberman & Hogg [25], Clearwater, Hogg & Williams [55] et Huberman [56] ont montré expérimentalement et à travers des modèles probabilistes que les recherches multiples pouvaient obtenir de meilleures performances lorsqu'il y a échange d'information entre les recherches séquentielles. Nous identifierons par "recherche multiples coopérantes" cette dernière catégorie de stratégies de parallélisation.

Les stratégies de parallélisation par recherches multiples coopérantes posent des problèmes inconnus aux stratégies de parallélisation basées sur la décomposition

d'une recherche séquentielle ou sur l'exécution de plusieurs recherches séquentielles indépendantes. En effet, l'interaction entre les recherches multiples doit être spécifiée de manière directe ou indirecte en terme de quels attributs du chemin d'exploration devront être échangés entre les procédures de recherche séquentielle, à quelles itérations l'échange doit prendre place et entre quelles procédures de recherche séquentielle. Également, les procédures parallèles basées sur ce type de stratégies de parallélisation ont parfois des comportements plutôt inhabituels tel une détérioration des performances lorsque le nombre de processeurs augmente (la meilleure solution trouvée est parfois moins bonne lorsqu'on augmente le nombre de recherches).

Mais les stratégies de parallélisation par recherches multiples coopérantes offrent la possibilité d'étudier certains paradigmes du calcul, mieux adaptés aux notions de traitement de l'information par des systèmes complexes. C'est le cas du paradigme de calcul basé sur l'exploitation de la dynamique d'auto-organisation des systèmes complexes et du paradigme du calcul émergent [30, 32, 40, 50, 76, 79]. Très brièvement, ces paradigmes se fondent sur des principes d'organisation du calcul empruntés à certains modèles de la théorie physique des systèmes complexes, principes que l'on pourrait situer entre les modèles déterministes et non déterministes de la théorie classique du calcul, si un tel lieu existait. Par exemple, une interaction non contrôlée (non programmée) entre les processeurs d'une procédure parallèle peut donner naissance à un comportement organisé (auto-organisé) de cette interaction pouvant être interprété comme un deuxième niveau de calcul de la procédure parallèle. On aurait ainsi deux niveaux de calcul, le niveau explicite qui exécute les directives du programme sur chaque processeur, et un niveau implicite résultant de l'interaction (par effets de bord) entre les programmes explicites. Selon la théorie classique du calcul, le niveau implicite correspond à l'exécution d'une procédure non déterministe et ne peut donc pas être interprété en terme des étapes d'une procédure de calcul. Selon le paradigme émergent, les systèmes complexes de calcul reposent sur le même type d'organisation observé au niveau de plusieurs systèmes physiques,

cette organisation peut être utilisée comme outil implicite de programmation (ici programmation = organisation). Une des conclusions de la présente recherche est que ces deux niveaux de calcul semblent coexister lorsqu'il y a échange d'information entre les recherches multiples.

Le développement de l'argumentation de cette thèse commence par l'analyse des "anomalies" de comportement de certaines méthodes de recherche implicite dont la parallélisation est obtenue à partir d'une décomposition fonctionnelle ou du domaine de la procédure séquentielle. Nous montrerons que ces anomalies sont attribuables à des modifications des heuristiques utilisées par les méthodes de recherche implicite. Ces modifications proviennent de la décomposition de la méthode séquentielle. Cette décomposition modifie l'information heuristique servant à déterminer le parcours de l'espace de solutions. Puisque l'information heuristique n'est pas la même suite à la parallélisation d'une recherche implicite, et puisque cette information heuristique détermine la façon dont une procédure explore l'espace de solutions, l'exploration effectuée par la procédure parallèle peut être très différente de celle exécutée par la procédure séquentielle.

Nous allons montrer que l'interaction entre les procédures séquentielles des recherches multiples coopérantes a un effet similaire à celui de la décomposition d'une recherche implicite, elle change l'heuristique. En effet, dans une procédure avec recherches multiples, chaque recherche séquentielle exécute la stratégie d'exploration défini par la méthode de recherche. Mais dans le cas d'une procédure avec recherches multiples coopérantes, l'échange d'information entre les recherches multiples affecte l'information heuristique dont dispose chaque stratégie d'exploration. Conséquemment, le parcours de l'espace de solutions effectué par une procédure séquentielle p_i ne dépend plus uniquement de la stratégie d'exploration de p_i, mais dépend également de l'échange d'information heuristique effectué entre les procédures séquentielles.

Nous allons classifier les stratégies de parallélisation appliquées aux méthodes

de recherche en fonction de l'usage qui est fait par ces stratégies de parallélisation de l'information heuristique comme source de parallélisme. Par la suite, nous allons porter notre attention sur la parallélisation des méthodes de recherche itératives et les méthodes basées sur une exploration aléatoire biaisée (algorithmes génétiques) de l'espace de solutions. Plus particulièrement, nous allons définir une taxonomie des stratégies de parallélisation applicable de manière spécifique à la méthode de recherche tabou, taxonomie qui tient compte des raffinements découlant de l'échange d'information heuristique entre les tâches d'une procédure parallèle comme la méthode tabou. Presque toutes les stratégies de parallélisation de la taxonomie seront implantées et testées. Ce vaste ensemble d'expérimentations nous sera utile pour la phase suivante de modélisation du comportement des recherches multiples coopérantes.

Nous avons étudier en détail la nature de l'échange d'information heuristique entre les procédures séquentielles des stratégies de parallélisation avec recherches multiples coopérantes. Des approches empruntées à la mécanique statistique et à la théorie des systèmes dynamiques ont été utilisées pour modéliser l'interaction entre les stratégies d'exploration et ses effets sur l'exploration de l'espace de solutions. Notre recherche montre qu'il est possible d'expliquer certains aspects du comportement d'une procédure parallèle par recherches multiples coopérantes à partir de modèles utilisés pour décrire l'évolution dans le temps d'un système dynamique.

En utilisant ces modèles, nous avons découvert que l'interaction entre les stratégies d'exploration provoque une diminution de l'entropie de la recherche dans l'espace de solutions; ce qui signale un niveau d'organisation des méthodes par recherches multiples coopérantes supérieur à celui des recherches multiples. Notre analyse, à partir de méthodes empruntées à la théorie des systèmes dynamiques, nous permet d'affirmer que les procédures parallèles avec recherches multiples coopérantes manifestent un comportement d'auto-organisation de l'exploration de l'espace de solutions. Notre travail montre cependant que cette organisation de la recherche qui

résulte de l'interaction entre les stratégies d'exploration n'obéit pas nécessairement à la logique d'exploration de la méthode séquentielle de recherche. Cette interaction semble définir un autre paysage (landscape) dont on peut faire une approximation par une loi de la dynamique d'interaction entre les variables d'un système dynamique. Il apparaîtra de manière évidente que ce paysage interfère avec la logique d'optimisation des recherches séquentielles, ce qui semble être l'indication d'un deuxième niveau de calcul, un niveau de calcul émergent pour les recherches multiples coopérantes. Nos résultats contredisent une hypothèse sous-jacente aux modèles probabilistes de l'interaction entre les procédures séquentielles selon laquelle ce paysage serait uniforme et sans impact sur l'exploration de l'espace de solutions effectuée par la recherche parallèle.

Chapitre 2
Concepts du traitement parallèle

2.1 Introduction

Les ordinateurs sont apparus avec la promesse d'ambitieuses réalisations mais les limites, autant de la théorie que de la technologie, font que certaines de ces réalisations progressent à un rythme plus lent que prévu. C'est le cas par exemple de la notion de calcul parallèle qui date de la fin des années 50, cependant jusqu'à tout récemment, les ordinateurs n'étaient capable d'exécuter qu'une seule instruction à la fois. Les ordinateurs contemporains sont encore presque tous basés sur ce qu'on appelle des *architectures algorithmiques*, c'est-à-dire que les problèmes sont résolus en formulant des algorithmes qui sont directement traduits dans des instructions machine. Or nous savons depuis les travaux de Pitts & McCulloch [70] dans les années 40 et de Rosenblatt [80] dans les années 50 que ce n'est pas la seule manière de faire du calcul et dans certain cas la plus efficace, du moins notre cerveau est capable de solutionner certains problèmes plus efficacement que les super-ordinateurs.

Malgré des progrès importants faits en informatique, il existe encore des domaines d'application où nos meilleurs super-ordinateurs n'arrivent pas à satisfaire la demande en capacité de traitement. Les problèmes d'optimisation combinatoire constituent un domaine d'applications de l'informatique où les demandes en capacité de calcul ne sont pas encore satisfaites et où il faut faire appel à tous les raffinements de l'informatique pour obtenir de meilleurs taux d'efficacité. Le traitement parallèle fait certainement partie de ces raffinements qui permettent d'augmenter les performances du calcul par ordinateur.

Le présent chapitre introduit certains résultats de la théorie classique du

calcul, bon nombre de concepts qui sont apparus depuis le début des années 60 sur l'architecture des ordinateurs, quelques notions sur les modèles de programmation et d'algorithmique parallèle, quelques considérations sur des architectures d'ordinateur non algorithmiques et enfin les métriques habituelles portant sur la mesure des performances des algorithmes parallèles appliquées au calcul numérique.

L'élaboration portant sur les différentes architectures d'ordinateurs parallèles de même que certaines considérations relativement à la théorie du calcul pourront ne pas sembler complètement nécessaires à la compréhension des résultats de la présente recherche. Cependant, ce travail porte sur des méthodes de résolution et des stratégies de parallélisation qui sortent du cadre restreint des méthodes numériques auxquelles nous sommes habitués. Nous avons pensé qu'il était nécessaire de rappeler certains résultats fondamentaux dans le prolongement desquels se situe la présente recherche. Deuxièmement, l'architecture immuable pendant plus de 30 ans de l'ordinateur séquentiel nous a fait oublié qu'il existe un lien très étroit entre la conception des méthodes de résolution algorithmiques et l'architecture des ordinateurs. L'émergence des systèmes de calcul complexe que sont certains ordinateurs parallèles nous a fait redécouvrir l'étroitesse de ce lien entre la théorie et la conception d'algorithmes (parallèles) d'une part, et l'environnement technologique d'autre part. Nous avons donc fait un effort pour introduire les notions de l'architecture des ordinateurs qui nous permettent de comprendre les grands paradigmes de l'algorithmique parallèle applicables aux méthodes de résolution heuristiques.

Ce chapitre est organisé de la manière suivante. Nous introduirons d'abord brièvement les concepts de la théorie du calcul. Par la suite, nous introduirons des notions reliées à l'architecture de la machine de von Neumann suivies de la présentation des principaux modèles d'architectures parallèles. Nous poursuivrons avec plusieurs notions sur la programmation de certaines classes d'ordinateurs parallèles. Enfin nous ferons une digression sur les architectures non algorithmiques d'ordinateurs parallèles et nous terminerons avec certains modèles pour la mesure

des performances des algorithmes parallèles.

2.2 Environnement formel du calcul

Les méthodes algorithmiques élaborées pour la résolution de problèmes par ordinateurs ne peuvent généralement pas être traitées directement par une machine réelle. Ces procédures sont conçues à partir d'un schéma théorique du calcul, schéma qui lui-même découle de modèles extrêmement simplifiés de l'architecture des ordinateurs. La présente section a pour objectif d'introduire quelques éléments de ce schéma théorique reliés aux trois aspects suivants du traitement par ordinateur: les modèles de calcul qui constituent les postulats de la machine algorithmique (une architecture mathématique de l'ordinateur), les modèles de programmation (abstractions qui permettent d'avoir une vue simplifiée et cohérente de la programmation des ordinateurs) et la notion d'algorithme (qui est parfois définie de manière confuse dans la littérature).

2.2.1 Modèles de calcul

Un modèle de calcul est une abstraction mathématique d'une machine capable d'exécuter des algorithmes. Les modèles de calcul constituent un pré-requis essentiel pour pouvoir faire l'étude de la complexité des algorithmes et c'est la raison principale de leur étude en informatique. Un objectif secondaire qui découle cependant de cette formalisation est l'obtention de définitions précises pour certaines classes d'algorithmes et d'ordinateurs. C'est précisément l'usage que nous souhaitons faire des modèles de calcul dans ce travail.

Il existe un grand nombre de modèles de calcul, la machine de Turing [98] est l'un de ces modèles. Nous nous limiterons cependant à définir certaines caractéristiques communes à la plupart de ces modèles. Dans presque tous les modèles, on retrouve les éléments suivants: un *processus de contrôle* qui est activé par un *programme* et qui exécute des opérations sur une structure appelée *mémoire*. Parfois

le processus de contrôle réfère à la notion de *processeur* exécutant un programme. En tout temps le processus de contrôle se trouve dans un *état* donné, le nombre d'états étant fini. La mémoire est vue comme étant une structure régulière de cellules qui emmagasine des données. Un programme pour cette machine abstraite est une suite d'opérations exécutables par le processus de contrôle, c'est-à-dire des opérations pour lire, écrire, modifier, tester le contenu des cellules de la mémoire et effectuer des branchements dans le programme.

Une *configuration* est une description complète de l'état de la machine, c'est-à-dire l'instruction courante dans le programme et le contenu de la mémoire qui a participé au traitement depuis le début du programme jusqu'à l'instruction courante. Une *relation de transition* entre deux configurations C_1 et C_2, $C_1 \vdash C_2$, existe si l'instruction du programme qui se trouve dans la configuration C_1 provoque un changement de configuration de la machine de C_1 vers C_2.

Le terme *calcul* réfère à des séquences de configurations connectées par la relation $\vdash$. Une machine est dite *déterministe* si pour chaque configuration C_1 il existe au plus une configuration C_2 telle que $C_1 \vdash C_2$, tandis qu'une machine est *non déterministe* s'il peut exister plusieurs configurations C_2. La source du non-déterminisme provient du programme, où l'instruction courante pourra correspondre à un choix de plusieurs instructions pouvant être exécutées à partir d'une configuration donnée.

Dans la précédente formalisation, il n'existe qu'un seul processus de contrôle connecté à la mémoire, ce qui implique que la machine n'exécutera qu'une seule instruction à la fois, c'est le *modèle d'une machine séquentielle*. Un *modèle d'une machine parallèle* correspond à la situation où il existe plusieurs processus de contrôle chacun ayant un accès direct à la mémoire. Une *relation de transition globale* dans une machine parallèle est obtenue par l'effet combiné des transitions de chaque processus de contrôle actif de la machine parallèle. Un *calcul synchrone* réfère à des séquences de configurations obtenues par des transitions globales effectuées par

les processus de contrôle exécutant des opérations identiques en même temps. Un *calcul asynchrone* réfère à des séquences de configurations obtenues par des transitions globales effectuées par les processus de contrôle exécutant des opérations différentes sans consensus entre les processus de contrôle sur le temps d'exécution. Les processeurs peuvent communiquer directement via un *réseau d'interconnexion* ou indirectement par une *mémoire partagée*. Pour un survol plus complet des principaux concepts de la théorie du calcul relatif au traitement séquentiel et parallèle, le lecteur est invité à consulter les articles de synthèse suivant: Boas [16], Karp [60] et Lamport [64].

2.2.2 La notion d'algorithme

Knuth [61] a défini de la manière suivante la notion d'*algorithme*: "...un ensemble fini de règles qui donnent une suite d'opérations afin de résoudre un type spécifique de problème". À l'époque où Knuth nous a donné cette définition, la conception des ordinateurs s'inspirait presqu'uniquement d'un seul modèle d'architecture, le modèle de von Neumann. Selon cette architecture, le cycle d'exécution d'une machine consiste à obtenir une instruction en mémoire, la décoder, obtenir les opérandes de l'instruction, les décoder et finalement exécuter l'instruction. C'est le modèle séquentiel du traitement par ordinateur. C'est sans doute à cause de l'apparente immuabilité de cette architecture que Knuth fut entraîné à faire une généralisation incorrecte dans sa définition de la notion d'algorithme. Une méthode de résolution d'un problème exige normalement au plus un "ordonnancement partiel des opérations" et non pas à un ordonnancement strict comme semblerait l'indiquer le terme "suite".

À l'insu de la plupart des chercheurs de cette époque, la notion d'algorithme contenait des idiosyncrasies reliées à une architecture particulière d'ordinateur. Avec l'avènement des architectures parallèles, ces idiosyncrasies peuvent conduire à l'existence d'autant de définitions de la notion d'algorithme qu'il y a de types

d'ordinateurs parallèles. Il faut donc, dans un premier temps, faire un effort d'abstraction et définir une notion d'algorithme qui soit cohérente avec les méthodes de résolution et non pas avec les architectures d'ordinateurs.

Définition 2.1 *Nous dirons qu'un algorithme est l'expression logique d'une méthode de résolution qui obtient un résultat s'il existe une solution, en un nombre fini, minimal et essentiel d'étapes.*

Une procédure de résolution peut être représentée par un graphe orienté acyclique $G = \{N, A\}$ où N est un ensemble fini d'opérations utilisées pour la résolution du problème, et A est un ensemble fini de relations tel que $(u, v) \in A$ si le résultat de l'opération u est une opérande de l'opération v. Le graphe G exprime l'aspect logique du processus de résolution en spécifiant un ordonnancement partiel de l'application des opérations à un input x correspondant à la donnée du problème à résoudre [15]. Cet ordonnancement partiel et l'ensemble N des opérations identifient de façon unique la fonction à être calculée, c'est-à-dire l'algorithme.

L'ensemble N des opérations d'un algorithme peut varier. Dans le cas du calcul numérique, les primitives sont souvent des opérateurs mathématiques. Dans certain domaine de l'intelligence artificielle, on trouve des opérations qui correspondent à des règles reflétant la connaissance du domaine que possède le système. On exprime aussi parfois les algorithmes à l'aide de primitives en pseudo-code, qui sont des abstractions de primitives empruntées aux langages de programmation. De même, pour un même problème, il peut exister plus d'un ordonnancement partiel des opérations: c'est le cas par exemple entre l'algorithme classique de multiplication de deux entiers et cet autre algorithme dit "de la multiplication à la *russe*" [18]. La diversité des architectures d'ordinateurs, les performances théoriques (asymptotiques) ou réelles en temps de calcul et l'aisance d'implantation sont quelques uns des facteurs qui conduisent à l'utilisation de plusieurs algorithmes différents pour la résolution d'un même problème.

2.2.3 Modèles de programmation

Il existe des langages de programmation et des architectures d'ordinateurs pour lesquels la seule spécification logique de l'algorithme suffit pour que celui-ci puisse être traduit en un programme exécutable par ordinateur. C'est le cas pour certains langages fonctionnels, et pour les architectures d'ordinateurs à flot de données. Ce n'est pas la règle cependant, la plupart du temps la traduction d'un algorithme dans un langage de programmation passe par l'introduction d'opérations explicites ou implicites. Ces opérations reflètent le modèle de programmation auquel appartient le langage, l'architecture de l'ordinateur cible et la réalisation physique du processus utilisé pour le traitement. Ces opérations appartiennent à trois catégories principales: les opérations reliées à l'ordonnancement des opérations (boucle, conditions, ordonnancement implicite dû aux séquencements des instructions), celles concernant le stockage des données (écriture, lecture) et celles reliées à la manipulation de périphériques (lecture ou écriture de données sur un disque ou communications inter-processeurs) [15].

Bien que ces opérations reliées à l'implantation du processus physique de traitement s'ajoutent à la spécification logique d'un algorithme, ces primitives ne correspondent que très rarement à des opérations de base d'un ordinateur. En fait nous concevons et analysons nos algorithmes (programmes) pour un *modèle de programmation*, c'est-à-dire des abstractions qui facilitent la programmation des ordinateurs en cachant de nombreux détails de l'architecture physique. Ceci a permis l'utilisation pendant des décennies d'un même modèle d'architecture pour la conception et l'analyse des algorithmes et ce malgré une évolution considérable des ordinateurs pendant toute cette période. Mais ces abstractions sont utiles uniquement si elles permettent de concevoir des programmes efficients sur des machines réelles [88]. Comme nous le verrons, l'avènement de certaines classes d'ordinateurs parallèles ont, de ce point de vue, rendu désuet les modèles traditionnels et forcé la création de nouvelles abstractions au niveau des modèles de programmation.

2.3 Architectures des ordinateurs

On divise les architectures d'ordinateurs en deux grandes catégories selon que l'ordonnancement des opérations exécutées par l'ordinateur est basé principalement sur des dépendances de flot de contrôle ou sur des dépendances de flot de données. Dans cette section, nous introduirons brièvement ces deux grands principes de la conception des ordinateurs puisqu'ils sont à la base de presque tout l'édifice du traitement parallèle actuel. Nous décrirons d'abord l'architecture de von Neumann qui était jusqu'à tout récemment le modèle standard d'architecture. Puis nous ferons une brève incursion au niveau des architectures de flot de données. Nous nous intéresserons ensuite plus en détail aux architectures basées sur le flot de contrôle puisque ce type d'architecture se retrouve dans pratiquement tous les ordinateurs parallèles contemporains.

2.3.1 L'architecture de von Neumann

Le modèle d'architecture de von Neumann que nous présenterons ici est une grande simplification par rapport au fonctionnement réel des ordinateurs séquentiels, mais il nous sera utile pour introduire l'architecture de certains ordinateurs parallèles dont la conception dérive parfois fortement de l'architecture de von Neumann. Dans le contexte des ordinateurs parallèles on réfère aussi à l'architecture de von Neumann par SISD (Single Instruction stream, Single Data stream). La figure 2.1 identifie les principales composantes de l'architecture de von Neumann de même que les relations entre ces composantes.

L'architecture de von Neumann gravite autour d'une structure bi-polaire qui comprend une composante active, le processeur, avec son unité de contrôle et son unité arithmétique et logique (UAL), et une composante passive qui consiste en un système de mémoire centrale où les instructions et les données sont stockées. L'unité de contrôle supervise tous les organes de cette machine en envoyant des séquences de signaux à l'intérieur du processeur et entre le processeur et la mémoire. L'UAL,

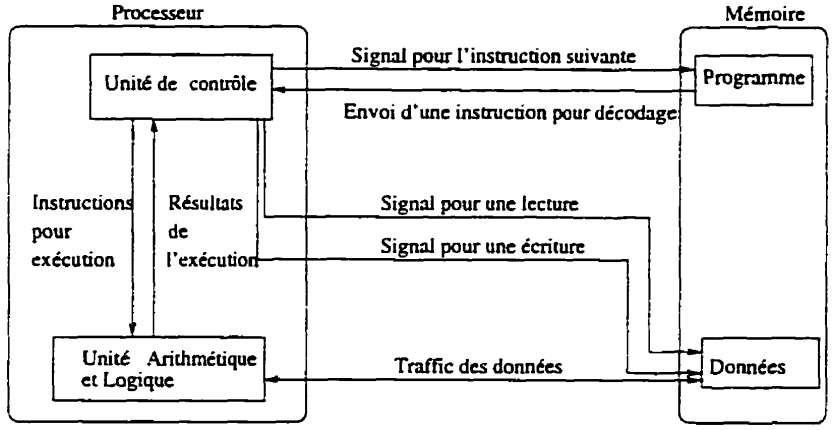


Figure 2.1 Architecture de von Neumann

qui a aussi un rôle actif, fait la transformation des données dans la mémoire. Le programme en mémoire représente la version codée, en termes d'instructions exécutables par l'ordinateur, de l'algorithme qui sera appliqué au problème se trouvant dans la section de données de la mémoire.

On dit des ordinateurs basés sur l'architecture de von Neumann qu'ils sont des ordinateurs de *flot de contrôle* (control flow computers) [58] car étant donné le fonctionnement de l'unité de contrôle et le faible couplage entre le processeur et la mémoire, le calcul s'exécute selon une *suite* d'instructions, c'est-à-dire de manière séquentielle. L'unité de contrôle envoie un signal à la mémoire pour obtenir une instruction, après décodage de l'instruction un ensemble de signaux est envoyé aux autres unités, dont la section de données de la mémoire, en vue d'obtenir les opérandes. Après l'exécution de l'instruction par l'UAL et la mise à jour de la mémoire, le cycle reprend avec la prochaine instruction.

Contraintes de l'architecture de von Neumann sur le modèle de programmation

Ce séquencement des instructions est parfois le résultat d'un artefact de l'architecture de l'ordinateur plutôt que la traduction d'une relation de dépendance de données

entre deux opérations au niveau de l'algorithme. Supposons que l'algorithme présente la relation ($OP_1 \xrightarrow{x} OP_2$) indiquant que l'opération OP_1 produit une opérande x utilisée par l'opération OP_2. On sait que conséquemment à cette relation, il y aura au niveau du programme au moins une instruction I_w d'écriture pour l'opération OP_1 et une instruction de lecture I_r pour l'opération OP_2. Dans ce cas, lors de l'exécution du programme, l'instruction I_w devra être exécutée avant I_r. C'est ce que l'on appelle les contraintes de *cohérence séquentielle* d'un programme [84]. Étant donné qu'un algorithme représente un ordonnancement partiel des opérations, il existe des situations où l'opérande x peut être produite par une troisième opération OP_3 et utilisée par OP_1 et OP_2. Dans ce cas, il importe peu dans quel ordre seront exécutées les opérations OP_1 et OP_2 l'une par rapport à l'autre. Cependant, étant donné l'organisation des différentes unités d'un ordinateur de von Neumann, un choix arbitraire d'ordonnancement des opérations OP_1 et OP_2 sera fait au niveau de la transposition de l'algorithme en un programme et parfois même au niveau du compilateur.

Avec le développement rapide de la technologie des micro-circuits, le cycle d'exécution des micro-opérations au niveau de la mémoire et de l'UAL a considérablement diminué. On en est venu à considérer que l'obstacle principal à l'accroissement de la puissance de calcul d'un ordinateur basé sur l'architecture de von Neumann était ce lien séquentiel entre la mémoire et le processeur. C'est ce que l'on a appelé le *goulot d'étranglement* des ordinateurs conçus sur le modèle de von Neumann. Puisqu'il existe au niveau des algorithmes un potentiel d'exécution concurrente des instructions, la recherche s'est orientée vers le développement de nouvelles architectures qui permettent l'exécution en parallèle de plusieurs instructions.

2.3.2 Architectures de flots de données

L'ordonnancement partiel des opérations au niveau des algorithmes permettant l'exécution concurrente de certaines instructions a donné naissance à la notion de *parallélisme logique* ou *degré de concurrence* présent dans un algorithme. Ce type de parallélisme expose le maximum d'instructions qui peuvent être exécutées concurremment à un instant donné du traitement. En fait, les restrictions au parallélisme au niveau du parallélisme logique sont uniquement dues aux dépendances de données qui existent entre les opérations. Ceci a inspiré les architectures *de flots de données* (data flow computers) où il n'existe aucun mécanisme de contrôle central comme c'est le cas pour la machine de von Neumann.

En théorie, les machines à flot de données sont capables d'exploiter tout le parallélisme logique présent au niveau d'une application. Pour ces ordinateurs, un programme tel que vu au niveau des instructions machine consiste en un "pool" d'instructions sans aucune forme d'ordonnancement entre les instructions. Lors de l'exécution du programme, c'est la disponibilité des opérandes qui fait qu'une instruction est exécutée (fire). L'ordonnancement des instructions se fait dynamiquement lors de l'exécution du programme sur la base de la disponibilité des données. Quelques projets reliés à ce type d'architecture sont bien connus: MIT *tagged-token dataflow architecture* [7], The Manchester Dataflow Computer [52] et la machine Sigma-1 au Japon [90].

Malheureusement, après avoir connu une vague de popularité dans les milieux académiques pendant les années 80 et début 90, les défis importants que posent ce type d'architecture au niveau physique et logiciel ont fait douter du succès commercial de cette approche. Les architectures de flots de données bousculent certains fondements de l'architecture des ordinateurs. On a dû repenser des concepts élaborés suite à plusieurs années d'expérience à concevoir des ordinateurs, d'où la dérive de complications qui s'est abattue sur ce type d'architecture. D'autres approches ont essayé de ne pas tout remettre en cause au niveau de l'architecture

des ordinateurs. Pour ces approches, les efforts se concentrent sur des méthodes d'organisation de plusieurs processeurs séquentiels pour faire du traitement parallèle, c'est le cas pour l'architecture de type SIMD.

2.4 Architecture SIMD

Comme mentionné à la section 2.2.2, un algorithme peut être représenté par un graphe orienté acyclique comme à la figure 2.2. Dans ce graphe, les noeuds représentent des opérations et les arcs des dépendances de flots de données entre les opérations. Au niveau 1, on voit que deux opérations produisent des données qui deviennent des opérandes pour toutes les opérations du niveau 2. Ces opérations au niveau 2 peuvent être exécutées de manière concurrente lorsque leurs opérandes deviennent disponibles. Au niveau du calcul numérique on a pu observer que dans certaines applications 80% du temps d'exécution séquentiel se passait dans 10% des instructions du programme, indiquant que plusieurs données sont transformées par la même opération. C'est ce que représente le niveau 2 où chaque noeud correspond à la même opération qui se répète sur des données différentes, par exemple une addition sur les éléments de deux vecteurs. Ce type de concurrence, où une seule opération ou séquence d'opérations s'applique de manière répétitive sur des données différentes, correspond au *parallélisme de données* d'un algorithme.

Cette observation selon laquelle certaines opérations se répétaient très souvent provoqua une modification de l'architecture de von Neumann consistant à remplacer l'unique UAL par une matrice de plusieurs UAL, comme on peut le voir dans la figure 2.3. Cette modification constitue la base de ce qu'on a appelé les architectures de type SIMD (Single Instruction stream, Multiple Data stream).

Une machine de type SIMD fonctionne comme un ordinateur séquentiel, ne décodant qu'une seule instruction à la fois sauf que l'unité de contrôle supervise une matrice d'UAL. On dit qu'un ordinateur de type SIMD est un ordinateur parallèle parce qu'au niveau du modèle de programmation, on fait appel à un ensemble

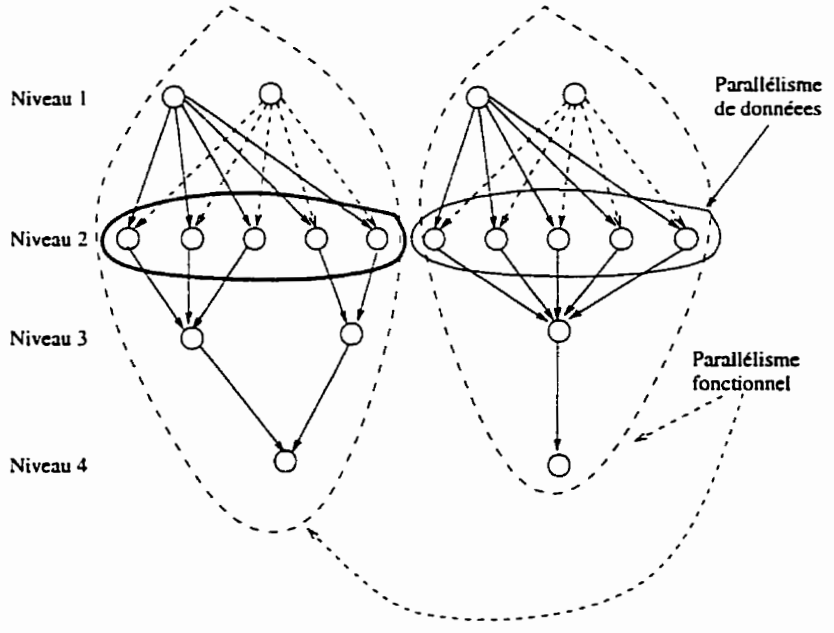


Figure 2.2 Parallélisme de contrôle et parallélisme de données

d'opérateurs dont la sémantique implique l'exécution d'un traitement simultané sur un grand nombre d'opérandes. On nomme opérateurs vectoriels ce type particulier d'opérations. On retrouve l'application de ces opérateurs pour le traitement de matrices ou encore de tableaux de données.

La Figure 2.3 montre la configuration de l'architecture de l'Illiac-IV, le premier ordinateur de type SIMD. Cette architecture comprend N unités de traitement (UT) synchrones sous la supervision d'une seule unité de contrôle (UC). Chaque unité de traitement est essentiellement une UAL avec un certain nombre de registres et une mémoire locale pour le stockage des données distribuées (ne contient aucune instruction). L'unité de contrôle possède sa propre mémoire pour le stockage des programmes. Les programmes des usagers sont exécutés sous le contrôle de l'UC. Le rôle de l'unité de contrôle est de décoder toutes les instructions et de déterminer sur quelle UT les instructions décodées doivent être exécutées. Les instructions de contrôle (comme par exemple les instructions de branchement) et les instructions

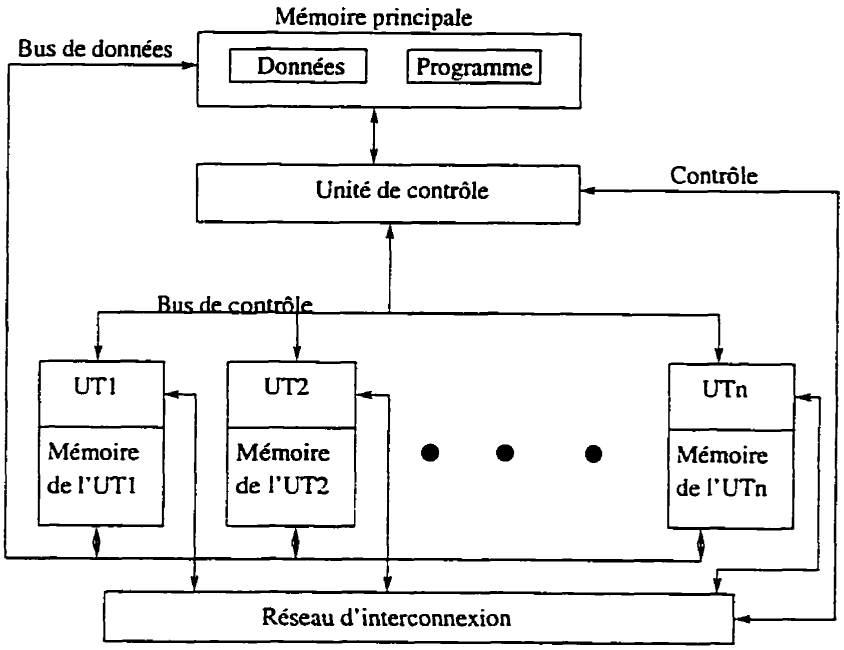


Figure 2.3 Architecture de l'Illiac-IV

de type scalaire sont exécutées directement à l'intérieur de l'UC. Les instructions de type vectoriel sont envoyées aux UT pour être exécutées en parallèle. Toutes les UT exécutent la même instruction de façon synchrone sous le contrôle de l'UC. Les opérandes sont distribuées à la mémoire des UT avant que l'exécution en parallèle des unités de traitement ne commence. Les données peuvent être chargées en mémoire des UT soit à partir d'une source externe par le bus de données du système, soit à partir de l'UC en mode de diffusion en utilisant le bus de contrôle.

Une autre configuration typique des architectures SIMD consiste à remplacer l'UC par un ordinateur séquentiel de type von Neumann, c'était la configuration de la Connection Machine CM-2. Les ordinateurs MPP, DAP, MasPar MP-1 et la MasPar MP-2 sont d'autres exemples avec l'Illiac-IV et la CM-2 d'ordinateurs parallèles basés sur l'architecture SIMD.

Domaine d'application et limites des architectures SIMD

Les ordinateurs de type SIMD sont un bon outil pour l'exécution en parallèle de méthodes numériques portant sur le traitement de matrices (la multiplication matricielle, la transposition de matrice, l'inversion de matrice, différentes opérations sur des matrices booléennes) la sommation des éléments d'un vecteur, le tri, les récurrences linéaires, et pour résoudre des équations différentielles [58]. En ce qui concerne les algorithmes et les programmes pour ces machines, il faut en général déployer beaucoup d'ingéniosité pour le chargement des données dans les mémoires des UAL et pour minimiser la distance de parcours entre les UAL des données générées durant le traitement.

Les ordinateurs de type SIMD ont connu du succès lorsque leur conception visait dès le départ un champ d'application assez restreint: on dit que ce sont des ordinateurs pour des applications spécifiques (special purpose computers). On construit encore de ces ordinateurs par exemple pour des applications en temps réel où le parallélisme est la seule façon de pouvoir respecter les contraintes de temps. Cependant la transition de la CM-2 à la CM-5 semble bien indiquer que les concepteurs d'ordinateurs parallèles ont abandonné l'idée d'utiliser ce type d'architecture comme outil pour appliquer le traitement parallèle à un large éventail de problèmes. On s'est plutôt tourné vers des architectures de type MIMD.

2.5 Architecture MIMD

Si l'on retourne à la figure 2.2, on remarque que les boucles en pointillées regroupent deux séquences d'opérations ne possédant aucune dépendance de données entre elles. Ces deux séquences d'opérations peuvent s'exécuter de manière concurrente. On réfère à ce type de parallélisme comme étant du *parallélisme de contrôle.*

Le parallélisme de contrôle constitue une source importante de parallélisme pour de nombreuses applications en dehors du calcul numérique (qui sont souvent des

applications moins structurées que le traitement matriciel). Cependant, la nécessité d'exécuter de manière synchrone la même instruction sur tous les processeurs rend difficile l'exploitation du parallélisme de contrôle avec les ordinateurs basés sur l'architecture SIMD. Cette constatation a provoquée l'avènement d'un autre type d'architecture d'ordinateurs parallèle où cette fois l'unité de contrôle et l'unité de traitement sont dupliquées et où par conséquent les processeurs fonctionnent de manière indépendante les uns par rapport aux autres. Une telle architecture correspond en gros à plusieurs ordinateurs capables chacun d'exécuter son propre programme. Flynn [38] réfère à ce type d'architecture par MIMD (Multiple Instructions stream, Multiple Data stream).

2.5.1 L'approche multi-processeurs

Selon le degré d'intégration de leurs composantes, les ordinateurs de type MIMD se divisent en deux catégories: les systèmes à mémoire partagée et les systèmes à mémoire distribuée. La figure 2.4A correspond à un modèle d'architecture de type MIMD où chaque processeur est constitué d'une unité de contrôle et d'une unité arithmétique et logique reliée à une mémoire centrale par différents types de réseaux d'interconnexion: soit le bus, le réseau multi-étages ou la commutation par circuits. On dit de cette architecture MIMD qu'elle est un système *étroitement couplé* (tightly coupled) ou encore un système *multi-processeurs*. Le haut degré d'intégration de ce type de machine MIMD vient du fait qu'il existe une seule mémoire (ou un ensemble de modules de mémoire) qui peut être *accédée directement* par chaque processeur à travers un réseau d'interconnexion extrêmement rapide. En effet, le bus est un circuit intégré qui permet d'avoir un réseau d'interconnexion rapide et peu coûteux. Cependant, il ne permet qu'une seule référence à la fois à la mémoire et donc se sature rapidement lorsque le nombre de processeurs augmente. Le réseau multi-étages permet plusieurs accès simultanés à la mémoire et diminue donc le temps d'attente des processeurs. La commutation par circuits est le réseau d'interconnexion

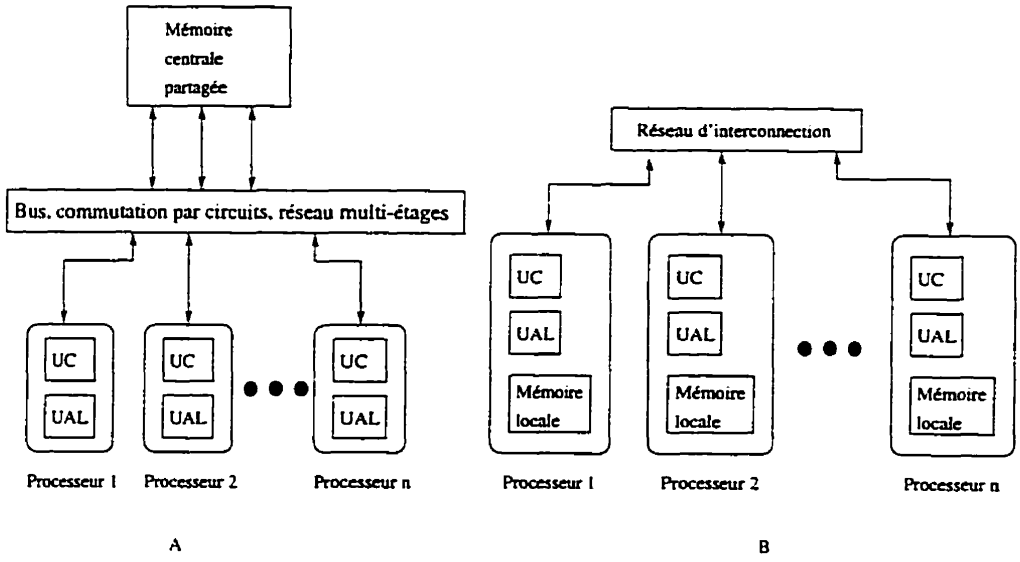


Figure 2.4 A) Architecture MIMD avec mémoire partagée, B) Architecture MIMD avec mémoire distribuée

le plus efficace en terme de bande passante mais son coût croît par un facteur de p^2 en fonction du nombre p de processeurs, en comparaison avec $p \log p$ pour les réseaux multi-étages. Les ordinateurs parallèles CMU/C.mmp, Illinois Cedar, NYU/Ultracomputer, Stanford/DASH, Fujitsu VPP500 et KSR1 appartiennent à la classe des multi-processeurs.

On dit qu'une mémoire est accédée directement lorsqu'une référence à une variable dans un programme est transformée immédiatement en un signal envoyé au réseau d'interconnexion pour accéder à une adresse en mémoire centrale. Chaque processeur d'un système MIMD étroitement couplé possède cette même capacité. De plus, pour ces systèmes à mémoire partagée, il n'existe qu'un seul espace d'adressage, c'est-à-dire qu'une référence à une variable globale se transforme en une même adresse mémoire peu importe le processeur qui effectue la référence. On dit qu'une telle variable est une *variable partagée* entre les différents processeurs. Les processeurs utilisent cette caractéristique de la configuration multi-processeurs comme

moyen de communication, c'est-à-dire pour s'échanger de l'information. Malheureusement la faible bande passante ou les coûts exorbitants du réseau d'interconnexion font que l'architecture MIMD basée sur la mémoire partagée n'est pas une alternative viable pour la construction de machines parallèles avec des milliers de processeurs.

2.5.2 L'approche multi-ordinateurs

Pour remédier à cette difficulté, on a développé une configuration de l'architecture MIMD où chaque processeur possède sa propre mémoire locale et où il n'y a pas en théorie de mémoire globale. C'est la configuration que l'on voit dans la figure 2.4B où chaque processeur ne peut accéder directement que sa mémoire locale. On appelle cette configuration *multi-ordinateurs* ou *système à mémoire distribuée* parce qu'elle correspond à un système avec plusieurs ordinateurs autonomes communiquant entre eux par un réseau à faible débit. Dans cette configuration, il n'y a pas de variable visible au niveau de plus d'un processeur. Lorsqu'un processeur p_i a besoin d'une information qui ne se trouve pas dans sa mémoire locale, il doit en faire la demande explicitement au processeur p_j qui possède cette information. Le processeur p_j fait alors une référence directe à une adresse de sa mémoire locale pour obtenir l'information et retourne celle-ci au processeur p_i qui a fait la requête. Pour cette raison on dit que les processeurs dans cette configuration communiquent par *passage de messages*. Les ordinateurs parallèles Cosmic Cube, nCube-2/6400, Intel iPSC, Mosaic, Intel Paragon et MIT/J machine appartiennent à cette classe des multi-ordinateurs.

L'approche distribuée permet effectivement d'intégrer un grand nombre de processeurs dans un même système. Il y a deux problèmes cependant:

- Le transfert d'information par passage de messages demande une intervention au niveau logiciel et se fait sur un réseau plus lent (même aujourd'hui) que l'interconnexion sur un bus;
- Les systèmes distribués sont plus compliqués à programmer puisque le

programmeur doit être conscient de chaque échange d'information et en faire la spécification en terme d'instructions au niveau de son programme.

On ne rencontre pas ces problèmes avec un système multi-processeurs étant donné que l'environnement physique et logiciel (le système d'exploitation) permet de résoudre automatiquement toutes les références à des variables sans intervention du programmeur, rendant ce type d'ordinateur plus facile à programmer.

La lenteur des communications des multi-ordinateurs fait que ces derniers sont efficaces lorsque le degré d'interaction entre les processeurs est faible, tandis que les multi-processeurs peuvent tolérer un plus haut degré d'interaction.

2.6 Modèle de programmation des ordinateurs MIMD

La composante de base d'une architecture MIMD étant un processeur de type von Neumann, le modèle de base pour la programmation de ces ordinateurs parallèles est similaire à celui des ordinateurs séquentiels. Chaque processeur d'une machine MIMD exécute un sous-programme séquentiel que nous identifierons par le terme *processus* et nous désignerons comme étant une *tâche* la partie d'un algorithme traitée par un processus.

Définition 2.2 *L'unité de traitement de base d'un ordinateur MIMD est le processus et un programme dans ce contexte est une collection de processus [57].*

Définition 2.3 *La taille du grain ou granularité est une mesure de la quantité de traitement impliqué dans un processus.*

On peut évaluer la quantité de traitement d'un processus en terme du nombre d'instructions. On a l'habitude de distinguer trois degrés de granularité des processus: *fin*, *moyen* et *large*.

Sources de parallélisme et niveaux de programmation

Le parallélisme peut-être exploité à différents niveaux de programmation. Au niveau des instructions, une taille de grain typique contient moins de 20 instructions [57]: il s'agit de parallélisme à grain fin. Les ordinateurs à flot de données et les architectures pipelines exploitent ce niveau de parallélisme. On peut exploiter le parallélisme au niveau des boucles dans un programme. Une boucle typique contient environ 500 instructions machine. C'est souvent à partir de ces structures de contrôle d'un programme que la parallélisation pour les machines SIMD et MIMD se réalise. Il s'agit de parallélisme aussi de taille fine. Pour ce type de parallélisme, on utilise souvent des compilateurs pour découvrir le parallélisme. Le parallélisme peut être exploité au niveau des procédures, des sous-programmes ou des programmes. On parle de taille de grain allant de moyen à large, souvent implanté par passage de messages avec des multi-ordinateurs.

Stratégies pour la création de tâches parallèles

Lorsqu'on destine l'exécution d'un algorithme à une machine de type MIMD, on crée rarement les tâches à partir du graphe logique de l'algorithme. On code plutôt l'algorithme à l'aide d'un langage de programmation quelconque et les tâches sont identifiées à partir de ce programme. Il existe deux stratégies possibles pour définir le contenu d'une tâche: soit qu'on considère une séquence d'opérations liées entre elles par une suite de dépendances de données; soit qu'on considère un ensemble de données liées entre elles par l'exécution de la même séquence d'opérations. Dans le premier cas, on dit qu'on fait une décomposition du programme en fonction des structures de contrôle, *décomposition fonctionnelle*, dans le deuxième cas, on dit que la décomposition se fait en fonction des structures de données, c'est-à-dire une *décomposition du domaine*. La décomposition fonctionnelle correspond à l'exploitation du parallélisme de contrôle, tandis que la décomposition du domaine correspond à l'exploitation du parallélisme de données. Bien que dans la littérature,

on a pris l'habitude de dissocier ces deux modes de décomposition, il s'avère dans la pratique que la programmation des machines MIMD se fait en alternant ces deux perspectives.

Programmation de type SPMD

Les modèles de programmation qui dérivent d'une architecture particulière peuvent parfois servir de modèle de programmation pour une autre catégorie d'architectures; c'est le cas pour le parallélisme de données. Bien qu'en principe l'architecture des machines MIMD favorise l'exploitation du parallélisme de contrôle, il est tout à fait possible d'exploiter le parallélisme de données en distribuant le domaine des itérations d'une boucle sur plusieurs processeurs. La synchronisation globale de l'exécution de chaque instruction étant un artefact de l'architecture physique des machines SIMD, on relâche la fréquence de synchronisation des instructions pour obtenir un modèle de programmation de type SPMD (Single Program, Multiple Data). Dans ce cas, chaque processeur exécute la même séquence d'instructions de manière asynchrone, c'est-à-dire sans tenir compte de l'évolution de l'exécution sur les autres processeurs (par contre, le parallélisme de contrôle qui réfère à l'exécution simultanée de plusieurs séquences différentes d'instructions ne s'implante pas facilement sur une machine de type SIMD).

2.6.1 Interaction entre les tâches d'une architecture multi-processeurs

On a cru pendant un certain temps pouvoir développer un ordinateur de type MIMD capable d'exploiter de manière optimale toutes les formes que peut prendre le parallélisme. On a renoncé pour le moment à ce rêve. Le degré de parallélisme logique et la topologie du graphe de dépendances de données entre les opérations varient grandement d'une application à l'autre. Au lieu d'essayer d'adapter la machine à toutes les formes possibles de parallélisme, on tente plutôt d'adapter le parallélisme logique aux caractéristiques de la machine parallèle disponible. La

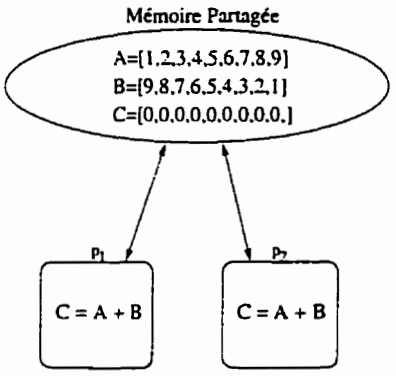


Figure 2.5 Exemple d'un programme SPMD

conséquence immédiate de cette adaptation forcée du parallélisme logique à l'architecture des ordinateurs est qu'il y a des dépendances entre les tâches, lesquelles dépendances se transforment en échange de données et en synchronisations entre les processus lors du traitement. C'est la raison pour laquelle on répète souvent que le traitement parallèle implique que plusieurs tâches peuvent s'exécuter simultanément, ce qui ne veut pas dire indépendamment! Nous allons maintenant examiner les difficultés de programmation des systèmes multi-processeurs engendrées par ces interactions entre les tâches au niveau des systèmes multi-processeurs.

Synchronisation de l'accès à la mémoire partagée

On l'a vu, les systèmes multi-processeurs exécutent l'échange d'information à l'aide de la mémoire partagée et de variables globales. Supposons que deux processeurs p_1 et p_2 doivent coopérer pour faire l'addition de deux matrices A et B dont le résultat serait placé dans une matrice C. Les trois matrices seront déclarées comme variables globales et deux processus identiques seront créés qui effectueront l'addition sur des entrées différentes des matrices A et B. Il s'agit alors d'une exploitation du parallélisme de données à l'aide du paradigme SPMD.

Pour qu'il y ait échange d'information au niveau de la mémoire partagée, il

faut qu'il y ait au moins un processeur qui exécute une opération d'écriture dans une zone mémoire d'une variable globale qui sera ensuite lue par un autre processeur. Dans l'exemple de la Figure 2.5, il n'y a pas un tel échange d'information puisque les deux processeurs écrivent dans la variable globale C, mais aucune lecture n'est faite de cette variable. Le programme se déroulera donc sans problème puisqu'il n'y a pas vraiment d'échange d'information entre les processus.

Supposons que le calcul de C soit suivi par une autre addition matricielle $D = B + C$ exécutée par un seul processeur p_3. Dans ce cas, il y a communication entre processeurs puisque les processeurs p_1 et p_2 écrivent dans les zones mémoire de la matrice C et que le processeur p_3 va lire les entrées de cette matrice. Pour que l'exécution respecte la sémantique du programme, il faut que $D = B + C$ s'exécute uniquement lorsque les deux processeurs p_1 et p_2 auront terminé l'addition matricielle $C = A + B$. Mais les machines de type MIMD sont asynchrones, c'est-à-dire qu'il n'y a pas de composante matérielle qui joue le rôle d'une horloge globale synchronisant les cycles d'exécution de chaque processeur. Conséquemment, il n'y a aucun moyen au niveau de l'architecture de l'ordinateur de garantir que l'exécution de $D = B+C$ commencera lorsque celle de $C = A+B$ sera complètement terminée. C'est le phénomène de "*race conditions*" où les processeurs compétitionnent entre eux pour terminer le plus rapidement sans égard à l'exécution correcte du programme. On dispose au niveau logiciel (et maintenant parfois matériel) de certains mécanismes comme les *barrières* (barrier) qui ont pour objectif de *synchroniser* l'exécution des processus pour qu'ils respectent la sémantique du programme. Le programmeur en introduisant une barrière au niveau du processus qui effectue l'addition $D = B + C$, pourrait forcer ce processus à attendre que les deux autres processus aient terminé.

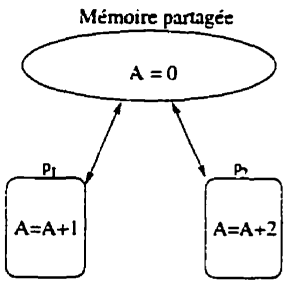


Figure 2.6 Race conditions et accès simultané

Contrôle de l'accès simultané à une variable partagée

La mémoire partagée introduit un autre problème au niveau des communications entre processeur, celui de l'accès simultané à une variable partagée. Supposons comme le montre la figure 2.6 que deux processeurs fassent la mise à jour d'une même variable A. Après l'exécution de ces deux processeurs, la valeur de A pourra être 1, 2, ou 3 en fonction de l'ordre de mise à jour de la variable A par les deux processeurs. Le résultat de cette exécution est non déterministe puisqu'avec le même input on pourra obtenir différents outputs. Par exemple, processeur p_1 lit A qui vaut 0 et fait l'addition $0 + 1 = 1$ mais entre temps le processeur p_2 a terminé son calcul $0 + 2 = 2$ et écrit 2 dans la mémoire partagée, puis le processeur p_1 écrase la valeur de A et la remplace par 1. Résultat après exécution de ces deux instructions: $A = 1$. Ce problème est résolu en synchronisant l'accès aux variables partagées à l'aide d'une *section critique* qui est un segment de code qui doit être exécuté par un seul processus à la fois, et qui, lorsque démarré, doit être complété sans interruption. Dans notre exemple, les instructions d'accès à la variable A seraient placées dans une section critique. Lorsqu'un processeur lit la variable A, l'autre processeur sera placé en attente jusqu'à ce que le premier processeur ait écrit le résultat de son opération en mémoire partagée. Ainsi on force ce programme à avoir un comportement déterministe.

2.6.2 Modes d'interaction entre les tâches parallèles

Le mode d'interaction entre les tâches parallèles est critique au point qu'il sert de critère pour la classification des algorithmes parallèles.

Définition 2.4 *Les* **points d'interaction** *sont des sections de code permettant aux processus d'interagir entre eux pour se synchroniser ou échanger des données.*

Les points d'interaction divisent un processus en étapes. À la fin d'une étape, un processus peut communiquer avec d'autres processus avant de commencer le calcul de la prochaine étape. À cause des points d'interaction, des processus peuvent être bloqués pour certaines périodes dans l'attente de réponses des autres processus.

Définition 2.5 *Les algorithmes parallèles où les processus doivent attendre d'autres processus sont appelés* **algorithmes synchrones**.

Les algorithmes exécutés sur une machine de type SIMD sont synchrones avec des étapes correspondant à une seule instruction. L'exploitation du parallélisme de données à travers le paradigme SPMD se fait presque toujours par des algorithmes synchrones lorsque le parallélisme utilise des données intermédiaires. Dans la littérature on qualifie également de semi-synchrones [43] les algorithmes de type SPMD étant donné que ce sont des algorithmes divisés en étapes asynchrones séparées par des points d'interaction pour la synchronisation. En règle générale les performances d'un algorithme synchrone sont fonction du processus le plus lent à chaque étape.

Définition 2.6 *Un* **algorithme parallèle asynchrone** *ou chaotique est un algorithme où les processus n'ont pas à s'attendre les uns les autres, les communications étant obtenues par la mise à jour dynamique de variables globales en mémoire partagée.*

Les processus asynchrones n'attendent jamais pour de l'information, mais continuent leur exécution ou se terminent selon l'information la plus récente qui se trouve dans

les variables globales. Les opérations sur les variables globales sont programmées à l'aide de sections critiques pour assurer le respect de la sémantique du programme.

En général, les programmes asynchrones s'exécutent plus rapidement que les programmes synchrones parce que l'exécution des processus n'est jamais bloquée pour attendre que les autres processus signalent leur arriver à un point d'interaction. Il peut cependant y avoir des délais reliés aux conflits pour l'accès concurrent des variables partagées pour ce type d'algorithme. Par contre la programmation est plus difficile avec les algorithmes chaotiques due à l'ordonnancement non déterministe durant l'exécution et au manque d'états globaux simples. En effet, étant donné que les processus peuvent utiliser de l'information qui n'est pas à jour, les tests d'exactitude (correctness) des programmes sont presque impossibles, et il y a des problèmes pour l'analyse de la complexité du temps de traitement de ce type d'algorithme. La résolution de systèmes d'équations linéaires par approximations successives [14] est un exemple d'application où on fait usage d'algorithmes asynchrones.

2.6.3 Communication par passage de messages

Au niveau de la configuration MIMD sans mémoire globale, l'échange d'information entre les processeurs s'effectue par passage de messages. Pour ce type de communication, les processus utilisent des instructions explicites comme les "send" et "receive" pour signaler l'envoi ou la réception d'information. Un passage de messages est dit synchrone si le processus qui envoie le message et celui qui le reçoit se synchronisent, c'est-à-dire que lorsqu'un des deux processus n'est pas prêt à communiquer, il sera placé en attente, c'est-à-dire bloqué. Dans ce sens, la communication synchrone est aussi appelée communication bloquante. En mode asynchrone, le processus qui envoie et celui qui reçoit un message n'ont pas à se synchroniser dans le temps.

2.7 Architectures non algorithmiques

La conception d'ordinateurs du style von Neumann a donné naissance par extension aux architectures parallèles de type SIMD et MIMD. Cependant ces architectures algorithmiques qui consistent à lancer plus de processeurs pour faire un calcul restent encore peu efficaces pour résoudre des problèmes dans des domaines comme la vision, la reconnaissance de la parole, les problèmes d'optimisation. Plusieurs scientifiques pensent qu'il y a quelque chose de fondamentalement inexact dans la façon dont nous approchons quelques uns de ces problèmes. Cette idée se voit confirmée par des exemples de systèmes dans la nature qui sont capables de trouver des solutions à certains problèmes qu'on ne sait pas encore résoudre avec nos plus puissants ordinateurs.

2.7.1 Architecture d'un réseau de neurones

Les réseaux de neurones représentent une approche radicalement différente au calcul, à la conception d'architectures d'ordinateur et à leur fonctionnement. On attribue l'origine des travaux sur les réseaux de neurones aux articles de McCulloch et Pitts [70] et de Rosenblatt [80]. Avec le temps, les propriétés et le comportement des réseaux de neurones utilisés pour le calcul par ordinateur se sont éloignés des préoccupations originales des vrais neurones de notre cerveau.

Un modèle d'architecture de réseau de neurones est construit à partir d'unités de base appelées neurones. Chaque neurone est relié à d'autres neurones par des arcs qui correspondent aux synapses des vrais neurones. À chaque arc (i, j) est associé un poids c_{ij} correspondant à l'effet synaptique (qui inhibe ou excite le neurone récepteur). À chaque neurone correspond un état, qui est calculé en faisant la somme des inputs (qui sont fonction des poids des arcs entrants), en soustrayant une valeur seuil (qui représente le niveau d'excitation minimum auquel le neurone répond), et en appliquant une fonction d'activation du neurone. On a donc les principaux paramètres suivants:

c_{ij} = poids du lien entre i et j

Θ_j = seuil

S = fct d'activation.

La transition d'état d'un neurone x_j au temps t vers le temps $t+1$ peut être caractérisée par une fonction d'activation:

$$x_j(t+1) = S(\sum_i c_{ij}x_i(t) - \Theta_j)$$

Le nombre de neurones, la topologie d'interconnexion des neurones, le poids des liens, le seuil d'activation de chaque neurone sont les caractéristiques de ce type d'architecture.

2.7.2 Modèle de programmation d'un réseau de neurones

Les réseaux de neurones se programment par un processus d'induction du comportement de l'ordinateur neuronal que l'on appelle *apprentissage*. L'apprentissage des réseaux de neurones se fait en donnant des valeurs aux paramètres c_{ij} et Θ_j à l'aide d'une sélection basée sur des *règles d'apprentissage*. Le principe de ces règles est très simple: l'efficacité d'une synapse se renforce lorsque les neurones reliés par cette synapse sont actifs ou passifs en même temps, cette efficacité se détériore dans le cas contraire. On peut implanter ce principe en faisant varier le poids de chaque arc en fonction du produit de l'activité des neurones du lien: $\Delta c_{ij} = c(x_i \times x_j)$.

L'apprentissage d'un réseau de neurones est obtenu à l'aide d'un vecteur v soumis en input au réseau et à la variation des paramètres c_{ij} et Θ jusqu'à ce que l'ensemble des états des neurones se stabilise autour d'un vecteur (prototype) v'. On dit alors qu'on a calculé les efficacités synaptiques du système tel qu'il sera capable de reconnaître v' si un vecteur semblable est soumis en input. La phase d'apprentissage est alors terminée. L'ajustement de ces paramètres c_{ij} et Θ doit se faire de tel sorte que le réseau puisse se stabiliser après un certain nombre d'itérations de ses neurones

et non pas osciller à l'infini. Le processus par lequel le calcul au niveau d'un réseau de neurones s'effectue dépend de cette convergence vers un état stable.

2.7.3 Fonctionnement d'un réseau de neurones

Une fois la phase d'apprentissage terminée, un réseau de neurones fonctionne de la façon suivante: supposons un réseau de n neurones, où chaque neurone pour une itération donnée t est dans un état représenté par $x_i(t)$ qui peut prendre les valeurs +1 et -1. À l'itération t, chaque neurone reçoit différents signaux d'inputs en provenance de plusieurs autres neurones. Un neurone i se voit associé un potentiel $v_i(t)$ défini par

$$v_i(t) = \sum_{j=1}^{n} c_{ij} x_j(t).$$

Chaque neurone calcule un signal d'output, c'est-à-dire l'état $x_i(t + \Delta t)$ du neurone i à l'itération $t + \Delta t$ est calculé à partir de $v_i(t)$ et du seuil Θ_i:

$$x_i(t + \Delta t) = \begin{cases} +1 \text{ si } v_i(t) > \Theta_i \\ -1 \text{ si } v_i(t) < \Theta_i \\ x_i(t) \text{ si } v_i(t) = \Theta_i \end{cases}$$

et ce signal d'output est transmis à d'autres neurones. Pour chaque neurone i, $v_i(t + \Delta t)$ est calculé à partir des valeurs de $x_j(t + \Delta t)$. Le système atteint alors un état stable $\underline{x}' = (x'_1, ..., x'_n)$ après quelques itérations. L'état final $\underline{x}'$ est fonction de l'état initial $\underline{x}$ à l'itération $t = 0$ du réseau de neurones, de la matrice $C = c_{ij}$ et du seuil Θ.

2.7.4 Comparaison avec les architectures algorithmiques

Les réseaux de neurones représentent donc un changement important par rapport aux architectures algorithmiques. Quelques-unes des caractéristiques qui distinguent les réseaux de neurones des autres ordinateurs sont:

- Un parallélisme massif; un réseau de neurones peut avoir des centaines de milliers de processeurs;
- Un rôle fondamental donné à la topologie d'interconnectivité des neurones en ce sens que les liens ne font pas que transmettre de l'information mais font aussi du traitement et du stockage d'information;
- Une représentation distribuée de l'information, c'est-à-dire que les données sont stockées sous forme de patterns de connections à travers tout le réseau de processeurs;
- Un traitement collectif de l'information, c'est-à-dire qu'un réseau de neurones ne traite pas d'instructions individuelles; tous les neurones du réseau contribuent de manière collective (synchrone ou asynchrone) à résoudre un problème;
- Capacité d'auto-organisation: un réseau de neurones peut adapter de manière autonome ses structures pour apprendre de nouveaux patterns.
- Absence d'unités de contrôle; le système a un contrôle distribué, chaque processeur exécute ses opérations de manière tout à fait autonome sans connaissance explicite de l'état global du système.

Les réseaux de neurones n'exécutent donc pas *d'instructions programmées* comme pour les ordinateurs conventionnels: ils répondent en parallèle à un stimulus qui est un pattern d'inputs soumis aux neurones responsables de faire l'interface avec l'environnement. Le calcul effectué par un réseau de neurones est apparent à travers les vagues successives d'activation des neurones et le résultat correspond à l'état dans lequel le réseau se stabilise.

Il n'existe pas encore d'architecture physique de réseau de neurones, mais ce modèle de calcul a été simulé sur des architectures algorithmiques. Ces simulations ont été utilisées jusqu'à maintenant avec succès pour traiter des problèmes où le nombre de variables est très grand, pour la découverte d'associations ou de régularités

entre patterns (vision et reconnaissance de la voix), pour des problèmes où les relations entre les variables ne sont que vaguement comprises (systèmes complexes, nonlinéaires) et pour des problèmes d'optimisation combinatoire.

2.8 Performances des algorithmes parallèles

Dans la présente section, nous introduirons quelques notions relatives à la mesure des performances des algorithmes parallèles.

Mesures de la vitesse d'exécution

Le temps d'exécution T_s d'un algorithme séquentiel est le temps écoulé entre le début et la fin de l'exécution du programme sur un ordinateur séquentiel. Le temps d'exécution T_p d'un algorithme parallèle est le temps écoulé entre le moment où l'exécution parallèle débute et le moment où le dernier processeur termine son exécution. Un *système parallèle* est la combinaison d'un algorithme parallèle et de l'architecture parallèle utilisée pour exécuter l'algorithme. Pour évaluer les performances d'un système parallèle, on cherche à connaître le gain obtenu en terme du temps d'exécution de l'implantation parallèle d'une application par rapport à son implantation séquentielle. L'*accélération* (speedup) $S = \frac{T_s}{T_p}$ est une mesure de ce gain basée sur le ratio du temps T_s pris pour résoudre un problème par un seul processeur avec le meilleur algorithme séquentiel connu versus le temps T_p pour résoudre le même problème par un ordinateur parallèle à l'aide de p processeurs identiques. L'accélération est dite linéaire si $T_p = \frac{T_s}{p}$, mais en pratique il est difficile d'obtenir cette accélération. L'*efficacité* $E = \frac{S}{p}$ est une mesure de la fraction du temps parallèle écoulé T_p pour lequel un processeur est utilisé à faire du traitement. Dans un système parallèle où l'accélération est $S = p$, l'efficacité est de 1, mais en pratique l'accélération est inférieure à p et l'efficacité est inférieure à 1. Le *coût* ou le *travail* $C = T_p \times p$ pour résoudre un problème avec un système parallèle représente le temps parallèle multiplié par le nombre de processeurs.

Efficacité en fonction de la taille du problème et du nombre de processeurs

L'accélération ne s'accroît pas de manière linéaire en fonction du nombre de processeurs parce que possiblement les sections d'un algorithme peuvent avoir un degré variable de parallélisme logique. Ceci inclut la possibilité que pour certaines sections il n'y ait aucun parallélisme. Cette observation est à l'origine de ce qu'on a appelé la *loi d'Amdahl* [6] selon laquelle l'accélération est bornée par la partie séquentielle d'un programme. Lorsque ce phénomène est en cause, il est un des facteurs responsables de la diminution de l'efficacité en fonction de l'augmentation du nombre de processeurs. Il existe cependant une contre-argumentation à la loi d'Amdahl selon laquelle une augmentation de la taille du problème peut résulter en une diminution du poids relatif des sections séquentielles du programme [53]. C'est effectivement le cas pour plusieurs problèmes où on obtient alors un accroissement de l'accélération et de l'efficacité d'un système parallèle. L'*extensibilité* (scalability) est une mesure de la capacité d'un système parallèle de maintenir la même efficacité en accroissant à la fois le nombre de processeurs et la taille du problème.

Sources de dégradation des performances

Si l'on fait abstraction du phénomène relatif à la loi d'Amdahl, il y a d'autres facteurs qui font que l'accélération peut être inférieure au nombre de processeurs. On identifie par *sources de dégradation des performances* (overhead) ces facteurs responsables de la diminution des performances d'un système parallèle. Il y a trois sources principales de dégradation: les communications et la synchronisation entre les tâches, le non-balancement des charges de traitement entre les différents processeurs et les instructions supplémentaires qu'il faut ajouter à un programme pour exprimer le parallélisme.

La synchronisation et l'échange d'information entre les processus s'effectuent à partir du transfert de données sur un réseau de communication physique. La

vitesse de transfert entre deux processeurs sur ce type de réseau peut être de beaucoup inférieure à la vitesse de traitement d'un processeur ou encore à la vitesse de transfert entre un processeur et sa mémoire locale. On appelle *latence de communication* les délais causés par les communications inter-processeurs. Selon le réseau d'interconnexion utilisé, ces délais sont fonction de la vitesse de transfert du signal, de l'efficacité des algorithmes de routage, de la distance à parcourir et du pattern de communication de l'algorithme (qui peut être responsable des contentions au niveau des commutateurs dans le réseau ou d'une sur-demande pour la capacité des liens physiques). Ces latences peuvent parfois être cachées si l'architecture et l'application permettent une exécution en mode multi-tâches (multithreading), c'est-à-dire l'exécution de plusieurs tâches concurremment sur un même processeur. Si une tâche i est bloquée dans l'attente de données, le processeur peut cacher cette latence en exécutant une autre tâche j jusqu'à ce que la tâche i soit de nouveau prête à s'exécuter. Lorsque ces latences ne peuvent pas être cachées, elles créent une augmentation du temps de traitement d'un système parallèle.

Pour les algorithmes synchrones, la synchronisation entre deux tâches pour effectuer un transfert de données ou pour attendre avant d'entrer dans une nouvelle phase de traitement parallèle, peut occasionner une augmentation appréciable du temps de traitement d'un système parallèle. De leur côté, les algorithmes chaotiques peuvent aussi subir certaines dégradations des performances dues aux conflits pour l'accès à une même variable partagée contrôlée par une section critique.

La décomposition (fonctionnelles ou du domaine) en p tâches parallèles en apparence de même taille d'un programme séquentiel peut résulter en des processus où le nombre d'instructions à exécuter varie grandement. C'est le cas par exemple de la décomposition d'une matrice creuse en sous-matrices de même taille mais où le nombre d'éléments non nul peut varier d'une sous-matrice à l'autre. De même pour la décomposition de l'arbre de recherche en sous-arbres de même taille pour certaines procédures de fouille, décomposition qui peut résulter en des tâches d'exploration

différentes d'un sous-arbre à l'autre. Pour ces deux exemples, certains processeurs seront en attente (et donc inutilisés) pendant que d'autres processeurs travaillent sur le problème. Ces temps d'attente correspondent au non-balancement des charges entre processeurs, ils contribuent au temps total du traitement parallèle ce qui fait qu'on les inclut comme source de dégradation des performances.

Les opérations qui doivent être ajoutées pour les différentes procédures de synchronisation, pour le passage de messages, la création de sections critiques, etc. font que la somme totale d'instructions exécutées par un programme parallèle est généralement supérieure à sa contrepartie séquentielle. Il existe parfois des situations où un programme séquentiel réutilise certains résultats pour éviter de refaire un calcul. En parallèle, on refait le calcul si les résultats ne se trouvant pas sur le même processeur provoquent un transfert de données sur le réseau qui serait supérieur au coût d'un calcul redondant. Dans ce cas le programme parallèle peut recalculer plusieurs fois les mêmes résultats ce qui devient une autre source de dégradation des performances due à la parallélisation. Autre facteur qui fait que le nombre d'instructions exécutées en parallèle peut être supérieur à celui du traitement séquentiel vient de ce que le meilleur algorithme séquentiel ne se parallélise pas toujours bien, on utilise parfois un algorithme moins bon en séquentiel pour construire l'algorithme parallèle.

Enfin d'autres facteurs peuvent encore devenir des sources de dégradation des performances. Une mauvaise adéquation entre le modèle de calcul pour lequel l'algorithme parallèle a été conçu et l'architecture sur lequel l'algorithme est effectivement implanté. Une mauvaise affectation des processus sur les processeurs qui augmente la distance de communication entre les processeurs. Une mauvaise décomposition de l'algorithme peut aussi rendre un algorithme parallèle moins performant. D'autres sources non négligeables de dégradation des performances dans l'exécution d'un processus sont les conflits pour l'accès à la mémoire, les interruptions système, les fautes de page, les fautes de cache et la charge de travail du

système d'exploitation.

2.9 Conclusion

Nous avons introduit un certain nombre de concepts portant principalement sur l'architecture, les modèles de programmation, l'algorithmie et les mesures de performances associés au traitement parallèle. Tous ces développements sont largement tributaires de la théorie classique du calcul que nous avons introduit à la section 2.2.1 et de l'expérience acquise par la parallélisation de méthodes de résolution pour des problèmes de calcul numérique. Les défis du traitement parallèle apparaissent alors comme étant de concevoir des architectures d'ordinateurs capables d'exécuter efficacement plusieurs instructions en parallèle et de paralléliser le code ou les algorithmes existant pour qu'ils puissent être exécutés par ces nouvelles architectures.

Il y a eu, bien sûr, beaucoup de travaux de recherche portant sur la parallélisation de méthodes de résolution en dehors de celles appliquées au calcul numérique, mais ce travail s'est fait dans le même esprit que celui par exemple des stratégies de parallélisation appliquées à la décomposition de matrices pour obtenir des tâches parallèles. On a fait peu d'efforts pour comprendre les caractéristiques de méthodes de résolution faisant appel à des heuristiques du point du traitement en parallèle et pour essayer de voir quelles sont les opportunités pour le traitement parallèle de ces méthodes de résolution. Or, les méthodes heuristiques font partie du quotidien de la résolution de problèmes d'optimisation combinatoire. L'objet principal de cette thèse et des chapitres qui suivent se situe donc en dehors de l'univers classique du traitement parallèle pour explorer la parallélisation de méthodes de résolution de problèmes en optimisation combinatoire dont le comportement diffère des méthodes du calcul numérique.

Chapitre 3

Parallélisation des méthodes de recherche

3.1 Introduction

Selon la définition 2.1, les algorithmes constituent une classe particulière de méthodes de résolution dont les résultats sont obtenus après un nombre fini, minimal et essentiel d'étapes. Toute méthode de résolution n'est pas un algorithme au sens de cette définition, c'est le cas par exemple des preuves non-constructives en mathématique qui demandent un nombre infini d'étapes, et des méthodes heuristiques dont les résultats sont obtenus après un nombre non minimal d'étapes.

Au sens de la théorie du calcul (section 2.2.1), les algorithmes correspondent à la classe de programmes pouvant être exécutés par une machine déterministe. Cette définition de la notion d'algorithme ne recoupe pas le même ensemble de méthodes de résolution que la définition 2.1 puisque par exemple, les heuristiques peuvent être traduites en un programme pouvant être exécuté par une machine déterministe.

Le présent chapitre porte sur les concepts et stratégies de parallélisation appliqués aux méthodes de résolution de problèmes d'optimisation combinatoire, en particulier les méthodes de recherche locale. Ces méthodes de résolution sont des algorithmes au sens de la théorie du calcul mais ce n'est pas le cas selon la définition 2.1. Cette observation n'est pas nouvelle, mais avons-nous réellement bien saisi toutes les implications de cette observation pour la parallélisation de ces méthodes?

Nous montrerons que pour plusieurs méthodes de résolution appliquées aux problèmes d'optimisation combinatoire, les sources de parallélisme, le comportement de la procédure parallèle, la relation existant entre les tâches parallèles et les

mesures de performance ne sont pas nécessairement les mêmes que pour les méthodes de résolution considérées comme des algorithmes au sens de la définition 2.1. En particulier nous introduirons la notion de traitement parallèle spéculatif et montrerons que ce type de parallélisme est le plus important pour les méthodes de résolution appliquées aux problèmes d'optimisation combinatoire.

Ce chapitre est organisé de la façon suivante. La prochaine section décrit les principales méthodes de résolutions appliquées aux problèmes d'optimisation combinatoire. Ensuite nous montrerons comment certaines stratégies de parallélisation applicables à ces méthodes de résolution se distinguent de celles que nous avons rencontrées au chapitre 2. Enfin, nous décrirons les différentes stratégies de parallélisation en fonction des méthodes de résolution que nous aurons identifiées.

3.2 Méthodes de résolution des problèmes combinatoires

Au chapitre précédent, nous avons représenté les procédures de résolution en terme d'un graphe orienté où les noeuds correspondent aux opérations et les arcs au flot de données entre les opérations. On peut aussi représenter une procédure de résolution en terme d'un parcours dans un espace de configurations du problème à résoudre. Un espace de configurations correspond alors à tous les états possibles d'un sous-ensemble pertinent des données du problème à résoudre. La méthode de résolution est identifiée au parcours de l'espace de configurations, c'est-à-dire à la génération d'un graphe de configurations où les noeuds représentent les configurations visitées et les arcs l'ordre dans lequel ces visites s'effectuent. Une règle de transition de la méthode de résolution est définie par l'ensemble des opérations exécutées provoquant un changement dans la configuration des données du problème à résoudre.

Exemple 3.1 Supposons que l'on veuille utiliser cette approche pour représenter le déroulement d'un processus de tri sur le vecteur [3,2,1]. L'espace de configurations

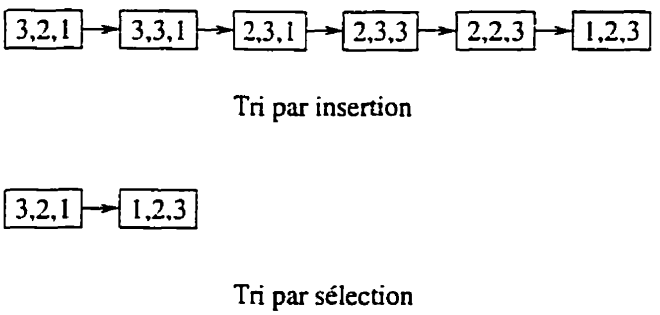


Figure 3.1 Graphe de configurations pour le tri par insertion et par sélection

correspond à tous les vecteurs différents comprenant les trois entiers du vecteur à trier. La configuration initiale du graphe de configurations correspond au vecteur à trier et la configuration finale à celle du vecteur trié. Le processus de tri lui-même est représenté par le graphe de configurations généré entre la configuration initiale et la configuration finale par suite de l'application des opérations du programme de tri sur les données à trier. Considérons deux processus différents de tri: le tri par sélection et le tri par insertion. La règle de transition du tri par sélection consiste à choisir le plus petit entier dans le vecteur dont l'indice est supérieur à 1 et à faire l'échange de cet entier avec celui situé à l'indice 1 du vecteur. Puis l'algorithme répète cette même opération à partir de l'indice 2, etc. La règle de transition du tri par insertion qui consiste à déplacer tous les entiers vers la droite du vecteur pour "insérer" un entier à sa bonne position dans le vecteur trié. Le graphe de configurations généré lors de chaque tri caractérise la méthode de résolution employée, chaque configuration étant absolument nécessaire pour en arriver à la configuration représentant le vecteur trié. La figure 3.1 montre les graphes de configurations pour les tris par insertion et par sélection appliqués au vecteur d'entiers [3,2,1].

Ce type de représentation n'est pas habituel pour des méthodes de résolution comme le tri. Par contre il existe plusieurs classes de problèmes en intelligence artificielle et en optimisation combinatoire pour lesquels la seule méthode de résolution

que l'on connaisse consiste à exécuter une fouille dans l'espace de configurations. Dans ce cas cette forme de représentation est fort utile.

La représentation utilisée dans l'exemple 3.1 va nous aider à préciser la notion d'un nombre minimal et essentiel d'étapes de la définition 2.1. Nous définirons en effet cette notion en fonction du nombre de noeuds générés par le graphe de configurations d'une méthode de résolution.

Définition 3.1 *Soit $\mathcal{G}$ le graphe de configurations généré par une méthode de résolution R suite au parcours de l'espace de configurations entre C_0 et C_f, respectivement les configurations initiale et finale de la donnée d'un problème. Une méthode de résolution utilise un nombre minimal et essentiel d'étapes pour la résolution d'un problème s'il n'existe aucun autre graphe $\mathcal{G}'$ dont le nombre de noeuds est inférieur à celui du graphe $\mathcal{G}$ à partir du même espace de configurations, de la même configuration initiale C_0 et avec la même règle de transition de R.*

Pour les deux procédures de tri de l'exemple 3.1, étant donné la configuration initiale [3,2,1], l'espace de configurations considéré, et la règle de transition de chaque méthode, il n'existe aucun autre graphe $\mathcal{G}'$ ayant un nombre inférieur de noeud à celui apparaissant à la figure 3.1. Dans ce cas, ces méthodes de résolution utilisent un nombre minimal et essentiel d'étapes, et peuvent être considérées comme des algorithmes au sens de la définition 2.1.

Proposition 3.1 *Soit P un problème pour lequel la seule méthode de résolution que l'on connaisse consiste à exécuter une fouille dans l'espace de configurations de ce problème. Une méthode de résolution consistant à exécuter une fouille dans l'espace de configurations de P n'est pas un algorithme au sens de la définition 2.1.*

Preuve: Quel que soit l'espace de configurations et la règle de transition qu'une méthode de fouille se donne, il existe un graphe de configuration $\mathcal{G}$ dont le nombre de noeuds est minimal en terme du nombre d'applications de la règle de transition à la configuration initiale. Supposons qu'une méthode de fouille puisse construire

systématiquement ce graphe minimal pour le problème P. Selon la définition 3.1, cette méthode de fouille serait un algorithme au sens de la définition 2.1 pour le problème P. Ceci contredit l'hypothèse selon laquelle la méthode de fouille est utilisée parce qu'il n'existe pas une telle méthode algorithme.

Tout le présent chapitre porte sur des méthodes de résolution dont la représentation correspond à celle donnée dans l'exemple 3.1. Ces méthodes de résolution appartiennent à une classe différente de celles de la définition 2.1.

3.2.1 Concepts et techniques d'exploration de l'espace de configurations

Pour les problèmes d'optimisation combinatoire, l'espace de configurations peut être limité à l'ensemble de solutions réalisables du problème d'optimisation, c'est-à-dire l'*espace de solutions*, ou encore consister de toutes les combinaisons possibles de l'état des variables de décisions du problème combinatoire, incluant donc des solutions non réalisables. L'espace de configurations est structuré par la méthode de fouille qui définit un graphe orienté sur l'espace de configurations. Les noeuds du graphe orienté correspondent aux configurations et il y a un arc $(x \rightarrow y)$ dans ce graphe si la configuration y peut être obtenue par une seule application de la règle de transition de la méthode de fouille à la configuration x. On dit alors que y est dans le voisinage de la configuration x. Un noeud y est *généré* dans le graphe de configurations si la configuration y est visitée à partir du noeud x par une seule application de la règle de transition de la méthode de fouille. L'*évaluation* d'un noeud y s'applique aux variables de décision représentée par le noeud y, il peut s'agir d'une simple approximation de la valeur de la configuration ou encore correspondre au calcul de la valeur de la fonction objectif pour le noeud y. Le *déploiement* d'un noeud s'obtient à partir de la génération et de l'évaluation de tous les voisins du noeud. En plus de la règle de transition, une méthode de fouille détermine l'ordre des configurations générées.

Méthodes de recherche systématique

Pour le reste de cette section nous supposerons que l'espace de configurations est limité à l'espace de solutions du problème d'optimisation combinatoire.

Lorsque l'espace de solutions n'est pas trop grand, on peut utiliser des techniques de *recherche systématique* qui explorent de manière exhaustive l'espace de solutions. La fouille en largeur et la fouille en profondeur entrent dans cette catégorie de techniques. Cependant les recherches exhaustives ne sont pas des outils très efficaces pour faire face à l'explosion combinatoire qui caractérise certains problèmes d'optimisation combinatoire. De fait, leur application se limite souvent au traitement de problèmes de petite taille.

Une façon de réduire la complexité des techniques de recherche systématiques consiste à les combiner avec des heuristiques afin de les rendre plus sélectives au niveau du parcours de l'espace de solutions. Une *heuristique* est constituée par des règles pour décider du meilleur choix d'alternative parmi plusieurs. Ces règles peuvent être obtenues par différentes relaxations du modèle original du problème, par des connaissances spécifiques au domaine du problème, en utilisant des modèles probabilistes ou encore à travers des connaissances acquises par l'exploration du domaine particulier d'un exemplaire à travers le processus de recherche [77]. Une technique de recherche peut être combinée avec une heuristique sans faire de compromis sur le caractère systématique de la méthode de recherche et donc sur sa capacité de trouver la solution optimale.

La recherche du meilleur d'abord (best-first search) est un exemple de recherche systématique qui fait appel à une heuristique. La stratégie de la recherche du meilleur d'abord consiste à ordonner les noeuds déjà visités du graphe selon une heuristique qui estime les possibilités de chaque noeud, de conduire la recherche vers la solution optimale. À chaque cycle de la recherche, le noeud ayant reçu la meilleure évaluation est déployé. Puisque les noeuds ne sont déployés qu'une seule fois, la recherche du meilleur d'abord explore de façon systématique l'espace de

configurations. Cependant cette stratégie demande beaucoup d'espace mémoire pour la structure de données qui conserve les noeuds déjà visités.

La recherche du meilleur d'abord n'est pas un bon choix de stratégie si la résolution du problème exige une exploration exhaustive de l'espace de solution. On utilise plutôt des heuristiques qui permettent une exploration systématique du graphe de configurations sans avoir à visiter tous les noeuds du graphe. C'est le cas notamment pour la technique d'énumération implicite par évaluation et séparation. Cette méthode de résolution explore de manière implicite le graphe de configurations en calculant à chaque noeud généré x une borne inférieure sur la valeur possible de toute solution se trouvant dans le sous-arbre ayant pour racine le noeud x. Si la borne calculée montre que toutes ces solutions sont nécessairement plus mauvaises que la meilleure solution connue alors (borne supérieure), aucun autre noeud du sous-arbre ne sera généré. L'application par exemple de l'algorithme d'évaluation et séparation au problème du voyageur de commerce génère les chemins un à la fois en gardant en mémoire le plus court jamais trouvé (borne supérieure). Si l'algorithme détermine que la meilleure extension possible d'un chemin aura un coût plus grand que la borne supérieure, ce chemin partiel est éliminé de même que toutes ses extensions possibles. Ceci réduit le nombre de noeuds à générer tout en préservant le caractère systématique de la recherche et sa capacité à trouver la solution optimale.

Selon la définition 2.1, la recherche du meilleur d'abord et l'énumération implicite par évaluation et séparation ne sont pas des algorithmes parce que les règles qui dirigent le branchement de ces méthodes sont basées sur une connaissance partielle du domaine du problème. La résolution d'un problème s'effectue en explorant inutilement plusieurs sous-arbres de l'arbre de configurations, ce qui veut dire que le résultat n'est pas obtenu après un nombre minimal et nécessaire d'étapes comme le veut la définition au sens strict d'un algorithme. Par contre, certains auteurs considèrent que ces méthodes sont des algorithmes puisqu'elles sont assurées de trouver la solution optimale (propriété de "completeness").

Méthodes de recherche approchées

Au-delà d'une certaine taille de problèmes, l'exploration systématique de l'espace de configurations n'est pas envisageable en pratique même avec une heuristique permettant d'élaguer certains sous-arbres. Dans ce cas, on a recours à des stratégies d'exploration dites *irrévocables* [77] ou encore à des méthodes de *génération aléatoire* du parcours de l'espace de solutions. Les *méthodes de descente* (ou recherche locale) appartiennent à la catégorie des méthodes dites irrévocables. La stratégie d'exploration de l'espace de configurations des méthodes de descente consiste à générer un noeud du graphe, à évaluer chacun de ses fils et à déployer le noeud fils dont l'évaluation est la meilleure. Cette méthode ne conserve pas en mémoire les noeuds déjà visités, ce qui empêche le retour à un noeud antérieur du graphe de configurations, d'où le qualificatif de méthode irrévocable.

Les méthodes de descente sont souvent utilisées comme un premier jalon au niveau de techniques d'exploration plus sophistiquées faisant appel à une hiérarchie de règles heuristiques. C'est le cas pour des méthodes comme la recherche tabou et le recuit simulé où s'ajoute à la règle consistant à choisir la meilleure solution dans un voisinage, d'autres règles basées sur l'exploration du domaine (tabou) ou sur les probabilités (recuit simulé). Ces règles visent à empêcher que l'exploration locale des méthodes de descente ne reste bloquée dans des optima locaux. Enfin il existe des méthodes de recherche basées uniquement sur la génération aléatoire de configurations. Le cas le plus typique de cette catégorie est celui des méthodes dites de "generate and test". Les algorithmes génétiques dérivent de cette approche en utilisant la valeur de la fonction objectif comme mécanisme pour créer un biais au niveau de la génération aléatoire de solutions de meilleure qualité. Contrairement aux approches systématiques, il n'existe pas de garanties que ces méthodes basées sur une exploration locale ou encore aléatoire de l'espace de solutions vont trouver une solution optimale.

3.3 Stratégies de parallélisation

Nous avons vu au chapitre précédent qu'il existe deux sources principales et complémentaires de parallélisme, le parallélisme de données et le parallélisme de contrôle. On peut considérer une autre classification des sources du parallélisme selon que la méthode de résolution correspond ou non à un algorithme au sens de la définition 2.1: *parallélisme obligatoire* (mandatory work) et *parallélisme spéculatif* (speculative parallelism).

3.3.1 Parallélisme obligatoire

Le parallélisme obligatoire se base sur l'exploitation du parallélisme logique d'un algorithme déterministe, c'est-à-dire l'exploitation de l'ordonnancement partiel qui résulte de la prise en compte des dépendances de données entre les opérations. Dans le cas où le programme parallèle dérive d'un programme séquentiel, le parallélisme obligatoire provient de la relaxation des contraintes de précédence liées à l'architecture de von Neumann pour produire des opportunités d'exécution concurrente au niveau du programme parallèle. Un programme concurrent écrit dans le style du parallélisme obligatoire exécute le même ensemble d'opérations que le programme séquentiel. La seule différence provient de l'ajout d'instructions pour contrôler le parallélisme et l'affectation (scheduling) des opérations aux processeurs. Le parallélisme obligatoire peut s'appuyer soit sur une décomposition fonctionnelle ou une décomposition du domaine.

3.3.2 Parallélisme spéculatif

Dans la littérature [19], on identifie de *traitement spéculatif* une étape de calcul exécutée par une méthode de résolution sans savoir si cette étape fait partie du nombre minimal et essentiel d'étapes à la résolution d'un problème. On peut déduire de la proposition 3.1 que les méthodes de résolution appliquées aux

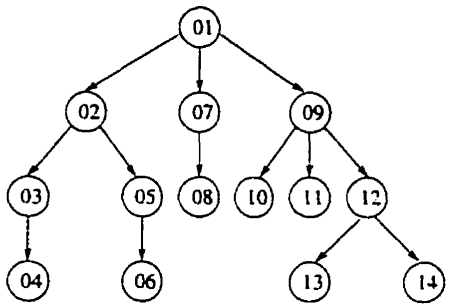


Figure 3.2 Traitement obligatoire versus traitement spéculatif

problèmes d'optimisation combinatoire sont basées sur l'exécution d'étapes de traitement spéculatif, c'est-à-dire des étapes qui sont exécutées sans savoir si elles sont nécessaires pour trouver une solution optimale. Tous les noeuds du graphe de configurations n'appartenant pas au graphe minimal $\mathcal{G}$ peuvent être considérés comme des étapes de traitement spéculatif. L'exemple suivant illustre la différence entre une méthode de résolution de type algorithmique où chaque étape est strictement nécessaire à la résolution du problème et une méthode basée sur du traitement spéculatif.

Exemple 3.2 Supposons que dans l'arbre de la figure 3.2, chacun des noeuds représente une clef d'une valeur différente. Dans un premier temps, nous devons trouver une méthode de résolution qui nous permette de faire la somme de la valeur des clefs se trouvant dans l'arbre. Supposons que la méthode choisie consiste à faire un parcours en profondeur de l'arbre en additionnant la valeur de chaque noeud. Dans ce cas, cette méthode de résolution correspond à la définition d'un algorithme puisque chaque étape de ce parcours en profondeur est strictement nécessaire pour obtenir la solution au problème. Supposons que dans un deuxième temps le problème consiste à extraire de l'information contenue dans le noeud ayant la clef numéro 11. Supposons que la même méthode de résolution basée sur le parcours en profondeur de l'arbre soit appliquée à ce nouveau problème. Cette méthode de résolution va

effectivement résoudre le problème, mais en exécutant plusieurs étapes à caractère spéculatif, puisque la visite de tous les noeuds des deux premiers sous-arbres n'est pas strictement nécessaire pour arriver à la clef 11. Pour trouver une clef particulière, l'utilisation de l'information concernant l'ordonnancement des clefs suffirait pour définir une méthode de résolution qui soit un algorithme, c'est-à-dire qui exécute un nombre minimal et strictement nécessaire d'étapes pour la résolution de ce deuxième problème.

Le parallélisme spéculatif consiste à exploiter le traitement spéculatif exécuté par les méthodes de fouille comme source de parallélisme. Par exemple, pour une méthode d'énumération implicite par énumération et séparation, chaque noeud dans le voisinage d'un noeud nouvellement déployé correspond au noeud racine d'un sous-arbre du graphe de configuration. Chacun de ces sous-arbres constitue une alternative à caractère spéculatif pour l'exploration du graphe de configurations. Le caractère spéculatif de l'exploration de ces sous-arbres découle du fait que l'on ignore si aucun de ces sous-arbres se trouve sur le chemin d'une solution optimale du problème. Cependant, l'exploration de chacun de ces sous-arbres peut se faire de manière indépendante de celle des autres sous-arbres, ce qui veut dire que l'exploration de chaque sous-arbre peut se faire en parallèle avec celle des autres sous-arbres. Cette décomposition en sous-arbres de l'exploration de l'arbre de configurations constitue donc une source de parallélisme, c'est-à-dire une source de parallélisme basée sur l'exploitation du caractère spéculatif de la méthode de recherche.

Bien que les concepteurs de méthodes parallèles n'y fassent pas souvent référence, cette forme de parallélisme basée sur l'exploitation du traitement spéculatif est, *de facto*, l'une des principales stratégies de parallélisation des méthodes de résolutions employées pour les problèmes d'optimisation combinatoire. Elle est également une source importante de parallélisme en intelligence artificielle où il existe souvent un degré important de non-déterminisme [104]. Notons enfin, que

selon que la source d'exploitation du traitement spéculatif est basée sur une décomposition fonctionnelle ou du domaine, on fera référence à une *décomposition fonctionnelle du traitement spéculatif* ou à une *décomposition du domaine du traitement spéculatif.*

3.3.3 Performances du traitement parallèle spéculatif

Les mesures de performances des algorithmes parallèles que nous avons introduites à la section 2.8, peuvent s'appliquer sans problème aux stratégies de parallélisation basées sur l'exploitation du parallélisme obligatoire. L'utilisation directe de ces mesures aux stratégies de parallélisation basées sur l'utilisation du traitement spéculatif d'une méthode de résolution peut poser un certain nombre de problèmes. Nous allons maintenant examiner deux cas précis de parallélisation de méthodes de recherche où les mesures de performance de la section 2.8 ne sont pas applicables directement.

Définition 3.2 *L'* **information heuristique** *correspond à l'information fournie par une règle heuristique d'une méthode de recherche en vue de la génération d'un noeud du graphe de configurations*

Décomposition du traitement spéculatif et information heuristique

Supposons qu'une implantation séquentielle de la technique d'énumération implicite par évaluation et séparation pour un problème de minimisation soit basée sur une fouille en profondeur. L'information heuristique de cette méthode de recherche est constituée par la borne supérieure qui est mise à jour à chaque itération et constitue donc un input de l'itération suivante. Soit $n_{10}, n_{11}, \ldots, n_{1p-1}$ les noeuds dans le voisinage du noeud racine n_{00} du graphe de configurations. Chacun de ces p noeuds est le noeud racine respectivement des sous-arbres $a_0, a_1, \ldots, a_{p-1}$ du graphe de configurations. Ces sous-arbres constituent une alternative à caractère spéculatif puisque le parcours optimal se trouve dans au moins un de ces sous-arbres, ce qui

fait que l'exploration des autres sous-arbres va engendrer l'exécution d'opérations inutiles pour l'obtention de la solution optimale. Ces p sous-arbres peuvent faire l'objet d'une méthode de décomposition se basant sur l'exploitation du caractère spéculatif de la méthode pour découper des tâches parallèles. Supposons que c'est le cas et que chacun de ces sous-arbres est exploré en parallèle à partir du noeud racine n_{1i}, pour $i = 0, \ldots, p-1$ en utilisant la borne supérieure calculée pour le noeud n_{00}.

Proposition 3.2 *L'information heuristique (la borne supérieure) des noeuds* $n_{11}, n_{12}, \ldots n_{1p-1}$ *n'est pas la même pour la procédure séquentielle et la procédure parallèle.*

Effectivement, la borne supérieure du noeud n_{11} de la procédure séquentielle proviendra de l'exploration du sous-arbre ayant pour racine le noeud n_{10}, la borne supérieure du noeud n_{12} proviendra de l'exploration des sous-arbres ayant pour racine n_1 et n_{11}, etc. Par construction de la procédure parallèle, la borne supérieure des noeuds $n_{11}, n_{12}, \ldots n_{1p-1}$ provient de l'évaluation du noeud n_{00}.

La valeur de la borne supérieure n'étant pas la même, l'exploration implicite de chaque sous-arbre risque d'être différente, les noeuds et le nombre de noeuds générés par la procédure parallèle et par la procédure séquentielle ne seront pas les mêmes. Certaines exécutions parallèles feront plus de travail et d'autres moins que le traitement séquentiel. Si lors de la décomposition, les processeurs reçoivent beaucoup de travail à caractère spéculatif (c'est-à-dire le parcours de sous-arbres qui ne conduisent pas à la solution optimale) il sera alors possible qu'une partie de ce travail spéculatif n'ait pas été effectué par le traitement séquentiel, donnant l'impression que l'algorithme parallèle est peu performant. Les accélérations super-linéaires sont dues à une décomposition qui minimise le traitement spéculatif par rapport à l'algorithme séquentiel.

On voit donc que le travail W exécuté par la parallélisation et par conséquent la mesure d'accélération deviennent fonction du type de décomposition et du nombre

de sous-tâches puisque ces deux paramètres déterminent l'information heuristique disponible au niveau de chaque noeud généré. Or les mesures de performance de la section 2.8 reposent sur les hypothèses que (1) les opérations exécutées par la procédure parallèle sont les mêmes que celles de la procédure séquentielle et que (2) l'information dont dispose chaque opération d'une tâche parallèle est identique à l'information dont dispose la même opération dans la procédure séquentielle.

Échange d'information heuristique entre processus concurrents

Certaines implantations du traitement parallèle spéculatif transfèrent de l'information heuristique entre les tâches parallèles. Par exemple, dans le cas des techniques d'énumération implicite par évaluation et séparation, on échange cette information pour améliorer la qualité des bornes. Ce type d'implantation pose certaines difficultés à la mesure des performances en ce sens que le transfert d'information entre les tâches parallèles rend celles-ci interdépendantes et crée certaines dépendances par rapport au système physique. En 1966, Bernstein [13] a en effet défini un ensemble de règles nécessaires pour que deux processus puissent s'exécuter en parallèle. Soit I_i l'ensemble des *inputs* du processus P_i constituant l'ensemble des données nécessaire pour l'exécution de ce processus. De la même façon, soit O_i l'ensemble des données d'*output* générées par l'exécution du processus P_i.

Définition 3.3 *Deux processus p_1 et p_2 avec leurs ensembles d'input I_1 et I_2 et leurs ensembles d'output O_1 et O_2 peuvent s'exécuter en parallèle s'ils sont indépendants, c'est-à-dire si:*

$$I_1 \cap O_2 = \emptyset$$

$$I_2 \cap O_1 = \emptyset$$

$$O_1 \cap O_2 = \emptyset$$

Ces trois équations sont les *conditions de Bernstein.* Ces conditions garantissent que l'ordre d'exécution des processus ne va pas affecter le résultat du traitement

et que donc ces processus peuvent s'exécuter en parallèle. Le partage d'information heuristique entre processus concurrents du traitement parallèle spéculatif ne respecte évidemment pas ces conditions. Dans ce cas le comportement de chaque processus p_i dépend des autres tâches de la procédure parallèle de même que de facteurs tels que la congestion du réseau de communications et la charge de travail de chaque processeur. Le comportement de l'exécution parallèle devient alors non déterministe et les mesures de performances de la section 2.8 ne s'appliquent guère dans ce cas.

3.3.4 Stratégies de parallélisation des méthodes de recherche

Nous avons identifié trois grandes classes de stratégies de parallélisation utilisées pour la parallélisation des méthodes de recherche.

Définition 3.4 *Les stratégies parallélisation des méthodes de recherche se distinguent selon qu'elles exploitent le parallélisme logique ou spéculatif et selon que le parallélisme spéculatif est basé sur une décomposition fonctionnelle ou une décomposition du domaine:*

- *Parallélisme logique (obligatoire) à l'intérieur de chaque itération de la méthode de résolution;*
- *Parallélisme basé sur une décomposition du domaine du traitement spéculatif;*
- *Parallélisme basée sur une décomposition fonctionnelle du traitement spéculatif (recherches multiples avec différents degrés de synchronisation et de coopération entre chaque tâche parallèle).*

Dans les sections qui suivent, nous montrerons comment ces stratégies sont appliquées aux différentes méthodes de recherche.

3.4 Parallélisation des méthodes exactes

Ces trois classes de stratégies de parallélisation sont appliquées à la parallélisation des méthodes de recherche dites systématiques telles que la recherche en largeur, en profondeur, la recherche du meilleur d'abord et les méthodes d'énumération implicite. Les implantations de ces stratégies varient selon que l'architecture de l'ordinateur cible est de type MIMD ou SIMD ou selon le degré d'information heuristique employé par la méthode de recherche.

3.4.1 Type d'architecture cible

Les parallélisations visant une architecture de type MIMD exploitent un parallélisme basé sur une décomposition fonctionnelle du traitement spéculatif. Cette décomposition se fait soit de manière statique ou dynamique. La stratégie de *décomposition statique* repose sur une approche de type maître-esclave, un processeur déploie certains noeuds près de la racine et distribue aux autres processeurs des sous-arbres à parcourir. L'approche statique possède l'avantage de minimiser la dégradation des performances due aux communications mais donne des résultats peu satisfaisant au niveau du balancement des charges. On peut améliorer le taux d'utilisation des processeurs par une *décomposition dynamique* où, lorsqu'un processeur a terminé l'exécution de sa tâche, demande à autre processeur de diviser la charge de travail qui lui reste et d'en confier une partie au processeur qui se trouve sans travail. Une stratégie dynamique de décomposition est d'autant plus souhaitable pour une méthode comme l'énumération implicite par évaluation et séparation où la borne évolue constamment, faisant varier l'espace de configurations à explorer.

On peut voir les recherches systématiques sous la perspective du parallélisme de données et conséquemment viser une implantation sur un ordinateur de type SIMD. En effet, l'exploitation d'un arbre de recherche peut être considérée comme étant la répétition des mêmes opérations de déploiement de noeuds et d'évaluation de la fonction objectif s'effectuant sur des configurations différentes. Le traitement

sur ordinateurs SIMD alterne alors entre des phases synchrones de déploiement des noeuds et de balancement de charges où le travail d'exploration est redistribué aux processeurs. Pour un survol des stratégies de parallélisation appliquées aux méthodes de recherche systématiques, on peut consulter les articles de Gendron et Kumar [41, 49].

3.4.2 Méthodes de recherche sans information heuristique

Les méthodes de résolution qui ne font appel à aucune information heuristique, telles que la fouille en profondeur et la fouille en largeur, doivent générer chaque noeud de l'arbre de recherche pour prouver l'optimalité. Le caractère spéculatif de ces méthodes de résolution ne pose aucun problème au niveau de la parallélisation puisque chaque noeud doit être visité. Le parallélisme est obtenu par une décomposition du domaine du traitement spéculatif. Le travail effectué par la procédure parallèle est identique à celui de la procédure séquentielle. Les mesures de performance de la section 2.8 s'appliquent directement, la principale source de dégradation des performances rendant l'accélération inférieure à p (le nombre de processeurs) provient du non-balancement des charges dans le cas d'une décomposition statique.

3.4.3 Parallélisation des méthodes avec information heuristique

Ces méthodes se caractérisent par le fait que l'ensemble de noeuds générés dans l'arbre d'exploration est fonction de l'information heuristique disponible à chaque itération de la méthode. C'est le cas par exemple pour la technique d'énumération implicite par évaluation et séparation. Gendron & Crainic [41] formulent de la façon suivante l'application des trois classes de stratégies de parallélisation aux techniques d'énumération implicite par évaluation et séparation:

1. Parallélisation des opérations à l'intérieur de chaque itération de la méthode;
2. Parallélisation de l'exploration des sous-arbres de l'arbre de recherche;

3. Parallélisation en exécutant plusieurs recherches différentes.

Le premier type de parallélisation n'exploite pas le traitement spéculatif et respecte les conditions de Bernstein. Les mesures de performances s'appliquent sans difficultés pour comparer les approches parallèles entre elles et avec l'implantation séquentielle.

Le type 2 de parallélisation appliqué à la technique d'énumération implicite par évaluation et séparation correspond à une décomposition du domaine du traitement spéculatif de la technique. Cette décomposition engendre potentiellement les problèmes que nous avons décris à la section 3.3 rendant très hasardeuse l'application directe des mesures de performance décrites au chapitre 2 pour ces parallélisations.

En ce qui concerne le troisième type d'approche, les problèmes sont similaires à ceux du deuxième type.

3.5 Parallélisation des méthodes approchées

En principe les méthodes de recherche approchées sont déterministes, c'est-à-dire que les noeuds visités correspondent à un chemin d'exploration (walk) dans le graphe de configurations par opposition à un arbre d'exploration qui résulte des méthodes systématiques. Le caractère spéculatif de ces méthodes ne provient pas des alternatives d'exploration à chaque noeud du graphe de configurations mais provient plutôt du choix de la stratégie globale d'exploration du graphe de configurations. Les méthodes approchées sont en effet basées sur un ensemble de paramètres de recherche qui dictent le choix du noeud à générer à chaque itération de la méthode ou qui restreignent de manière arbitraire l'exploration à une sous-région de l'espace de solutions. Les méthodes approchées ne sont pas assurées de trouver une solution optimale ce qui fait que les stratégies de parallélisation peuvent viser à augmenter à la fois la vitesse de traitement et la qualité de la solution. Ces deux objectifs peuvent être atteints de deux façons différentes:

1. En effectuant exactement le même chemin d'exploration que la version séquentielle de la méthode;

2. En effectuant une exploration qualitativement différente de l'implantation séquentielle soit en changeant le chemin d'exploration ou en exécutant plusieurs chemins d'exploration concurremment.

Les parallélisations du premier groupe sont des parallélisations obligatoires qui répartissent sur plusieurs processeurs les phases intensives de calcul de l'implantation séquentielle. C'est le cas notamment de l'évaluation des voisins pour les heuristiques construites à partir d'une approche de recherche locale, c'est-à-dire les méthodes de descentes, le recuit simulé et la recherche tabou. On retrouve même ce type de parallélisme au niveau des algorithmes génétiques qui doivent évaluer la qualité (fitness) des individus d'une population à chaque nouveau cycle de la méthode.

Les parallélisations visant à effectuer une exploration qualitativement différente de l'implantation séquentielle se fondent sur l'exploitation du caractère spéculatif des méthodes approchées pour obtenir le parallélisme.

Nous allons maintenant examiner comment chacune des trois grandes classes de stratégies de parallélisation est appliquée aux méthodes approchées suivantes: la recherche tabou, le recuit simulé et les algorithmes génétiques.

3.5.1 Exploitation du parallélisme obligatoire

Les méthodes de résolution comme la recherche tabou, du recuit simulé et les algorithmes génétiques sont des méthodes de recherche purement spéculatives. Cependant, les stratégies de parallélisation basées sur le parallélisme obligatoire n'exploitent pas le caractère spéculatif de ces méthodes, elles se limitent à l'exploitation du parallélisme logique des opérations normalement exécutées par la procédure séquentielle au niveau de chaque itération. Ce parallélisme obligatoire se base soit sur une décomposition fonctionnelle ou une décomposition du domaine. L'exploration de l'espace de solutions effectuée par la procédure parallèle est la même que celle de la

procédure séquentielle. La solution obtenue est la même pour les deux procédures si le nombre d'itérations de la méthode est le même et si les opérations réalisées à chaque itération n'ont pas été volontairement changées par le concepteur de la procédure parallèle. Lorsque l'implantation de ce type de parallélisme se fait sur une machine à mémoire distribuée, on utilise habituellement une approche de type maître-esclave, où le maître distribue les tâches aux processeurs esclaves. Il n'y a pas de communication entre les processeurs esclaves, la communication se faisant uniquement entre le maître et les esclaves au début et à la fin de chaque phase d'exécution en parallèle. Toutes les mesures de performance de la section 2.8 s'appliquent au parallélisme obligatoire des méthodes de recherche approchées.

Parallélisme obligatoire appliqué aux algorithmes génétiques

Les *parallélisations globales* (global parallelization) [20, 46] des algorithmes génétiques s'appuient sur une décomposition du domaine des individus d'une population. Ces parallélisations portent sur deux catégories d'opérations au niveau de chaque génération de la méthode: l'évaluation des individus de la population et l'application en parallèle des opérateurs génétiques telles que les opérations de sélection, crossover, mutation et inversion [4]. L'évaluation en parallèle des individus se base sur une décomposition de la population de chaque génération en p sous-populations où p est le nombre de processeurs. En ce qui concerne l'application en parallèle des opérateurs génétiques, la population est décomposée en sous-ensembles à chaque génération de la méthode, chaque sous-ensemble d'individus se voyant appliquer les opérateurs génétiques séquentiellement sur différents processeurs. Les parallélisations de Abramson & Abela [4], Fogarty & Huang [39] et Hauser & Männer [54] appartiennent aux approches de parallélisation globale des algorithmes génétiques.

Parallélisme obligatoire appliqué aux procédures de recuit simulé

Relativement au recuit simulé, le parallélisme obligatoire peut prendre sa source soit à partir de décompositions fonctionnelles ou de décompositions du domaine [1, 81]. Pour le parallélisme à partir d'une décomposition fonctionnelle (single-trial parallelism), la fonction d'évaluation de chaque transition du voisinage de la solution courante est décomposée pour être exécutée en parallèle. Cette stratégie dépend fortement de la fonction d'évaluation. Les implantations de Kravitz & Rutenbar [62, 63] appartiennent à cette catégorie de parallélisation du recuit simulé. Pour le parallélisme basé sur une décomposition du domaine (multiple-trial parallelism), l'évaluation des transitions dans le voisinage de la solution courante est divisée entre les processeurs, c'est-à-dire que ceux-ci évaluent simultanément certaines transitions à partir de la seule solution courante. Lorsqu'un processeur accepte une solution, tous les processeurs sont arrêtés et cette solution devient la nouvelle solution courante de tous les processeurs pour l'itération suivante. Les implantations parallèles du recuit simulé de Roussel-Ragot & Dreyfus [81], Virot [102], Kravitz & Rutenbar [62], Rutenbar & Kravitz [82], Aarts et al. [2] appartiennent à cette catégorie.

Parallélisme obligatoire appliqué à la recherche tabou

Pour la recherche tabou, une seule procédure tabou est exécutée par le processeur maître qui délègue une partie plus ou moins grande du travail à accomplir à chaque itération de la méthode à des processeurs esclaves. Le processeur maître accumule toute l'information heuristique qui résulte de l'exécution de la procédure tabou, distribue les tâches devant être exécutées par les autres processeurs, et détermine quand la recherche doit se terminer. Les tâches qui sont déléguées aux processeurs esclaves consistent par exemple à explorer en parallèle le voisinage [21, 23], à construire et évaluer la liste candidate [45] ou à exécuter un certain nombre d'itérations de la recherche tabou (probing) [29].

3.5.2 Parallélisme par décomposition du domaine du traitement spéculatif

Une décomposition du domaine du traitement spéculatif au niveau des méthodes approchées s'obtient en exécutant la même procédure séquentielle avec des sous-ensembles différents de variables de décisions. Chaque exécution fait l'hypothèse que les variables des autres sous-ensembles ne changent pas d'état durant l'exécution. Soit $d_1, d_2, \ldots, d_n$ un ensemble de n variables de décisions et p le nombre de processeurs. L'ensemble de variables peut être partagé en p sous-ensembles de $\frac{n}{p}$ variables, chaque sous-ensemble i étant affecté au processeur P_i de la manière suivante:

$$\overbrace{d_1, d_2, \ldots, d_{\frac{n}{p}}}^{P_1}, d_{\frac{n}{p}+1}, d_{\frac{n}{p}+2}, \ldots, d_n$$

$$d_1, d_2, \ldots, d_{\frac{n}{p}}, \overbrace{d_{\frac{n}{p}+1}, d_{\frac{n}{p}+2}, \ldots, d_{2\frac{n}{p}}}^{P_2}, d_{2\frac{n}{p}+1}, d_{2\frac{n}{p}+2}, \ldots, d_n$$

$$\vdots$$

$$d_1, d_2, \ldots, d_{n-p}, \overbrace{d_{n-p+1}, d_{n-p+2}, \ldots, d_n}^{P_p}$$

L'exploration de l'espace de solutions de cette procédure parallèle sera tout à fait différente de celle exécutée par la procédure séquentielle, étant donné que la décomposition du vecteur des variables de décisions devient un paramètre qui a un impact direct sur le choix des régions explorées de l'espace de solutions. Chaque décomposition des variables de décisions exclut de l'exploration de larges régions de l'espace de solutions. Il est préférable d'exécuter plusieurs cycles d'exploration avec des affectations différentes des variables aux processeurs (par exemple en faisant un décalage des variables de décisions à chaque cycle). Si la parallélisation est synchrone, les conditions de Bernstein sont respectées et le comportement sera

déterministe. Une base de comparaison acceptable des performances entre la procédure séquentielle et une procédure parallèle basée sur une décomposition du domaine du traitement spéculatif consiste à comparer le temps d'exécution et la qualité des solutions sur la base d'un nombre égal d'itérations pour chaque méthode (c'est-à-dire le nombre d'itérations de la méthode parallèle multiplié par le nombre de processeurs).

Parallélisme spéculatif par décomposition du domaine appliqué aux méthodes de recherche approchées

À notre connaissance, des stratégies de parallélisation par décomposition du domaine du traitement spéculatif n'ont pas été appliquées aux algorithmes génétiques.

Relativement au recuit simulé, chaque processeur exécute un recuit simulé sur le sous-ensemble de variables qui lui sont affectées, visant à trouver un optimum local à la région de l'espace de solutions définie par ce sous-ensemble. Ensuite les solutions de chaque processeur sont combinées pour obtenir une solution pour le problème original. Règle générale, ce cycle de décomposition du vecteur des variables de décision et d'optimisation dans plusieurs régions de l'espace de solutions est répété un certain nombre de fois, à partir d'une affectation différente des variables de décision aux processeurs pour pouvoir considérer le plus grand nombre possible de sous-régions. Des exemples de ce type de parallélisation pour le recuit simulé: Felten, Karlin & Otto [36], Damera-Rogers, Kirkpatrick & Norton [33] et Devadas & Newton [34].

L'application des stratégies de parallélisation par décomposition du domaine du traitement spéculatif pour la recherche tabou est similaire à celle du recuit simulé, c'est-à-dire décomposition des variables de décisions en sous-ensembles et exécution de la même procédure de recherche tabou sur chaque sous-ensemble. Ces approches furent testées pour le problème de tournées de véhicule par Taillard [92] et pour le problème du voyageur de commerce par Fiechter [37] .

3.5.3 Parallélisme par décomposition fonctionnelle du traitement spéculatif

Les stratégies de parallélisation par décomposition fonctionnelle du traitement spéculatif appliquées aux méthodes de recherche approchées se fondent sur l'exécution de plusieurs recherches concurrentes, c'est-à-dire l'exécution de plusieurs chemins d'exploration dans le même espace de solutions. Dans cette thèse nous considérerons que des parallélisations où chaque recherche appartient à la même méthode de recherche. C'est principalement ce que l'on retrouve actuellement dans la littérature, mais on pourrait concevoir des approches où les recherches appartiennent à des méthodes différentes de recherche.

On distingue deux catégories de stratégies de parallélisation basées sur une décomposition fonctionnelle du traitement spéculatif: les recherches multiples, et les recherches multiples coopérantes. Dans le cas des recherches multiples, le caractère spéculatif de ce type de stratégies de parallélisation provient du fait que seules les opérations exécutées par la procédure séquentielle ayant obtenue la meilleure solution constituent du traitement nécessaire à la résolution du problème. Le traitement effectué par les autres recherches ne contribue d'aucune manière à l'obtention de cette meilleure solution. Le caractère spéculatif des procédures parallèles par recherches multiples coopérantes est plus complexe à définir, nous y reviendrons principalement dans le chapitre 7.

Recherches multiples coopérantes appliquées aux algorithmes génétiques

Les stratégies de parallélisation avec recherches multiples coopérantes sont celles que l'on retrouve le plus dans la littérature portant sur la parallélisation les algorithmes génétiques. On subdivise ces stratégies en deux sous-groupes: *parallélisations à grains larges* (coarse-grained parallelization) qui sont des approches synchrones et les *parallélisations à grains fins* (fine-grained parallel genetic algorithm) qui sont des approches asynchrones d'algorithmes génétiques parallèles. Dans le sous-groupe à

grains larges, la population est divisée en un petit ensemble de sous-populations, chaque sous-population étant soumise à l'exécution d'un algorithme génétique sur un processeur différent. Un opérateur de **migration** est ajouté à l'ensemble des opérateurs de la méthode génétique pour permettre l'échange d'information entre les sous-populations. Les enjeux importants pour ce type de méthodes parallèles pour les algorithmes génétiques sont: la **topologie** qui définit les connexions entre les sous-populations (on utilise souvent celle de l'ordinateur parallèle qui exécute les tests), le **taux de migration** qui contrôle combien d'individus émigrent, et l'**intervalle de migration** qui définit quand la migration doit se produire. Les parallélisations de Pettey, Leuze & Grefenstette [78], Tanase [95] et Gordon & Whitley [46] sont quelques unes parmi plusieurs implantations de l'approche à grains larges.

La stratégie à grains fins partage la population en un grand nombre de sous-populations de petite taille, la taille idéale étant d'un individu par processeur. Les opérateurs génétiques s'exécutent toujours à travers une interaction asynchrone entre deux processeurs. Chaque processeur possède un voisinage de processeurs avec lesquels l'individu associé au processeur peut exécuter les opérateurs génétiques, les voisinages sont inter-reliés entre eux ce qui fait que les modifications des individus peuvent se propager dans la population. Manderick & Spiessens [68], Mühlenbein, Gorges-Schleuter & Kramer [73] et Gordon, Whitley & Bohm [47] présentent des implantations de ce type de parallélisation.

Recherches multiples et recherches multiples coopérantes appliquées aux procédures de recuit simulé

Au niveau du recuit simulé, on retrouve des stratégies de parallélisation par recherches multiples, chacune avec une solution initiale différente et la même schédule de température (voir Azencott [8]). On utilise aussi des stratégies de parallélisation

par recherches multiples coopérantes synchrones où toutes les recherches interagissent périodiquement pour échanger leur meilleure solution: avec la même stratégie de recherche ou avec des stratégies de recherche différentes (températures différentes pour chaque processeur (voir Graffigne [48]).

Recherches multiples et recherches multiples coopérantes appliquées à la méthode tabou

Au niveau de la recherche tabou, les stratégies d'exploration des procédures séquentielles peuvent se différencier selon que les solutions initiales de chaque recherche soient identiques ou différentes (SP.., MP..), ou encore que les paramètres de recherche sont uniques ou multiples (..SS, ..DS), la combinaison de ces valeurs donnant les stratégies de différentiation SPDS, MPDS, SPSS, MPSS, la stratégie de différentiation SPSS n'étant jamais implantée puisqu'elle consiste à répéter la même recherche p fois. Les recherches peuvent effectuer l'exploration de l'espace de solutions de manière complètement indépendante (recherches multiples), ou peuvent s'échanger de l'information à intervalle régulier (recherches multiples coopérantes synchrones), ou selon la logique interne de chaque processus (recherches multiples coopérantes asynchrones).

Les mêmes enjeux se posent pour la recherche tabou que pour les parallélisations avec recherches multiples coopérantes pour les algorithmes génétiques, à savoir: la topologie définissant le réseau d'interconnexions entre les procédures séquentielles (à quelle recherche l'information doit être envoyée), l'intervalle d'échange d'information pour les approches synchrones (quand faire l'échange d'information), et l'information à échanger entre les recherches (quelle information échangée). La problématique portant sur l'information à échanger est encore plus complexe pour la méthode tabou à cause de la variété d'information qui peut être échangée (la recherche tabou accumule explicitement beaucoup d'information sur l'espace de solutions, ce qui n'est pas le cas pour le génétique et le recuit) et des stratégies

globales d'accumulation d'information qui peuvent être envisagées (pool ou sans pool). Certaines implantations plus avancées visent à utiliser la diversité de l'information heuristique obtenue à partir de plusieurs trajectoires de recherche différentes pour déduire de nouvelles connaissances sur l'espace de solutions. Ici l'information accumulée n'a plus simplement un rôle passif d'emmagasinage de la connaissance, mais elle devient un processus de création de connaissances en appliquant différentes procédures d'analyse sur l'information heuristique.

Les comparaisons au niveau des performances entre la parallélisation par p recherches indépendantes et la procédure séquentielle, utilisent certaines mesures basées sur des modèles probabilistes pour la recherche tabou et les algorithmes génétiques [10], pour la recherche tabou [94] et pour le recuit simulé [8]. Les résultats qui ont été rapportés pour ce genre de parallélisation semblent indiquer que pour un même travail W, la solution obtenue par p recherches de traitement avec t itérations chacune, est meilleure que celle d'une procédure séquentielle exécutant pt itérations. Ces tests furent réalisés sur des tailles fixes de problème, on ne connaît pas l'effet de l'accroissement de la taille du problème sur cette relation. De plus les résultats qui furent obtenus dans [29] sont pour un ensemble de p stratégies de recherche qui se rapprochent de la meilleure stratégie séquentielle connue, on ignore qu'elles seraient les conséquences sur la relation précédente d'une augmentation de la valeur de p qui entraînerait soit des stratégies trop peu différenciées ou encore des stratégies de recherche générées aléatoirement.

3.6 Conclusion

Les méthodes de résolution appliquées aux problèmes d'optimisation combinatoire se caractérisent par le recours au traitement spéculatif, c'est-à-dire l'exécution d'étapes de calcul avant de savoir si ces étapes sont nécessaires à la résolution directe du problème. Ces étapes de traitement spéculatif sont souvent indépendantes les unes par rapport aux autres au sens des conditions de Bernstein et constituent une source

de parallélisme importante. Il y a donc en plus du parallélisme logique, la possibilité d'utiliser un parallélisme spéculatif pour la parallélisation de ces méthodes, ce qui accroît le choix des stratégies de parallélisation qui peuvent être appliquées. La mesure des performances des procédures parallèles découlant de l'utilisation du parallélisme spéculatif qui est cependant problématique. En effet, l'exploration de l'espace de solutions et les opérations exécutées par la procédure parallèle ne sont pas les mêmes que pour la procédure séquentielle. Il devient ainsi difficile de mesurer l'accélération et l'efficacité attribuable à la procédure parallèle et de comparer les stratégies de parallélisation entre elles (un problème de plus en plus souvent soulevé au niveau des algorithmes génétiques parallèles).

Chapitre 4

Stratégies de parallélisation de la méthode tabou

4.1 Introduction

La recherche tabou est souvent décrite comme étant une heuristique de *haut niveau* pour la résolution de problèmes d'optimisation combinatoire, conçue pour guider d'autres heuristiques afin d'éviter que ces dernières ne s'arrêtent dans des optima locaux. Cette *métaheuristique* se comporte comme une technique de recherche adaptative qui vise à diriger l'exploration de l'espace de solutions vers la découverte de bonnes solutions ou de solutions optimales. Généralement, deux mécanismes sont utilisés pour guider la trajectoire de recherche. Le premier mécanisme vise à prévenir que la recherche ne cycle à partie de listes tabous qui gardent une trace des solutions visitées récemment lors de la recherche. Le second mécanisme fait usage d'une ou plusieurs *mémoires* pour diriger la recherche dans des régions prometteuses où vers des régions non encore explorées de l'espace de solutions.

Il est bien connu que ces mécanismes de mémoires de la méthode tabou peuvent être considérés comme des capacités d'apprentissage qui graduellement construisent des images de bonnes solutions ou de solutions prometteuses. L'existence de ces capacités d'apprentissage et les mécanismes de guidage impliquent, d'un côté, que la connaissance fournie par la recherche tabou sur le problème à être résolu est plus riche que celle générée pendant l'exécution d'une méthode de recherche telle l'énumération implicite par évaluation et séparation pour le même problème. D'un autre côté, ces caractéristiques distinguent clairement la recherche tabou des recherches de type aléatoire en introduisant un but dans le processus d'exploration

de l'espace de solutions. Ainsi, une plus grande variété de procédures de recherche tabou peut être conçu pour une classe particulière de problèmes, et cette caractéristique prend de l'importance lorsque des implantations parallèles sont prises en compte.

Les architectures d'ordinateurs parallèles permettent le développement de procédures qui explorent plus efficacement l'espace de solutions. Comme nous l'avons mentionné au chapitre 2, deux types d'approches sont employées pour accroître cette efficacité, l'affectation de plusieurs processeurs à certaines phases du calcul particulièrement répétitives et lourdes en temps de traitement de l'algorithme séquentiel, la conception de nouvelles procédures visant à utiliser d'autres formes d'exploration de l'espace de solutions. Dans le contexte des techniques d'énumération implicite par évaluation et séparation, Trienekens and Bruin [97] réfèrent à ces approches comme étant respectivement des parallélisations de bas et de haut niveau. Une implantation parallèle de bas niveau d'une méthode de recherche implique qu'il n'y a pas de changement dans l'interaction entre les différentes sections de la méthode. Dans ce cas, l'implantation parallèle n'est pas intrinsèquement différente de sa version séquentielle, seulement plus rapide. Dans le contexte de la recherche tabou, cette distinction peut devenir plus flou. En particulier, on doit considérer comment la stratégie de parallélisation affecte l'information relative à l'historique et à la trajectoire de la recherche globale qui est disponible au niveau de chaque procédures séquentielles. C'est alors que les problèmes concernant l'échange et le traitement de l'information entre les processus, central à la conception de beaucoup de procédures parallèles, prennent une dimension encore plus importante lorsque le tabou parallèle est considéré.

L'objectif du présent chapitre est d'introduire une taxonomie des techniques de parallélisation pour la méthode tabou qui vise à aborder explicitement ces considérations. Nous allons incorporer à notre taxonomie des critères de classification basés non seulement sur les méthodes traditionnelles de décomposition de

l'algorithme séquentiel, mais aussi sur les stratégies de partage de l'information et de contrôle de la recherche utilisées dans la conception d'algorithmes parallèles pour la méthode tabou. Cette taxonomie tente de présenter une image complète des stratégies de parallélisation pour la recherche tabou, et contribue à faire une meilleure analyse et comparaison des stratégies de parallélisation proposées dans la littérature. La taxonomie peut aussi aider à mieux comprendre la relation qui existe entre la nature de la recherche tabou et le traitement parallèle, particulièrement au niveau des mécanismes d'acquisition de la connaissance. Finalement, elle identifie de nouvelles stratégies de parallélisation et suggère des orientations intéressantes de recherche futures.

4.2 Introduction de la recherche tabou

Nous allons rappeler brièvement les principales articulations de la méthode tabou. Une procédure de recherche tabou visant à résoudre le problème d'optimisation

$$Min\{c(x)|x \in X \subset R^n\}$$

peut être vu comme une combinaison de trois phases principales: recherche locale, intensification de la recherche dans une région particulière de l'espace de solutions, re-démarrage de la recherche dans une région de l'espace de solutions non encore explorée.

Pendant l'exploration de l'espace de solution à chacune de ces phases, des données sont accumulées sur le problème et son espace de solutions afin d'obtenir une description qui se rapproche d'une solution de bonne qualité, d'identifier les régions de l'espace de solutions où de bonnes solutions peuvent se trouver et de guider la recherche. La figure 4.1 montre le diagramme de transition de cette méthode entre ces trois phases: exploration locale, intensification et diversification.

La phase de recherche locale s'exécute selon une *règle de transition* qui définit, pour chaque solution x visitée, un voisinage $V(x)$ correspondant à l'ensemble des

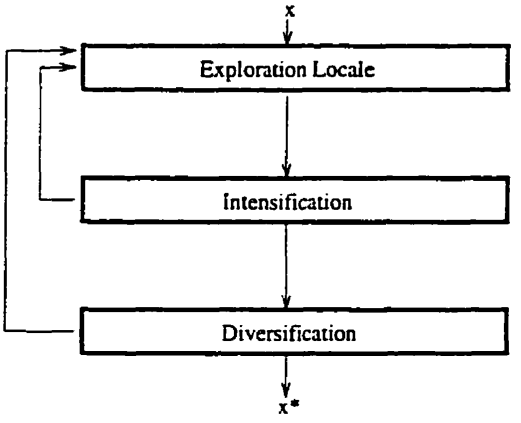


Figure 4.1 Tabou séquentiel

solutions pouvant être générées par une seule application de la règle de transition à la solution x. Lorsque l'évaluation de toutes les solutions dans le voisinage est trop coûteux, la recherche locale peut ne tenir compte que d'un sous-ensemble du voisinage identifié comme étant la *liste candidate*. Une méthode de recherche locale standard se termine lorsque pour la solution courante x, $c(x') > c(x)\ \forall x' \in N(x)$, c'est-à-dire lorsqu'il n'existe plus aucune modification par suite de l'application de la règle de transition r capable de provoquer un changement monotone de la valeur de la solution. Cette dernière solution n'est généralement pas une solution optimale. on dit alors que la procédure de recherche locale est bloquée dans un optimum local. Dans le cas de la recherche locale utilisée par la méthode tabou, il est permis d'avoir $c(x') > c(x)$, ce qui autorise la recherche tabou à se sortir des optima locaux. Pour éviter que la procédure de recherche ne cycle, le tabou garde une trace de l'historique récent de la recherche, cette mémoire à court terme est implantée comme une *liste tabou* qui interdit la sélection de certaines transitions. Le nombre, la taille, le contenu et les politiques de gestion de ces listes tabous constituent des aspects importants du travail de recherche effectué par plusieurs chercheurs sur la méthode tabou.

La phase d'*intensification* correspond à une exploration plus intense d'une partie de l'espace de solutions identifié par la phase de recherche locale comme

pouvant contenir de bonnes solutions. L'identification de ces régions est basée sur des attributs de solutions qui sont gardés dans des mémoires à moyen terme.

La *diversification* est un mécanisme utilisé pour diriger la recherche vers des régions qui semblent encore non explorées. Le choix de ces régions est obtenu à l'aide de mémoires à long terme qui gardent de l'information sur des attributs des meilleurs solutions rencontrées depuis le début de la recherche. La diversification correspond à la sélection d'une solution qui possède des valeurs différentes de ces attributs suivie d'une phase limitée de recherche locale contrainte par des listes tabous visant à empêcher l'exploration locale de retourner vers des régions déjà explorées.

Définition 4.1 *Une* **solution initiale** *est une configuration de valeurs des variables de décision du problème d'optimisation obtenue lors de la phase d'initialisation d'une procédure de recherche tabou à partir d'un choix aléatoire ou par une procédure prédéterminée.*

La solution initiale sert à initialiser la recherche dans une région particulière de l'espace de configurations du problème.

Définition 4.2 *Les* **paramètres de recherche** *de la méthode tabou correspondent à la taille des listes tabous, le critère d'arrêt, le critère d'aspiration, la durée de la phase d'exploration locale, la durée de la phase d'intensification, certaines politiques de gestion des listes tabous et de définition des différentes phases de la méthode.*

Définition 4.3 *L'ensemble des valeurs associées aux paramètres de recherche constituent la* **stratégie d'exploration** *de l'espace de solutions d'une procédure de recherche tabou.*

Définition 4.4 *Nous identifierons par* **historique de la recherche** *d'une procédure tabou, l'ensemble de l'information contenue dans les diverses mémoires d'une procédure de recherche tabou à un instant donné.*

4.3 Classification des approches tabous parallèles

Il existe relativement peu de recherche concernant la conception de procédures parallèles pour la méthode tabou, et nous ne connaissons qu'une seule tentative (Voß[103]) pour classifier les différentes approches de parallélisation qui peuvent s'appliquer dans ce contexte. La classification de Voß est basée sur une analogie avec la taxonomie de Flynn [38] relativement à l'architecture des ordinateurs parallèles. Cette taxonomie classe les approches de parallélisation de la méthode tabou en quatre catégories selon que:

- Les solutions initiales des procédures concurrentes de recherche tabou sont identiques ou différentes;
- Les stratégies d'exploration de l'espace de solutions des procédures concurrentes de recherche tabou sont identiques ou différentes.

Selon nous, cette classification est incomplète puisqu'elle ne tient pas compte des différences existant entre une implantation centralisée ou distribuée du contrôle du traitement parallèle, de l'utilisation partielle ou totale de la connaissance accumulée par les processus concurrents sur l'espace de solutions, du degré de diversité de la connaissance accumulée et des problèmes de même que les opportunités que cette diversité génère, enfin des implications entre une exécution synchrone ou asynchrone du traitement parallèle. La taxonomie que nous présentons à pour objectif de combler ce vide.

4.3.1 Dimensions de la taxonomie

La méthode tabou repose sur l'usage qui est fait de l'historique de la recherche, c'est-à-dire l'information accumulée concernant les régions de l'espace de solutions déjà visitées et les attributs des solutions trouvées. Cette information joue un rôle déterminant entre autre pour guider la recherche à long terme (phase de diversification) et pour identifier les régions qui font l'objet d'une exploration plus intensive

(phase d'intensification) de la recherche. Bien qu'ici nous ne classifierons pas les algorithmes parallèles tabous selon leur mode d'exploration de l'espace de solutions ou encore selon leur mode d'acquisition de la connaissance, il faut néanmoins tenir compte de ces deux composantes importantes lorsque l'on envisage de paralléliser cette méthode. Par exemple, certaines stratégies de parallélisation de haut niveau impliquant une décomposition implicite du domaine d'exploration ont pour effet de répartir l'historique de la recherche parallèle sur plusieurs processeurs. La décision de partager ou pas entre les processus concurrents cette information a un impact évident sur le choix des régions pour les phases d'intensification et de diversification de la recherche tabou. Conséquemment, pour la méthode tabou, les décisions relatives au partage de l'historique de la recherche sont des choix tout aussi important dans une stratégie de parallélisation que la manière de décomposer le domaine d'exploration et l'allocation des tâches aux différents processeurs.

Notre taxonomie est construite selon trois dimensions visant à tenir compte de tous ces facteurs. La première dimension distingue les approches de parallélisation de la recherche tabou selon le "nombre" de stratégies d'exploration différentes utilisées, ce qui correspond en fait au nombre de recherches ou chemins différents exécutés dans l'espace de solutions. La deuxième dimension distingue les approches de parallélisation selon leur mode de partage entre les processus de l'historique de la recherche. Enfin, la troisième dimension distingue les approches de parallélisation selon le "type" de stratégies d'exploration de l'espace de solutions. Ces trois dimensions sont illustrées dans le tableau 4.1.

4.3.2 Cardinalité de la recherche

Le contrôle de l'exploration parallèle peut se situer soit au niveau d'un seul processus, le *processus maître*, ou bien être distribué parmi plusieurs processus sur différents processeurs. Les choix effectués au niveau de cette dimension ont un effet direct sur le degré de diversité de la connaissance accumulée, sur la distribution de celle-ci et

Cardinalité de la recherche	1-chemin p-chemins
Stratégies de partage de l'information	Synchronisation par Tâches Partage Synchrone Collégiale Asynchrone Réflexif Asynchrone
Stratégies d'exploration	SPSS SPDS MPSS MPDS

Tableau 4.1 Dimensions de la Taxonomie

sur le type d'architecture de l'ordinateur parallèle qui convient pour l'exécution de la procédure tabou parallèle.

Le premier cas que nous identifierons par **1-chemin**, correspond de façon triviale au traitement séquentiel, c'est-à-dire l'exécution d'une seule procédure de recherche tabou. Dans le contexte parallèle, le cas 1-chemin représente l'approche où un seul processus, le processus maître, exécute une procédure de recherche tabou, mais délègue une partie de son travail à d'autres processeurs. Le processus maître accumule toute la connaissance qui résulte de l'exécution de la procédure de recherche tabou (il y a donc un seul historique de la recherche de l'espace de solutions) et distribue les tâches devant être exécutées par les autres processeurs. Le processus maître exécute toute les décisions importantes de la recherche tabou: début et fin des phases de recherche locale, intensification et diversification, choix des régions d'exploration et détermine quand la recherche doit se terminer. Les tâches qui sont déléguées aux processeurs esclaves peuvent être du traitement numérique demandant beaucoup de temps de calcul, l'exploration en parallèle du voisinage [21, 23], ou la construction et l'évaluation de la liste candidate. La diversité des connaissances accumulées est faible et le degré de cohérence entre les éléments de celles-ci tend à être élevé puisque ces connaissances résultent du progrès d'un seul chemin

de recherche dans l'espace de solutions. Pour ce type de parallélisation, on minimisera les temps d'attente dus aux communications si le réseau d'interconnexion de l'ordinateur parallèle est configuré en étoile, ce qui correspond à la structure logique des communications entre le processus maître et les processus esclaves.

Dans le deuxième cas, le contrôle de la recherche est complètement distribué à travers p processus différents, $p > 1$, d'où le nom de **p-chemins** pour ce type de stratégies. Chaque processeur exécute une procédure complète de recherche tabou, l'historique de recherche de chaque procédure peut être partagée ou pas avec les autres procédures de recherche distribuées à travers un réseau de plusieurs processeurs. La recherche globale se termine lorsque chaque processus rencontre son critère d'arrêt. La coordination de l'échange des connaissances et les tentatives pour assurer que l'information adéquate soit disponible lorsque requise, sont parmi les principaux enjeux dans ce type de parallélisation. La diversité de la connaissance accumulée peut être très grande puisque provenant de différents chemins de recherche, la réconciliation de ces connaissances diverses pouvant donner naissance à de nouvelles informations sur l'espace de solutions. Ici, comme nous le verrons, la topologie du réseau d'interconnexion peut varier grandement en fonction des stratégies de partage de l'information.

4.3.3 Stratégies de partage de l'information

La deuxième dimension de la taxonomie porte sur la façon dont la connaissance est partagée et traitée entre les processus. Cette dimension distingue entre les stratégies de parallélisation synchrones et asynchrones qui influencent pour certaines stratégies de parallélisation le degré de connaissances disponibles au niveau de chaque processus à différentes étapes du traitement parallèle. Cette dimension reliée à l'échange d'information entre les processeurs se décompose en quatre catégories qui se regroupent selon les deux niveaux de cardinalité de la recherche pour définir les stratégies de parallélisation relatives à l'accumulation, au partage et au traitement

de l'information échangée entre les processeurs.

Selon la définition 2.5 du traitement synchrone, l'exécution synchrone d'une procédure de résolution en parallèle repose sur le fait que chaque processus est décomposé en étapes à la fin desquelles les processus interagissent entre eux pour se synchroniser ou échanger de l'information. Au niveau de la méthode tabou, les points d'interaction peuvent être définis en fonction du nombre d'itérations, d'un intervalle de temps ou être relatifs aux phases de la procédure de résolution. De plus, les points d'interaction peuvent être, soit codés au niveau de chaque procédure ou exécutés par un processus externe à l'aide d'interruptions des processus concurrents.

Nous qualifierons de **synchronisation par tâches** la première catégorie de cette dimension de la taxonomie. Rappelons qu'une tâche résulte de la décomposition (fonctionnelle ou du domaine) d'une procédure de résolution pour son traitement en parallèle. Une tâche est la partie de la procédure de résolution exécutée par un seul processeur. La synchronisation par tâches inclut les stratégies de parallélisation par recherches multiples où les tâches ne comprennent qu'une seule étape avec un point d'interaction au début et à la fin de la tâche. Les stratégies parallèles de synchronisation par tâches se caractérisent également par le fait que soit qu'il y a un seul historique de la recherche parallèle, ou encore s'il y a plusieurs historiques différents, chaque processus n'utilise que l'information en provenance de son propre historique, il n'y a donc pas de partage d'information entre les différents processus.

En particulier, la synchronisation par tâches est un complément idéal pour l'approche 1-chemin. C'est un cas classique de maître-esclave, où un processus maître exécute ce qui correspond à une recherche séquentielle tabou en utilisant les autres processeurs pour exécuter les tâches qui demandent plus de temps de traitement. Généralement les processeurs esclaves exécutent des tâches simples et d'une courte durée. Les tâches sont définies par le processeur maître et c'est ce processeur qui lance l'exécution des tâches sur les processeurs esclaves avec les valeurs d'input (point d'interaction initial pour les tâches des processeurs esclaves). Les tâches sur

les processeurs esclaves se terminent par elles-mêmes ou sont interrompues par le processeur maître et retournent leur output au processeur maître (point d'interaction final pour les tâches des processeurs esclaves). Il n'y a pas de communication entre les processus esclaves, et il y a un seul historique de la recherche puisque toute l'information est gardée et traitée uniquement par le maître, lequel aussi initie toutes les phases de traitement parallèle. L'exécution est synchrone puisque les processeurs reçoivent du travail sur la base des besoins en information du processus maître et non pas sur la base de la disponibilité de chaque processeur esclave.

L'extension au cas p-chemins correspond à une stratégie de parallélisation avec $p > 1$ procédures de recherche tabou séquentielles différentes (recherches multiples). Les p procédures peuvent soit commencer chacune avec une solution initiale différente ou avec la même solution initiale, et soit utiliser p stratégies d'exploration différentes ou une même stratégie d'exploration. Les procédures sont lancées par un processus d'affectation des tâches aux processeurs (point d'interaction initial) et se terminent par une synchronisation sur une barrière où les procédures s'attendent mutuellement jusqu'à ce qu'elles aient toutes rencontré leur critère d'arrêt. Cette étape est suivie d'un échange d'information avec un processeur élu ou entre tous les processeurs pour déterminer la meilleure solution (point d'interaction final). Dans cette catégorie de stratégies de parallélisation de la recherche tabou, il y a plusieurs historiques différents de la recherche mais ces historiques ne sont pas partagés entre les procédures. Il y a diversité au niveau de l'information mais cette diversité ne sert au mieux qu'à maintenir implicitement et localement la divergence des chemins de recherche. Une architecture de type multi-ordinateurs à grain large sans égard à la configuration ou aux performances du réseau d'interconnexion convient pour cette catégorie de stratégies parallèles.

La catégorie suivante de cette dimension de la taxonomie est aussi caractérisée par un mode synchrone d'échange d'information, mais un niveau accru de communication permet de générer de la connaissance au niveau de chaque processeur et

d'échanger celle-ci. C'est pourquoi nous identifions cette étape comme étant du **partage synchrone** de la connaissance.

Lorsque le traitement s'effectue dans le cadre 1-chemin, le maître continue d'être le processus qui emmagasine l'information, synchronise les processus, distribue les tâches aux esclaves, mais le maître délègue une plus grande part du travail. Les processus esclaves continuent de ne pas communiquer entre eux, mais leurs tâches se complexifient en comparaison avec le cas de la synchronisation par tâches et peuvent impliquer la génération distribuée de connaissance et un processus d'exploration indépendant. Par exemple, un processus esclave peut exécuter une séquence limitée d'itérations du tabou sur un sous-ensemble du voisinage (c.a.d. intensification sur des solutions candidates prometteuses). Mais sur la requête du maître, le processus esclave retourne les résultats de son exploration et attend l'affectation d'une nouvelle tâche. Une implantation un peu plus sophistiquée "of the fan candidat list" appartient à cette catégorie.

Lorsqu'une stratégie p-chemins est adoptée, il y a partage de l'information à partir de communications synchrones s'effectuant à des points d'interactions entre plusieurs procédures de recherche tabou (recherches multiples coopérantes). Chaque procédure est divisée en plusieurs étapes sur la base d'un nombre pré-défini d'itérations de la méthode tabou. À la fin de chaque étape, les processus s'attendent mutuellement sur une barrière que tous aient complété le nombre pré-défini d'itérations. Cette période d'attente est suivie d'une phase d'échange d'information où les processus échangent entre eux et non pas avec un processus maître. Dans ce type de stratégies de parallélisation, il y a plusieurs historiques de recherche différents (un pour chaque processus) et partage complet de l'information, c'est-à-dire qu'à la fin d'une phase d'échange chaque processus a accès à toute l'information disponible sur l'espace de solutions généré par l'ensemble des procédures. L'historique d'une procédure A peut donc permettre d'identifier et corriger la stratégie de recherche d'une autre procédure B si la stratégie originale de B provoque l'exploration de

régions non intéressantes de l'espace de solutions. Un ordinateur parallèle doté d'une commutation par circuits assure le degré de connexion nécessaire entre les processeurs pour ce type de stratégies parallèles.

Pour résumer, en mode synchrone, la stratégie 1-chemin implique une communication que l'on pourrait qualifiée de verticale entre le maître et les esclaves tandis qu'il n'y a qu'une communication horizontale processus à processus en mode p-chemins. La différence entre les stratégies de synchronisation par tâches et de partage synchrone n'est pas toujours claire dans le contexte du 1-chemin, puisque cette différence se base principalement sur la quantité de travail que le maître donne à chaque esclave. Cette différence est beaucoup plus significative pour le p-chemins puisque ces stratégies correspondent à la présence ou l'absence d'échange d'information.

Les troisième et quatrième catégories de cette dimension de la taxonomie comprennent des stratégies de parallélisation asynchrones ou chaotiques de la méthode tabou. Dans ce contexte, les processus n'ont pas à s'attendre les uns les autres, le partage des connaissances étant obtenu par la mise en oeuvre d'une architecture de type "blackboard". Cette architecture est caractérisée par une sorte de mémoire centrale qui joue le même rôle que les variables globales du traitement chaotique ou d'une architecture à mémoire partagée, c'est-à-dire qu'elle emmagasine l'information partagée entre les processus. Nous définissons ces deux catégories selon la quantité, la qualité et le traitement de l'information échangée entre les processus. À noter que nous n'avons pas l'intention de classifier les procédures parallèles selon les moyens spécifiques de communiquer l'information (voir, par exemple, l'étude par Gendron et Crainic [41] ou le travail récent de Karp and Zhang [59]). Plutôt, nous nous concentrerons sur le rôle que la communication joue au niveau du partage globale de la connaissance sur l'espace de solutions lorsque plusieurs processus indépendants de recherche explorent cet espace sans synchronisation.

La troisième catégorie repose sur un partage collégial et asynchrone de l'information, nous identifierons par **collégiale asynchrone** ces stratégies de parallélisation. Chaque processeur exécute une procédure de recherche tabou séquentielle, à la différence de l'approche synchrone, il n'y a pas de points d'interaction avec d'autres procédures. Le partage de l'information en provenance de l'historique de recherche de chaque procédure s'effectue par la lecture et la mise à jour de variables partagées dans la mémoire centrale. Le choix par une procédure d'accéder ces variables partagées est fonction de la qualité relative (par rapport aux autres procédures) des régions explorées par la procédure ou encore de phases de la méthode tabou qui dépendent fortement de l'historique de la recherche (par exemple avant une phase d'intensification ou de diversification). Dans tous les cas, comme pour les stratégies p-chemins synchrones, les communications sont simples, en ce sens que tous les messages transmettent de l'information qui origine de l'exploration de l'espace de solutions par les procédures tabous concurrentes. Cependant, bien qu'il y ait partage de l'historique de recherche des différentes procédures concurrentes, étant donné le caractère asynchrone de ce partage, l'information partagée est relative à une sous-région particulière de l'espace de solutions. En ce sens, l'approche asynchrone se distingue du cas p-chemins avec partage synchrone où l'information est globale provenant de toutes les procédures, ou encore du cas p-chemins synchronisation par tâches où l'information est purement locale au niveau d'un processeur ou d'un chemin de recherche. Ce partage asynchrone de l'information entre les procédures tabou confère à ce type de parallélisation un mode de parcours de l'espace de solutions et un mode d'acquisition de la connaissance tout à fait différent de celui des approches p-chemins synchrones.

La quatrième catégorie que nous identifions par **réflexif asynchrone** de l'information échangée, distingue les stratégies de parallélisation asynchrones visant à utiliser la diversité de l'information obtenue à partir de plusieurs chemins de

recherche différents pour déduire de nouvelles connaissances sur l'espace de solutions. Ici la mémoire centrale n'a plus simplement un rôle passif d'emmagasinage de la connaissance, mais elle devient un processus de réconciliation des informations échangées entre les processus à l'aide de différentes procédures d'analyse de cette information. Le contenu des communications est analysé au niveau de la mémoire centrale pour inférer de l'information additionnelle concernant le comportement global de l'exploration en parallèle de l'espace de solutions. De nouvelles structures d'emmagasinage au niveau de la mémoire centrale peuvent être conçues, par exemple pour emmagasiner la fréquence des changements de statuts de certaines variables, ou pour créer de nouvelles listes tabous globales qui reflètent la dynamique de l'exploration parallèle asynchrone de l'espace de solutions. Aussi de nouvelles solutions peuvent être générées par des procédures qui se basent sur les solutions et le contenu des mémoires des processus individuels.

Pour les parallélisations réflexives asynchrones que nous avons réalisées, toutes les communications entre les processeurs passent par une mémoire centrale. Une architecture d'ordinateur avec mémoire partagée ou encore un ordinateur distribué avec un réseau d'interconnexion configuré en étoile conviendra mieux à ces parallélisations.

4.3.4 Les stratégies de différenciation de la recherche

La classification de Voß [103] ne prend en considération que le nombre de solutions initiales différentes et le nombre de stratégies d'exploration différentes utilisées par une implantation particulière. Ceci correspond à notre troisième dimension de la taxonomie, que nous identifions comme étant les *stratégies de différenciation de la recherche.*

L'utilisation que Voß fait de l'analogie des balles et des montagnes pour identifier les classes de sa taxonomie possède un certain côté intuitif. Nous préférons cependant référer directement à la décision de démarrer l'exploration de l'espace de

solutions à partir du même ou de différents points, et d'utiliser soit une seule ou p stratégies d'exploration différentes pour les p procédures tabous impliquées dans le traitement parallèle. Nous utilisons l'expression stratégie de différentiation de la recherche dans sa signification la plus générale qui inclut les solutions initiales, les différentes définitions du voisinage, du réglage des paramètres, des règles de gestion des mémoires, du schéma de diversification, etc. Nous avons identifié les quatre cas suivants:

SPSS: *Single (Initial) Point, Single Strategy* est le cas le plus simple et il ne permet en général qu'un bas niveau de parallélisme.

SPDS: *Single Point, Different Strategies* réfère au cas où chaque procédure utilise un ensemble différent de stratégies d'exploration mais toutes les procédures commencent avec la même solution initiale.

MPSS: *Multiple Points, Single Strategy* identifie le cas où chaque procédure démarre la recherche à partir d'une solution initiale différente, mais toutes utilisent le même ensemble de stratégies d'exploration pour la recherche de l'espace de solutions.

MPDS: *Multiple Points, Different Strategies* correspond à la classe la plus générale où chaque procédure commence avec une solution initiale différente et utilise un ensemble différent de stratégies d'exploration.

4.4 Revue de la littérature sur les algorithmes parallèles pour le tabou

Quoique la méthode tabou soit encore très récente, un certain nombre de contributions significatives ont déjà été réalisées au niveau de la parallélisation de cette méthode. Nous examinons maintenant comment notre taxonomie s'applique

à quelques-uns des algorithmes parallèles pour la méthode tabou que l'on retrouve dans la littérature.

Malek et al. [67] ont implanté et comparé des versions séquentielles et parallèles de la recherche tabou et du recuit simulé appliquées au problème du voyageur de commerce. L'expérimentation parallèle fut réalisée sur un ordinateur Sequent Balance 8000 de 10 processeurs. Les auteurs rapportent que l'implantation parallèle de la méthode tabou obtient de meilleurs résultats (meilleures solutions) que l'implantation séquentielle de cette méthode, et produit des résultats similaires ou meilleurs que les implantations séquentielles et parallèles du recuit simulé. Cette implantation parallèle de la recherche tabou peut être considérée comme une approche SPDS, 1-chemin et partage synchrone, utilisant un processus maître et quatre processus esclaves. Les tâches allouées aux processeurs esclaves correspondent à une recherche tabou séquentielle, chaque tâche exécutant une stratégie d'exploration différente. Les tâches sont allouées aux processeurs sur la base d'intervalles réguliers de temps, lorsqu'une tâche est complétée, les solutions trouvées lors de l'exploration sont transmises au processus maître. Le processus maître évalue l'intérêt des différentes régions explorées par les esclaves, celles qui semblent moins intéressantes ne sont plus considérées pour la création de nouvelles tâches d'exploration. Chaque processus esclave démarre ensuite avec une liste tabou à court terme vide, et se voit allouer une nouvelle tâche qui consiste dans la ré-initialisation d'une recherche tabou séquentielle avec comme solution initiale l'une des solutions intéressantes obtenue de l'exploration de l'espace de solutions à l'étape précédente. Il faut noter que dans le but d'implanter strictement cette stratégie, la diversification et les mémoires à long terme qui l'accompagnent ont été éliminées.

Taillard [92] a étudié des algorithmes parallèles de recherche tabou pour des problèmes de tournées de véhicules. Ses stratégies de parallélisation se basent sur une décomposition explicite du domaine de l'espace de solutions utilisant une approche MPSS, p-chemins et partage synchrone en simulation pour $p = 4$ sur une

station de travail Silicon Graphics 4D/3. Une première stratégie de parallélisation porte sur des problèmes de type Euclidiens avec une distribution uniforme des villes, et une décomposition de l'espace de solutions en régions polaires auxquelles on alloue des véhicules. Après une première décomposition initiale, chaque sous-problème est résolu à l'aide d'une recherche tabou indépendante. Les processus sont divisés en étapes sur la base du nombre d'itérations, ce nombre varie au cours du déroulement du traitement parallèle. À chaque point d'interaction, les processeurs voisins (correspondant à des régions voisines) s'échangent les tournées, les villes non visitées et les véhicules, dans le but de modifier la décomposition de l'étape précédente d'exploration et d'utiliser cette nouvelle décomposition pour la prochaine étape. Cette stratégie de parallélisation semble encourir une dégradation des performances importantes dues au niveau du balancement des charges. La seconde stratégie porte sur des problèmes non euclidiens ou des problèmes où les villes ne sont pas uniformément distribuées. La principale différence entre ces deux stratégies apparaît au niveau de la décomposition de l'espace de solutions qui cette fois repose sur une arborescence construite à partir d'un algorithme de chemins les plus courts entre les dépôts de toutes les villes; et au niveau de l'information échangée qui est constituée uniquement des meilleures solutions.

Fiechter [37] utilise aussi une approche MPSS, p-chemins et partage synchrone pour paralléliser un algorithme de recherche tabou appliqué au problème du voyageur de commerce. Les étapes de chaque processus correspondent aux phases de la méthode tabou, c'est-à-dire que les points d'interaction ont lieu lors des changements de phases de la recherche. Les opérations précises du traitement parallèle dépendent de la phase d'exécution de la méthode tabou. En phase d'intensification, une décomposition explicite du domaine d'exploration est effectuée en créant p sous-ensembles de villes, chaque sous-ensemble étant affecté à un processeur qui optimise le sous-chemin correspondant aux villes de sa décomposition. À la fin de la phase les processus se synchronisent pour reconstituer un chemin complet et pour modifier

(en décalant une partie du chemin vers un processeur voisin prédéterminé) la partie du chemin que chaque processeur aura à traiter lors de la prochaine étape. Lors de la phase de diversification, chaque processus détermine parmi son sous-ensemble de villes une liste candidate des transitions les plus prometteuses. Les processus se synchronisent alors pour échanger ces listes, de telle sorte que tous les processus possèdent la même liste candidate finale pour appliquer les transitions. Cet algorithme a été implanté sur un réseau de transputers avec une configuration en anneau. L'auteur rapporte des solutions presque optimales pour des problèmes de 500, 3000 et 10000 villes, de même que des accélérations presque linéaires.

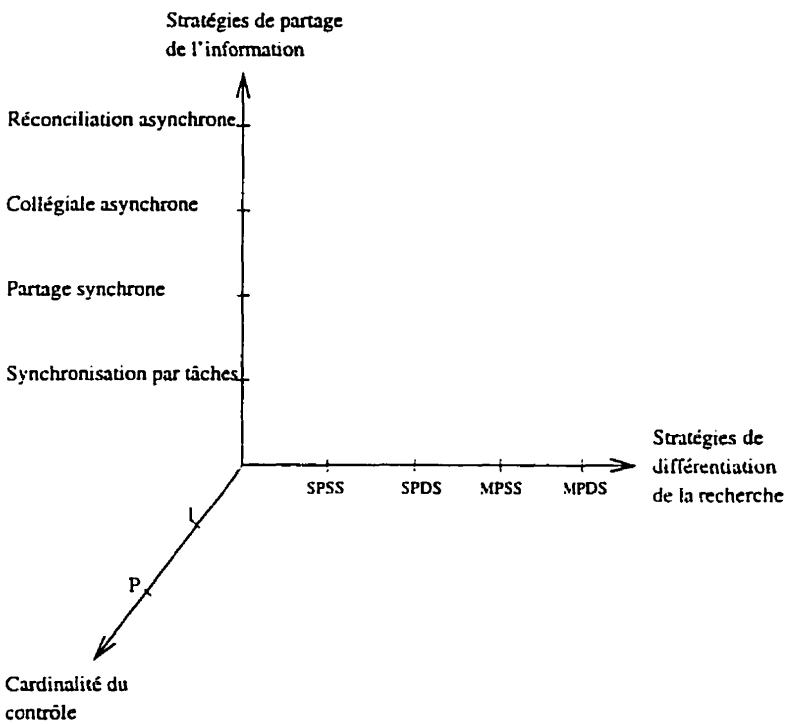


Figure 4.2 Dimensions de la taxonomie

Taillard [91] utilise une stratégie SPSS, 1-chemin et synchronisation par tâches pour la parallélisation de sa méthode tabou appliquée au problème d'affectation quadratique. À chaque itération du processus maître, le voisinage de la solution

courante est décomposé en p sous-ensembles pour constituer p tâches lesquelles sont allouées à un même nombre de processeurs. Chaque processeur évalue alors la meilleure "pairwise interchange move". Ici, étrangement, bien que le modèle de stratégie de parallélisation soit de type 1-chemin, l'implantation est de type p-chemins, c'est-à-dire qu'il n'y a pas de processeur maître. À la fin de l'exécution d'une tâche, chaque processeur sélectionne la meilleure solution du voisinage qu'il a exploré et communique cette solution à tous les autres processeurs. Ensuite, chaque processeur esclave exécute de manière redondante les tâches normalement effectuées par le processus maître: choix de la nouvelle solution courante, implantation de cette solution, faire les ajustements et mises à jour, et décomposition explicite du domaine pour la prochaine étape du traitement. Le balancement de la charge à travers la décomposition du voisinage est critique, mais aucune indication n'est donnée sur cet aspect particulier de la parallélisation. Une configuration en anneau de 10 transputers (TS00C-G20S) a été utilisée pour l'expérimentation.

Le travail de Chakrapani et Skorin-Kapov [21, 23] porte aussi sur le problème d'affectation quadratique. Leur approche de parallélisation est essentiellement de type SPSS, 1-chemin, synchronisation par tâches où la recherche est exécutée de manière séquentielle, tandis que l'évaluation des transitions candidates est effectuée en parallèle. Cependant, l'implantation est spécialement conçue pour tirer avantage de l'architecture SIMD de la CM-2: pour un problème de taille n, n^2 processeurs sont utilisés pour évaluer les transitions et communiquer l'information. Les auteurs rapportent que leurs solutions sont aussi bonnes ou meilleures que les meilleures solutions présentées dans d'autres études et que leur méthode nécessite un nombre significativement moindre d'itérations. De plus, ils ont été capables de déterminer des solutions sous-optimales à des problèmes de plus grande taille dans des temps raisonnables.

Dans le but de réduire la dégradation des performances due aux communications, Chakrapani et Skorin-Kapov [22] appliquent une stratégie similaire au

problème d'affectation des tâches aux processeurs dans un système parallèle. Ce problème d'affectation des tâches est approximé par un très gros problème d'affectation quadratique avec une matrice creuse de flots. On sait qu'à cause de la faible densité du graphe des tâches, l'implantation d'une transition (le swap d'une seule paire de tâches) n'affecte pas de manière significative les valeurs des autres transitions possibles, ce qui fait que les transitions améliorantes le demeurent, après l'implantation d'une transition. Deux types d'opérations sont alors exécutées en parallèle: les transitions candidates sont identifiées et évaluées, et en second lieu plusieurs transitions sont implantées. Pour diminuer l'erreur d'évaluation inhérente dans une telle procédure (la valeur totale des swaps multiples n'est pas égale à la somme des transitions individuelles), une phase de diversification agressive est introduite dans la procédure. De très bons résultats sont rapportés sur une configuration en hypercube 8192 de la Connection Machine CM-2.

Battiti et Tecchiolli [10] utilisent aussi le problème d'affectation quadratique pour introduire une méthode tabou faisant appel à des fonctions de hashing et pour discuter d'une parallélisation de la méthode tabou avec plusieurs procédures indépendantes. La fonction de hashing est utilisée pour "réagir" à la détection de cycles dans la recherche en modifiant la taille des listes tabous. Les auteurs analysent alors un schéma de parallélisation où plusieurs processus indépendants commencent l'exploration de l'espace de solutions à partir de configurations initiales différentes, construites de manière aléatoire. Ceci correspond à l'approche MPSS, p-chemins, synchronisation par tâches de notre taxonomie. Les auteurs obtiennent ensuite un modèle probabiliste pour évaluer le succès de cette stratégie de parallélisation. Ce modèle tend à montrer que si la procédure tabou ne cycle pas, le schéma de parallélisation par procédures indépendantes est efficace en ce sens que la probabilité de succès s'accroît pendant que la moyenne de temps pour un succès décroît en fonction du nombre de processeurs.

Taillard a aussi étudié la stratégie de parallélisation MPSS, p-chemins, synchronisation par tâches où plusieurs procédures séquentielles indépendantes s'exécutent en utilisant une solution initiale différente pour chaque procédure. La principale étude se trouve dans son article sur les méthodes tabous parallèles pour des problèmes d'ordonnancement de tâches [94]. Pour ce type de problèmes, Taillard montre que la méthode tabou (qui inclut une phase de diversification) est simple à implanter et généralement plus efficace que le recuit simulé et les procédures de "shifting bottleneck" (les deux procédures proposées alors). La méthode tabou a permis d'améliorer les meilleures solutions connues pour chaque problème de deux ensembles de problèmes tests. On a aussi été capable de trouver la solution optimale en un temps moyen polynômial à des problèmes générés de manière aléatoire où le nombre de processeurs $p << n$, le nombre de tâches à allouer (c.a.d. $p = 5$, $n = 2000$). Plusieurs stratégies de parallélisation visant l'accélération du calcul relié à l'évaluation des voisins (1-chemin, synchronisation par tâches) n'ont pas obtenues de bons résultats, soit que l'implantation n'était pas appropriée aux plates-formes matérielles alors disponibles (anneau de transputers ou Cray avec 2 processeurs), soit que les temps de communication étaient trop grands par rapport aux temps de traitement.

Taillard a ensuite examiné les bases théoriques des approches avec plusieurs processus indépendants pour des méthodes itératives comme le tabou et le recuit simulé. Ses résultats semblent indiquer que les probabilités sont très bonnes que ces parallélisations donnent de meilleurs résultats que les approches séquentielles, c'est-à-dire que la probabilité que l'algorithme parallèle obtienne un succès selon certaines conditions (en termes de la solution optimale ou presque optimale) en temps t est plus grande, que la probabilité correspondante de l'algorithme séquentiel en temps $p \times t$. Cependant, l'auteur mentionne aussi que dans plusieurs cas, la fonction de probabilité empirique des algorithmes itératifs n'est pas très éloignée d'une

exponentielle, rendant l'approche par plusieurs procédures indépendantes très efficace. Les résultats pour le problème d'ordonnancement de tâches [94] et le problème d'affectation quadratique [91] semblent justifier cette affirmation.

Les constations suivantes se dégagent de cette brève revue de la littérature sur les stratégies de parallélisation de la méthode tabou:

- L'utilisation du parallélisme peut améliorer les performances de la méthode tabou.
- La parallélisation peut entrer en conflit avec quelques-uns des mécanismes de base de la méthode tabou (par exemple, la diversification dans [67]).
- La taxonomie que nous proposons est suffisamment large pour inclure les stratégies de parallélisation déjà proposées dans la littérature.
- En dépit de différences d'implantation significatives dues à la spécificité des problèmes, les caractéristiques de la méthode tabou, l'environnement informatique, etc., très peu de stratégies différentes de parallélisation ont été expérimentées. En fait, comme illustré dans la figure 4.2, les approches synchrones semblent être la norme, le traitement parallèle étant principalement utilisé pour évaluer les transitions ou pour accélérer une stratégie de "restarting".

Dans les sections qui vont suivre, nous montrons que d'autres stratégies, identifiées par notre taxonomie, sont disponibles pour construire des procédures de recherche tabou parallèle efficaces.

4.5 Modèle et procédure tabou séquentielle

Dans la présente section, nous introduisons le problème de multicommodité sur lequel nos expérimentations ont porté et nous faisons une brève introduction de la procédure tabou séquentielle qui a été mise au point pour ce problème, le détail de cette procédure se trouve dans Crainic et al. [28].

Le problème de multicommodité localisation-allocation avec "balancing requirements" apparaît de façon typique dans un contexte de gestion à moyen terme d'une flotte hétérogène de véhicules. Le problème consiste à choisir un ensemble de dépôts, à faire l'affectation des clients à ces dépôts pour chaque type de véhicule, à planifier le trafic inter-dépôt pour tenir compte des différences entre les offres et demandes des diverses zones du territoire géographique desservi par la compagnie. L'objectif est de minimiser le coût total du système qui comprend: les coût fixes associés à une sélection donnée de dépôts, les coûts de transports entre les client et les dépôts, les coûts des mouvements requis pour balancer l'offre et la demande pour chaque type de véhicule. Le problème est formulé comme un modèle de programmation linéaire entier mixte, où les variables entières (binaires) représentent la décision de choisir ou pas le dépôt correspondant, tandis que les variables continues représentent les flots de véhicules sur les arcs du réseau. À part les restrictions habituelles au niveau des signes, deux ensembles de contraintes déterminent la région réalisable de ce problème: (i) un ensemble de contraintes liantes qui interdisent l'utilisation de dépôts non sélectionnés, et (ii) les équations de conservation du flot de l'offre pour un problème de réseau avec multicommodité sans capacité. Le modèle est formulé de la façon suivante:

$$\text{Minimiser } Z = \sum_{j\in D} f_j x_j + \sum_{r\in P}\left\{\sum_{i\in C}\sum_{j\in D}(c_{ijr}y_{ijr} + c_{jir}y_{jir}) + \sum_{j\in D}\sum_{k\in D} s_{jkr}w_{jkr}\right\}$$

$$\text{subject to} \quad \sum_{j\in D} y_{ijr} = O_{ir} \quad \forall i \in C, r \in P$$

$$\sum_{j\in D} y_{jir} = D_{ir} \quad \forall i \in C, r \in P$$

$$y_{ijr} \leq O_{ir}x_j \quad \forall i \in C, j \in D, r \in P$$

$$y_{jir} \leq D_{ir}x_j \quad \forall i \in C, j \in D, r \in P$$

$$\sum_{i\in C} y_{ijr} + \sum_{k\in D} w_{kjr} - \sum_{i\in C} y_{jir} - \sum_{k\in D} w_{jkr} = 0 \quad \forall j \in D, r \in P$$

$$y_{ijr}, y_{jir}, w_{jkr} \geq 0 \quad \forall i \in C, j \in D, k \in D, r \in P$$

$$x_j \in \{0,1\} \quad \forall j \in D$$

où C, D et P représentent respectivement, les clients, les dépôts candidats et les produits (véhicules). et

$x = (x_j)$: Variables de décision pour les dépôts;
$x_j = 1$ si le dépôt j est ouvert et 0 sinon, $j \in D$;

y_{ijr}, y_{jir} : Flots du produit $r \in P$ entre le client $i \in C$ et le dépôt $j \in D$;

w_{jkr}, w_{kjr} : Flots du produit $r \in P$ entre les dépôts $j \in D$ et $k \in D$;

f_j : coût fixe pour le dépôt $j \in D$;

c_{ijr}, c_{jir} : Coût de transport unitaire pour le produit $r \in P$ entre le client $i \in C$ et le dépôt $j \in D$;

s_{jkr}, s_{kjr} : Coût de transport unitaire pour le produit $r \in P$ entre les dépôts $j \in D$ et $k \in D$;

O_{ir} : Offre au client $i \in C$ pour le produit $r \in P$;

D_{ir} : Demande au client $i \in C$ pour le produit $r \in P$.

La formulation fait apparaître une structure de réseau, en particulier pour les variables binaires x_j, ce réseau devient un problème de flots dans un réseau à coût minimum avec multicommodité et sans capacité. Cette propriété a été utilisée

pour définir la procédure séquentielle de recherche tabou pour ce problème. Pour faciliter la présentation du développement de la parallélisation, nous avons illustré les principales caractéristiques de cette procédure dans la figure 4.3.

L'espace de recherche est défini à partir du vecteur x des variables de décisions qui spécifient la *configuration des dépôts*. Le *contexte* d'une solution correspond au vecteur x des variables de décisions. Il peut y avoir plusieurs contextes pour une fonction objectif donnée, mais pour un contexte particulier il n'existe qu'une seule valeur de fonction objectif possible. Pour chaque vecteur de configuration x, les valeurs optimales des variables de flot continue $y^*(x)$ et $w^*(x)$, de même que $Z^*(x)$ qui correspond à la valeur de la fonction objectif, peuvent être calculées en résolvant un problème de flot dans un réseau sans capacité et multicommodité. Dans cet espace de recherche, la stratégie vise à (i) déterminer le bon nombre de dépôts ouvert, et (ii) à trouver la meilleure configuration étant donné un certain nombre de dépôts ouverts. Ceci est obtenu en combinant une recherche locale avec des phases d'intensification et de diversification.

Au niveau de la recherche locale, le voisinage d'une solution x est constitué par toutes les solutions qui peuvent être obtenues en exécutant une des transitions suivantes:

- Add: Ouverture d'un dépôt actuellement fermé;
- Drop: Fermeture d'un dépôt actuellement ouvert;
- Swap: Ouverture d'un dépôt et fermeture d'un autre dépôt;

Un tel voisinage est habituellement trop large, une approche par échantillonnage est utilisée pour construire une liste candidate, c'est-à-dire un sous-ensemble choisi au hasard de voisins dans le voisinage de la solution courante. L'évaluation exacte des voisins dans la liste candidate revient à résoudre un problème de flot sur les réseaux, ce qui demande trop de temps de calcul. Des *fonctions substituts* (surrogates) sont utilisées dans la plupart des cas pour faire une approximation de la

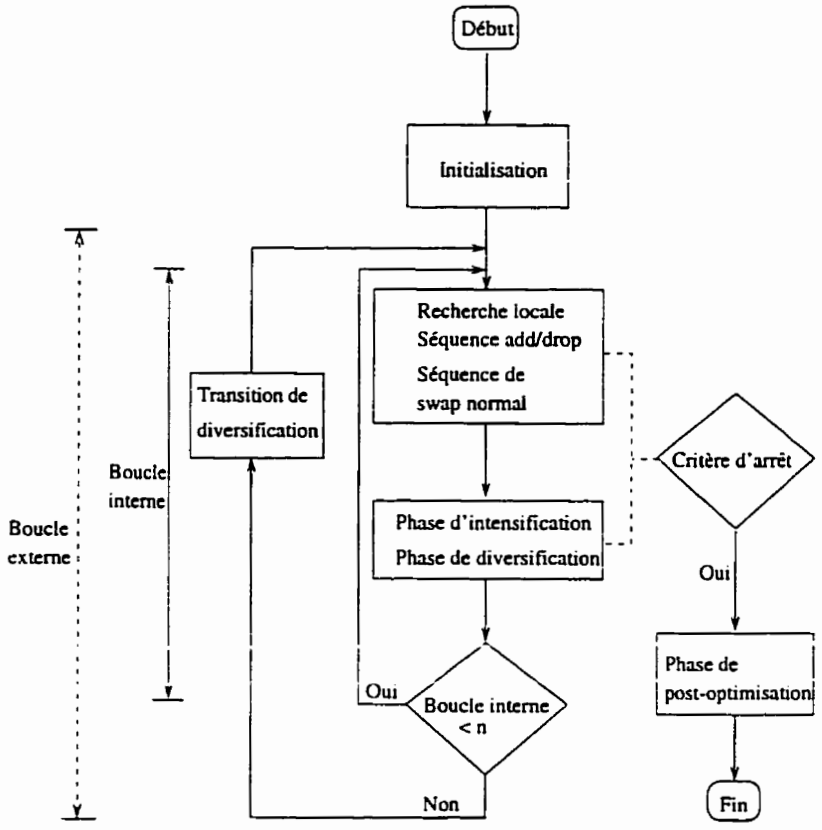


Figure 4.3 Procédure séquentielle de recherche tabou

valeur de la fonction objectif de chaque voisin de la liste candidate. La valeur exacte de la fonction objectif est cependant calculée lorsqu'une solution voisine est choisie et implantée.

La présente procédure séquentielle de recherche tabou combine une recherche locale avec des phases d'intensification et de diversification, pour se terminer par une phase de post-optimisation. La recherche locale consiste en une séquence de add/drop qui s'arrête lorsqu'un nombre pré-défini d'itérations ont été exécutées sans amélioration de la fonction objectif. Cette séquence est suivie par une séquence de swaps normal où à chaque itération, la meilleure solution candidate évaluée selon les fonctions substituts est implantée peu importe son impact réel sur la fonction objectif. Cette phase est initiée à partir de la meilleure solution trouvée lors de la précédente phase de add/drop. Lorsque la meilleure solution obtenue dans la phase de swaps normal est réalisable, une phase d'intensification est immédiatement exécutée, sinon une nouvelle phase de recherche locale démarre et se poursuit au moins jusqu'à ce qu'une solution locale réalisable soit trouvée.

Les séquences de add/drop et de swap utilisent différentes listes tabous à court terme. Pour les transitions add et drop, les listes tabous enregistrent les derniers dépôts ouverts ou fermés, et interdisent l'inversion de ces transitions. Les listes tabous du swap enregistrent les swaps récemment exécutés comme étant des paires de dépôts et interdisent l'inversion ou la répétition de ces transitions. Il faut noter que les listes tabous à long terme de la diversification influencent également le statut de la liste candidate lorsque la recherche locale est exécutée.

Une phase d'intensification consiste en une séquence de swaps strict. Cette phase démarre à partir de la meilleure solution identifiée lors de la phase précédente de recherche locale, et implante seulement les transitions qui améliorent la solution courante (d'où le qualificatif de swap strict). Une séquence de phases de recherche locale et d'intensification est appelée une *boucle interne*. Après l'exécution de N boucles internes, la procédure de recherche est réorientée vers des régions de l'espace

de solutions non encore explorées en exécutant une diversification, ce qui complète une *boucle externe.*

Une phase d'intensification est donc suivie d'un nouveau cycle de la boucle interne (commençant par une nouvelle phase de recherche locale) ou d'une phase de diversification (fin d'un cycle de la boucle externe). Une transition de diversification est exécutée à partir de la meilleure solution globale trouvée depuis le début de la procédure de recherche tabou. La diversification est basée sur une mémoire à long terme qui enregistre le "niveau d'activité" de chaque dépôt, c'est-à-dire le nombre de fois que le statut du dépôt a été modifié (d'ouvert à fermer ou vice-versa). En se basant sur les valeurs qui se trouvent dans cette mémoire, on pré-sélectionne un nombre fixe de dépôts dont le niveau d'activité est le plus bas, ensuite le statut de ces dépôts est inversé. Considérant le fait que les valeurs dans la mémoire à long terme tendent à évoluer plutôt lentement, une autre mémoire a été utilisée pour enregistrer le dernier ensemble de dépôts choisi pour une diversification. Cette liste est utilisée à la fois pour exclure la prise en compte de certains dépôts dans les phases suivantes de diversification et pour prévenir un renversement trop rapide des transitions durant les étapes de recherche locale qui suivent.

La procédure complète de recherche tabou démarre à partir d'une solution initiale et exécute une séquence de boucles externes jusqu'à ce que le critère d'arrêt soit rencontré. Dans la présente implantation, ce critère d'arrêt est le nombre total d'itérations depuis le début du lancement de la procédure de recherche tabou. Lorsque ce nombre pré-défini d'itérations a été exécuté, une phase de post-optimisation commence, qui vise à identifier l'optimum local se trouvant dans la région correspondant à la meilleure solution produite par la recherche tabou. Cette phase consiste dans une exploration exhaustive du voisinage de cette meilleure solution. Des fonctions substituts sont utilisées pour évaluer les transitions, tandis que des évaluations exactes déterminent la première transition améliorante à être implantée.

Cette procédure se poursuit aussi longtemps que des solutions strictement améliorantes sont trouvées.

Plusieurs paramètres influencent l'efficacité de la recherche: la taille des listes tabous, la taille des séquences de swaps et de add/drops, les probabilités de sélection des transitions add, drop et swap, la variation de ces probabilités durant la recherche, la solution initiale qui est choisi, etc. Crainic et al. [28] ont étudié ces questions, et montrent en particulier, que plusieurs combinaisons de paramètres peuvent être efficacement utilisées pour différents types de problèmes.

4.6 Environnement d'expérimentation

Seize problèmes ont été utilisés pour les tests: 12 ont été générés de manière aléatoire (P1 à P12), et quatre (P13 à P16) sont basés sur une application réelle [27]. Les problèmes aléatoires ont 44 dépôts (variables entières) et 220 clients, un ou deux produits, ce qui donne plus de 7000 et 14000 variables continues. Pour les quatre derniers problèmes, il y a 130 dépôts, et 56616 variables continues. Les problèmes P7 à P12 sont semblables aux problèmes P1 à P6 sauf que les coûts fixes ont été multipliés par 10. De façon similaire, les problèmes P14 à P16 ont été obtenus en multipliant les coûts fixes du problème P13 respectivement par 10, 100 et 1000. Les dimensions des problèmes sont indiquées dans le tableau 4.2.

Problèmes	1/7	2/8	3/9	4/10	5/11	6/12	13-16
Dépôts	44	44	43	44	44	44	130
Clients	219	219	220	219	219	220	289
Produits	1	2	1	2	1	2	12
Liens clients	5260	10520	5282	10516	5262	10588	45936
Liens dépôts	1892	3784	1892	3784	1892	3612	10680

Tableau 4.2 Caractéristiques des problèmes tests

Toutes les procédures sont arrêtées après 300 itérations, et la qualité de la

solution est évaluée en calculant le *gap*, en pourcentage, entre la meilleure solution déterminée par chaque procédure et la solution optimale calculée par un algorithme d'énumération implicite par évaluation et séparation [42]. Notons que l'objectif est d'illustrer la taxonomie, non de faire un long ajustement de certaines procédures pour un ensemble particulier de problèmes. Nous n'avons donc pas calibré chaque procédure individuelle pour obtenir les meilleures performances possibles sur chaque ensemble de problèmes. À la place, le meilleur réglage de paramètres pour la recherche tabou séquentielle [28] a été utilisé pour tous les tests rapportés dans ce chapitre.

Tous les tests ont été exécutés sur réseau hétérogène de stations de travail SUN-Sparc. Les communications sont traitées par notre propre ensemble de procédures, écrites en C, et qui utilisent le protocole TLI/UDP, modifié pour assurer que tous les messages atteignent correctement leur destination. La recherche tabou est programmée en Fortran77, tandis que le problème de flot à coût minimum est résolu par le code RNET [71]. Nous avons conçu et testé plusieurs variantes de la recherche tabou parallèle synchrone et asynchrone que nous examinerons dans les prochaines sections.

La définition d'une stratégie d'exploration de la recherche tabou porte sur plusieurs facteurs: le réglage des paramètres de recherche, le type des listes tabous et des mémoires de même que leurs mécanismes de mise à jour, l'existence et la conception de phases d'intensification et de diversification, etc. Dans le but de faciliter la comparaison entre une procédure séquentielle et des versions parallèles de celle-ci, nous avons utilisé au niveau de toutes les implantations parallèles le même mécanisme de recherche défini pour la procédure séquentielle, en modifiant les valeurs de certains paramètres clés de recherche. Ainsi, les meilleurs réglages obtenus pour les paramètres de recherche de la procédure séquentielle sont utilisés pour les approches SPSS et MPSS. Lorsqu'il faut utiliser différentes stratégies de recherche pour réaliser les approches SPDS et MPDS, d'une procédure à l'autre,

nous faisons varier la taille des listes tabous à court et à moyen terme, le nombre d'itérations consécutives de add/drop sans amélioration, et le nombre de dépôts fixés temporairement pour une transition de diversification. Pour les stratégies de type MP.., nous obtenons un ensemble de p solutions initiales différentes en fermant arbitrairement un certain nombre de dépôts, différentes solutions initiales sont donc obtenues en faisant varier ce nombre de dépôts fermés. Les résultats du calibrage pour la procédure de recherche séquentielle [28] sont utilisés pour définir les solutions initiales requises et les variantes de la recherche tabou.

4.7 Stratégies de parallélisation synchrones

Dans la présente section et la section suivante nous illustrons notre taxonomie en décrivant l'implantation, le comportement et les performances de quelques stratégies de parallélisation construites en faisant varier la cardinalité de la recherche (le nombre de chemins), les formes de synchronisation et les stratégies de différentiation. Nous analyserons les sources de dégradation des performances pour chaque stratégie de parallélisation, l'impact de paramètres importants, tels le nombre d'itérations entre chaque point d'interaction, le nombre de processeurs, etc. sur l'efficacité de la stratégie de parallélisation et la qualité des solutions.

Rappelons que la dimension de la taxonomie portant sur la cardinalité de la recherche permet de distinguer les stratégies de parallélisation de la méthode tabou sur la base du nombre de chemins différents (1 ou p) qui explorent concurremment l'espace de solutions d'un même problème.

La dimension portant sur le partage de l'information dans les catégories synchrones distingue les stratégies de parallélisation selon qu'il y a absence d'échange d'information entre les procédures (synchronisation par tâches, RS) ou présence d'échange d'information (partage synchrone, KS). On retrouve au niveau de la synchronisation par tâches les approches dites de maître-esclaves (1-RS) où il n'y a pas de génération de connaissances au niveau des processeurs esclaves et donc pas de

partage, et les approches où il y a p chemins d'exploration indépendants sans communication entre les différentes procédures séquentielles (recherches multiples). Au niveau des stratégies basées sur le partage synchrone de l'information, on peut concevoir des stratégies de parallélisation de la méthode tabou où un processus maître délègue suffisamment de travail aux processus esclaves pour qu'il y ait génération distribuée de connaissances au niveau de ces processeurs esclaves, ces connaissances étant partagées avec le processus maître pour être emmagasinées de manière permanente au niveau de ce processeur maître (1-KS). Dans cette même catégorie du partage synchrone, des stratégies de parallélisation peuvent être conçues où plusieurs procédures de recherche tabou se synchronisent pour partager de l'information lors de phases de communications inter-processeurs (p-KS).

La troisième dimension de la taxonomie distingue les stratégies de parallélisation selon que les solutions initiales de chaque procédure sont identiques ou différentes (SP.., MP..), ou encore selon que les stratégies d'exploration sont les mêmes pour toutes les procédures ou différentes pour chacune d'elles (..SS, ..DS). Nous allons maintenant examiner un exemple d'implantation de chacune de ces stratégies de parallélisation.

4.7.1 L'approche maître-esclave

Nous avons conçu une procédure de recherche tabou parallèle synchrone qui combine la cardinalité 1-chemin de la première dimension de la taxonomie avec la synchronisation par tâches de la deuxième dimension et la stratégie de différentiation SPSS de la troisième dimension, nous référerons à cette procédure parallèle par l'abréviation 1-RS SPSS. Cette procédure parallèle correspond à un cas classique de maître-esclave, où le maître exécute ce qui correspond à une recherche tabou séquentielle en utilisant les autres processeurs pour faire exécuter des tâches qui demandent beaucoup de traitement. Il n'y a pas de communication entre les processus esclaves; ceux-ci échangent de l'information seulement avec le processus maître,

lequel initie les communications pour distribuer les tâches et obtenir les résultats du traitement fait par les processus esclaves. Ainsi, l'information concernant l'état de la recherche de la procédure parallèle est gardée et maintenue exclusivement par le maître. Les méthodes développées par Taillard [91], et par Chakrapani et Skorin-Kapov [21, 23, 22] (quoique sur une échelle beaucoup moins grande) sont des exemples de ce type d'approche.

Dans l'implantation que nous avons réalisée, le maître délègue à chaque processeur esclave l'évaluation, à l'aide de fonctions substituts, de $\frac{n}{p-1}$ voisins à partir d'une liste tabou fournie par le processus maître. Lorsqu'un processeur esclave a terminé l'évaluation des $\frac{n}{p-1}$ voisins qui lui ont été alloués, il calcule la valeur exacte du voisin ayant obtenu la meilleure évaluation et retourne au processeur maître l'identité de ce voisin ainsi que la valeur de la fonction objectif de ce dernier. Le calcul de la valeur exacte de $p-1$ voisins permet au processeur maître, moyennant un coût minime, de choisir parmi plusieurs solutions qui ont été évaluées de manière exacte.

4.7.2 La stratégie de probing

Il s'agit ici d'une procédure parallèle qui combine la cardinalité 1-chemin avec le partage synchrone et la stratégie de différentiation SPSS, nous identifions cette procédure parallèle par 1-KS SPSS, cette procédure est similaire à l'approche par "probing" proposée par Glover [44]. Le processus maître emmagasine toute la connaissance accumulée suite à l'exploration de l'espace de solutions et il synchronise l'exécution du traitement parallèle en contrôlant l'affectation des tâches aux processeurs esclaves (qui ne communiquent pas entre eux). Il y a cependant une plus grande délégation du travail aux processeurs esclaves. En effet, au lieu de faire simplement l'évaluation des voisins de la solution courante à l'aide d'une fonction substitut, l'évaluation des n transitions du voisinage par les $p-1$ processeurs se fait par l'exécution de quelques itérations de recherche locale pour chacun des voisins, d'où

le nom de "probing" pour cette stratégie de parallélisation. Le maître sélectionne ensuite la séquence d'itération, parmi les $p-1$ processeurs, qui a résulté dans la meilleure amélioration de la fonction objectif et implante cette séquence en faisant une mise à jour appropriée des listes tabous et des mémoires.

La stratégie 1-KS est donc une généralisation de l'approche maître-esclave, il n'y a encore qu'une seule recherche tabou exécutée, avec un accroissement de la quantité d'information disponible pour le processus maître. Quoiqu'une approche de type SPDS aurait pu être définie (c.a.d. les processus esclaves effectueraient leurs quelques itérations de recherche locale chacun avec une stratégie différente d'exploration), nous avons préféré implanter uniquement la stratégie SPSS qui se conforme au paradigme maître-esclave. À noter que dans les implantations maître-esclaves que nous avons réalisées (1-RS SPSS et 1-KS SPSS), le voisinage est complètement exploré. En effet, puisque plusieurs processeurs sont disponibles, le mécanisme d'échantillonnage n'est pas utilisé, et chaque processus esclave évalue toutes les transitions qui lui sont allouées.

À noter que l'approche 1-RS SPSS exploite le parallélisme obligatoire de la méthode tabou visant essentiellement à accélérer le temps de calcul sans changer le parcours de l'espace de solutions par rapport à celui exécuté par la procédure séquentielle. Cependant le parcours de l'espace de solutions exécuté par une procédure 1-RS SPSS n'est pas tout à fait le même que celui d'une procédure séquentielle puisque la procédure 1-RS SPSS évalue toutes les solutions du voisinage contrairement au sous-ensemble de la liste candidate pour la procédure séquentielle. On pourrait cependant obtenir un parcours identique en forçant la procédure séquentielle à évaluer tout le voisinage. La disponibilité de plusieurs processeurs permet l'évaluation de tous les voisins pour la procédure 1-RS SPSS, ce qui se traduit comme nous le verrons plus loin par une amélioration générale dans la qualité des solutions.

4.7.3 Procédures parallèles par recherches multiples

Aucun travail particulier d'implantation n'est requis pour les quatre procédures parallèles suivantes basées sur la combinaison de la cardinalité p-chemins avec la synchronisation par tâches, et une des quatre stratégies de différentiation de la troisième dimension de la taxonomie: SPSS, SPDS, MPSS et MPDS. Nous avons identifié ces quatre procédures parallèles par p-RM SPSS, p-RM SPDS, p-RM MPSS et p-RM MPDS. À noter que parce que l'approche p-RM SPSS se réduit à p répétitions du même chemin de recherche, nous n'avons pas implanté cette stratégie.

4.7.4 Recherches coordonnées

Nous avons implanté trois procédures de recherche tabou parallèle en combinant la cardinalité p-chemins avec la catégorie partage synchrone (identifié par p-KS) de la deuxième dimension sur la base des stratégies de différentiation: SPDS (Malek et al. [67] ont également développé ce type d'algorithme), MPSS (l'algorithme de Taillard [92] appartient à cette catégorie) et MPDS (nous n'avons pas fait de combinaison avec SPSS puisque cette approche se réduit à p répétitions de la même stratégie d'exploration). Ici, p procédures séquentielles de recherche tabou explorent l'espace de solutions et échangent de l'information à des points d'interaction prédéterminés. Chaque procédure est décomposée en $\frac{t}{\delta}$ étapes où t est le critère d'arrêt de chacune des procédures (essentiellement c'est le nombre d'itérations de chaque procédure) et δ est le nombre d'itérations d'une étape entre deux points d'interaction. Soit s_{ij} la meilleure solution trouvée à l'étape j par une procédure i depuis le dernier point d'interaction.

Définition 4.5 *Nous dirons qu'une solution s est améliorante (pour un problème de minimisation) si la solution s a été visitée à l'étape j par la procédure i et que $s < s_{ij}$.*

Dans ce cas la solution améliorante devient la meilleure solution de l'étape j: $s_{ij} = s$. À chaque point d'interaction, chaque procédure envoie à toutes les autres procédures ses solutions améliorantes visitées depuis le dernier point d'interaction. Lorsqu'il s'agit d'une stratégie de différentiation de type SPDS, toutes les procédures commencent l'étape suivante avec comme solution initiale la meilleure solution améliorante parmi toutes les procédures lors de l'étape précédente. Pour les stratégies de différenciation basées sur des solutions initiales différentes, MPSS et MPDS, les p meilleures solutions améliorantes sont identifiées et distribuées parmi les p processeurs. Nous identifierons par *solution partagée* cette ou ces solutions qui servent de solutions initiales à une ou plusieurs procédures au début d'une nouvelle étape d'exploration.

Pour l'implantation que nous venons de décrire de la recherche coordonnée, chaque procédure est forcée d'accepter la solution partagée reçue à un point d'interaction. Nous avons considéré une autre alternative à cette incorporation automatique de la solution partagée. Cette alternative consiste pour chaque procédure pr_i à utiliser la solution partagée uniquement si la valeur de celle-ci est meilleure que celle de la meilleure solution trouvée par le procédure pr_i lors de l'exploration effectuée à l'étape précédente. L'approche consistant à utiliser automatiquement la solution partagée est équivalente à faire une diversification forçée sans qu'elle soit basée sur l'historique local de la recherche. Par contre la deuxième alternative vise à rendre plus effective et plus cohérente l'utilisation des mémoires de la recherche tabou.

Lorsqu'une solution partagée est utilisée par une procédure au début d'une nouvelle étape de recherche, les différentes listes tabous et les mémoires de la procédure ne sont pas mises à jour. La recherche se poursuit, et éventuellement il pourra y avoir des phases de diversification qui seront exécutées en utilisant un contenu des mémoires à long terme qui pourra n'avoir que très peu de relation avec la solution partagée. Mais l'historique de recherche enregistré dans les diverses mémoires à long terme d'une procédure reflète le comportement particulier de l'exploration effectuée par celle-ci. C'est d'autant plus vrai lorsque les procédures

utilisent différentes stratégies d'exploration (c.a.d. pour les implantations SPDS et MPDS), et ce phénomène s'accentue lorsque des processeurs avec des caractéristiques différentes sont utilisés. Ainsi, cette approche permet à la fois d'exécuter des phases de diversifications "normales" (spécialement lorsque les étapes de chaque procédure sont basées sur des séquences relativement longues) et de conférer une couleur locale à une exploration qui potentiellement démarre avec la même solution initiale pour plusieurs procédures. On peut faire l'hypothèse qu'une plus grande région de l'espace de solutions sera explorée.

Nous avons préféré et implanté cette approche plutôt que la stratégie alternative qui impose non seulement la solution partagée, mais également le contexte relatif à cette solution (ceci est l'approche de Malek et al. [67] où les listes tabous sont réinitialisées). Cette dernière stratégie est équivalente à redémarrer arbitrairement la recherche, et ainsi réduit l'algorithme parallèle à une juxtaposition de plusieurs recherches tabous de courtes durées. De plus, si on ne fait pas attention à l'implantation, et si les points d'interaction sont séparés par seulement quelques itérations, il peut y avoir des phases de la méthode tabou qui ne seront jamais exécutées: probablement la diversification à partir d'une solution locale au processus (la diversification est éliminée dans [67]), et même d'intensification.

4.8 Résultats expérimentaux des stratégies synchrones

Nous avons implanté et testé les différentes procédures de recherche tabou parallèle synchrone décrites dans les sections précédentes. Nous avions comme objectifs pour notre plan d'expérimentation d'explorer l'impact de la parallélisation sur l'efficacité de la recherche tabou, d'identifier les stratégies de parallélisation les plus efficaces, de mesurer l'effet de facteurs comme le nombre de processeurs, le nombre de points d'interaction et la profondeur du probing sur le comportement de la recherche tabou parallèle. Nous avons aussi voulu mesurer l'effet des

différentes sources de dégradation des performances sur les temps de calcul et comparer les stratégies les unes par rapport aux autres de ce point de vue. Bien que les procédures parallèles décrites dans les sections précédentes peuvent être implantées sur différentes architectures d'ordinateurs parallèles puisque le degré d'interaction entre les tâches est généralement faible (sauf pour le cas 1-RS et dans une moindre mesure pour l'approche par probing), ces procédures conviennent mieux à des architectures d'ordinateur à grain large tels les multi-ordinateurs et les réseaux de stations de travail. Pour cette raison, nous croyons que le faible débit du réseau d'interconnexion de notre environnement de même que sa configuration ne changeront pas les principales conclusions de cette étude quant à la relation existant entre les stratégies au niveau des temps de calcul, ce même, à travers d'autres types d'architectures avec des réseaux d'interconnexion plus performants et mieux configurés (évidemment sauf pour le cas 1-RS).

Les résultats des tests effectués pour ces procédures parallèles synchrones se trouvent condensés dans 4 tableaux. Le tableau 4.3 contient les résultats obtenus pour les deux stratégies de parallélisation maître-esclaves (1-RS SPSS et 1-KS SPSS). Les résultats obtenus pour les procédures basées sur les approches de différentiation SPDS, MPSS, et MPDS sont présentés respectivement dans les tableaux 4.4, 4.5 et 4.6 selon qu'il y a échange d'information (p-KS) ou pas d'échange (recherches multiples) entre les processus. Les résultats sont rapportés pour $p = 4$, 8 et 16 processeurs, et sont graphiquement synthétisés dans la figure 4.2. Toutes les procédures se terminent après 300 itérations (plus la phase de post-optimisation). Les tableaux montrent l'information suivante:

- Le *Gap*, en pourcentage, entre la meilleure solution trouvée par chaque procédure et la solution optimale.
- Lorsque la meilleure solution est trouvée lors de la phase de post-optimisation, nous indiquons entre parenthèse le gap correspondant de la solution à partir de laquelle la phase de post-optimisation a été commencée.

- L'itération (*Iter*) à laquelle la meilleure solution fut trouvée. Lorsque la meilleure solution est trouvée lors de la phase de post-optimisation, nous indiquons l'itération correspondant à la solution à partir de laquelle la phase de post-optimisation a été initiée.

- Le gap (*Seq*) entre la solution optimale et celle trouvée avec la procédure séquentielle.

Notons que les résultats de la procédure séquentielle utilisés dans ce travail sont généralement meilleurs que ceux rapportés dans [28]. Ces améliorations proviennent d'une observation faite durant le développement des implantations parallèles: il semble que lorsque le nombre d'itérations de la procédure est faible, il est plus efficace de ne pas permettre à la recherche d'explorer des régions non réalisables. Notons également que la procédure séquentielle fut calibrée en utilisant 8 des 12 problèmes P1 à P12.

Avant de comparer les différentes stratégies parallèles entre elles, et de regarder la relation existant entre le nombre de processeurs et l'efficacité des différentes procédures, nous examinerons certains détails d'implantation, nous ferons l'analyse des sources de dégradation des performances et étudierons le comportement d'exploration de l'espace de solutions pour chaque procédure parallèle.

4.8.1 Procédures 1-RS SPSS

À chaque itération de la stratégie 1-RS SPSS, le processeur maître envoie pour évaluation n voisins à $p-1$ processeurs, et reçoit $p-1$ solutions et leur contexte en provenance des $p-1$ processeurs. La taille des messages échangés entre le maître et les esclaves n'est constituée que de quelques octets. Par contre, la fréquence des messages et la topologie des communications inter-processeurs constituent une source importante de dégradation des performances pour cette stratégie. En effet, à chaque itération le processeur maître doit exécuter une communication de diffusion (broadcast) de $p-1$ messages différents à $p-1$ processeurs (single-node-scatter)

ce qui implique que les $p-1$ messages doivent être envoyés séquentiellement par le processeur maître. Si on tient compte du temps de mise en route (startup time) t_s qui est constant (ne dépend pas de la taille des messages) et considérable pour les stations de travail qui communiquent en utilisant le réseau Éthernet, le coût pour chaque itération qui est de $(p-1) \times t_s$ sera non négligeable. Le processeur maître doit aussi faire la réception de $p-1$ messages différents en provenance des $p-1$ processeurs esclaves (single-node-gather). Ici on peut faire l'hypothèse que les $p-1$ messages sont envoyés en parallèle au processeur maître, donc un coût de seulement t_s pour l'envoi, mais il peut y avoir congestion au niveau de la réception par le maître. Enfin, après réception des messages en provenance des processeurs esclaves, le maître doit comparer les solutions entre elles pour choisir la meilleure, cette opération de comparaison est dans $O(p-1)$ et fait partie des opérations supplémentaires qui découlent de la parallélisation de la procédure séquentielle. Le degré de concurrence de la stratégie de parallélisation 1-RS SPSS est limité par la taille du voisinage, il pourra varier d'une itération à l'autre et entre les différentes phases de la méthode tabou. Pour les problèmes que nous avons testés, le degré de parallélisme n était supérieur à $p-1$ pour toutes les itérations.

4.8.2 Procédures de probing

Une augmentation du nombre de processeurs semble bénéfique à la stratégie maître-esclave, mais nos résultats indiquent qu'un gain plus impressionnant peut être obtenu en augmentant le niveau de connaissances utilisées pour gouverner la recherche du processus maître. En fait, la stratégie de probing (1-KS SPSS) surpasse de manière significative l'approche classique maître-esclave, et apparaît même comme une approche compétitive sur l'ensemble des méthodes. Il y a deux paramètres auxquels nous avons dû porter un peu plus d'attention lors de l'implantation de cette stratégie de parallélisation: la profondeur du probing (nombre d'itérations), et la définition d'une itération du maître, ou itération de probing.

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	1-RS SPSS						
P1	0	27	0	27	0	29	0
P2	0.28 (0.75)	164	0.61 (0.74)	220	0.23 (0.25)	68	0.42 (1.48)
P3	0.03	214	0	212	0	293	0.01 (0.8)
P4	0.42 (0.79)	27	1.03 (1.05)	175	0.15 (0.47)	23	0.90 (1.1)
P5	1.49 (1.74)	223	0	130	0	140	0.77 (1.45)
P6	0 (0.08)	186	0.48 (0.73)	192	0 (0.18)	18	0.42 (3.67)
P7	0.07	245	0.07	179	0	185	0.07
P8	0.79	58	0.79	58	1.32	115	2.15
P9	0	285	0	46	0.82	27	0
P10	0.49	164	0.52	183	0.3	180	1.49
P11	2.22	205	1.95	229	2.55	23	1.52
P12	0	165	0	86	0.11	25	0.11
P13	0.01 (0.06)	42	0 (0.04)	74	0.02 (0.06)	13	0.002 (0.07)
P14	0.06 (1.26)	290	0.02 (0.32)	117	0.02 (0.77)	162	0.02 (0.59)
P15	1.38	95	1.22	145	0	170	1.23 (1.54)
P16	0.89	129	41.26	90	0.14	229	0.89 (9.21)
	1-KS SPSS						
P1	0	28	0	30	0	30	0
P2	0.21 (0.62)	20	0.21 (0.62)	16	0.14 (0.55)	18	0.42 (1.48)
P3	0.01	28	0	222	0	50	0.01 (0.8)
P4	0.14 (0.44)	30	0.12 (0.19)	108	0.15	199	0.90 (1.1)
P5	0.65 (0.93)	104	0.65 (0.68)	40	0	267	0.77 (1.45)
P6	0.05 (0.07)	204	0.05 (0.07)	197	0.09	164	0.42 (3.67)
P7	0	124	0.07	65	0	95	0.07
P8	0	162	0	32	0	71	2.15
P9	0	129	0	258	0	271	0
P10	1.42	282	0.3	270	0	98	1.49
P11	1.38	113	0	131	0	104	1.52
P12	0.23	39	0.18	109	0	61	0.11
P13	0.01 (0.03)	21	0.002 (0.02)	51	0.002 (2.93)	23	0.002 (0.07)
P14	0.06 (0.26)	45	0.02 (0.12)	47	0.08 (0.12)	283	0.02 (0.59)
P15	1.22	90	0.78 (0.59)	294	0.59	182	0.23 (1.54)
P16	0.89	91	0.44	182	0.89	46	0.89 (9.21)

Tableau 4.3 Implantations maître-esclaves

Dans une première variante de la procédure, nous avions considéré qu'une itération du processus maître devait correspondre au nombre total d'itérations de probing exécutées par les processus esclaves. Dans cette variante de l'implantation des procédures de probing, peu importe le nombre d'itérations de probing inclus dans une tâche des processus esclaves, le processus maître ne compte qu'une seule itération pour chaque étape d'exécution synchrone. Nous avons constaté cependant que cette implantation requiert que plusieurs paramètres critiques de la méthode tabou soient réajustés: le nombre total d'itérations du processus maître (ceci contrôle le temps d'exécution total de la recherche), le nombre total de add/drop consécutifs sans amélioration de la solution lors d'une recherche locale, et le nombre d'itérations où le statut d'un dépôt modifié par une diversification doit rester tabou. En fait, les valeurs données à tous ces paramètres doivent être multipliées par la profondeur de la phase de probing, et sont par conséquent critiques lorsque de longues séquences de probing sont utilisées. Par exemple, des valeurs trop grandes pour le nombre total de add/drop consécutifs sans amélioration font que la procédure tabou devient essentiellement une procédure de add/drop, sans avoir le temps d'exécuter des phases de swaps, d'intensification ou de diversification. De même, si la durée du statut tabou relié à la diversification est trop longue, l'efficacité de la recherche peut être significativement diminuée.

Cette phase ardue d'ajustement des paramètres n'est pas requise si on utilise une définition alternative de la procédure de probing consistant à compter une itération du maître pour chaque itération de probing exécutée par les processus esclaves. Ainsi, le maître met à jour non seulement la solution courante et les mémoires associées mais aussi son compteur d'itérations.

Le temps de calcul de cette stratégie de parallélisation est évidemment fonction du nombre d'itérations du processus maître, mais également de la profondeur de la phase de probing à chaque itération du processus maître. À travers nos tests, nous avons pu réaliser que l'accroissement de la profondeur du probing n'apportait

pas nécessairement une amélioration de la qualité de la meilleure solution trouvée. On explique ce phénomène en notant que le probing explore le voisinage de manière moins systématique que la recherche locale: certains voisins ne sont jamais évalués. ce qui a pour conséquence d'augmenter la possibilité que certaines vallées plus étroites de l'espace de solutions ne soient pas explorées par les procédures de recherche. On peut essayer de remédier à ce problème en accroissant le nombre d'itérations exécutées par le processus maître, mais nos tests nous indiquent que cela ne vaut pas l'effort additionnel de temps de calcul. Une phase de probing plus courte semble la meilleure approche pour empêcher que les procédures ne deviennent trop insensibles.

La variante consistant à mettre à jour les itérations du maître en fonction de la profondeur du probing évite la phase pénible de remise à jour de presque tous les paramètres de la recherche tabou et semble de toute façon donner de meilleurs résultats que la première variante. Conséquemment, pour les résultats du probing présentés dans ce travail nous avons utilisé la seconde variante où une itération du maître égale une itération d'un esclave, avec une profondeur de probing de deux itérations. L'analyse des sources de dégradation des performances et du degré de parallélisme associé à cette stratégie de parallélisation est la même que la stratégie 1-RS SPSS. Cependant, le surcroît total de temps associé à la parallélisation avec la stratégie de probing est relatif au nombre d'itérations effectuées par le maître. Pour nos tests ce surcroît total correspond à la moitié de celui du 1-RS SPSS étant donné que la profondeur du probing est de deux, divisant par un même facteur le nombre total d'itérations effectuées par le processus maître.

4.8.3 Parallélisations par recherches multiples (p-RM)

Les meilleures performances parmi les procédures parallèles synchrones appartiennent à la stratégie de parallélisation avec recherches multiples. Les recherches multiples sont les seules parmi celles considérées au niveau du tabou synchrone

ne requérant aucun calibrage particulier. En effet, au moins une des procédures séquentielles du traitement parallèle avec recherches multiples exécute la stratégie d'exploration ayant donnée les meilleurs résultats pour le traitement séquentiel. Lorsque plusieurs stratégies d'exploration différentes sont nécessaires comme c'est le cas pour SPDS et MPDS, des stratégies très proches de la meilleure stratégie séquentielle ont été utilisées comme stratégie de différentiation. En l'absence de communication entre les procédures séquentielles du traitement parallèle avec recherches multiples, la logique d'exploration du domaine définie pour chaque procédure séquentielle est la même que lors de l'exécution dans le traitement séquentiel. Ceci nous assure pour les recherches multiples d'une qualité des résultats au moins aussi bonne que ceux du traitement séquentiel, puisque la meilleure stratégie séquentielle est toujours exécutée par une des procédures séquentielles du traitement parallèle par recherches multiples.

Nous avons observé qu'une plus grande différentiation de la recherche donne de meilleurs résultats: l'approche SPDS est meilleure que l'approche MPSS, mais l'approche SPDS est dépassée largement par l'approche MPDS. En fait, plus la différentiation est grande, plus l'exploration de l'espace de solutions risque d'être large. C'est ce qui expliquerait pourquoi les performances des stratégies MPDS et SPDS s'améliorent avec le nombre de processeurs, tandis que ce facteur n'a pas d'impact significatif sur les performances de la stratégie MPSS.

En théorie, le degré de concurrence est pratiquement illimité pour ce type de stratégies de parallélisation, en pratique des facteurs comme la redondance des chemins d'exploration (la même solution étant explorée par plus d'une procédure) et la topologie de l'espace de solutions limitent le nombre de procédures qui peuvent être exécutées en parallèle de manière efficace. Pour ces stratégies de parallélisation basées sur une décomposition fonctionnelle du traitement spéculatif, le nombre de solutions qui sont explorées plus d'une fois tend à s'accroître avec le nombre de

procédures pour une taille de problème donnée. En effet, il devient difficile de maintenir divergents les uns par rapport aux autres des chemins de recherche lorsque le degré de différentiation des stratégies est moindre suite à l'accroissement du nombre de stratégies d'exploration ou encore parce que l'espace de solutions recèle peu d'optima locaux.

Il n'y a qu'une seule source d'importance de dégradation des performances pour les stratégies par recherches multiples. La méthode tabou que nous avons conçue pour le problème de localisation est telle que le temps d'exécution d'une itération varie selon les phases de la méthode. Par exemple, une itération de swap prend plus de temps d'exécution qu'une itération de add/drop. Or, dépendant de la solution initiale et de la stratégie d'exploration, la proportion d'itérations exécutées dans chaque phase pourra varier, la procédure ayant exécutée le plus d'itérations de swaps aura le temps d'exécution le plus long. Lorsqu'on compare l'exécution d'une procédure parallèle p-RM avec la procédure de la méthode tabou séquentielle, il se peut qu'au moins une des procédures de la stratégie p-RM exécute plus d'itérations dans la phase de swap que l'implantation séquentielle. En fait, le temps parallèle T_p d'exécution de la procédure p-RM sera égale à celui de la procédure ayant exécutée le plus d'itérations dans des phases de swaps. Lorsque ce temps est supérieur au temps de la méthode séquentielle T_s, on pourra considérer la différence $T_p - T_s$ comme étant une dégradation des performances de la procédure p-RM lorsque le même nombre d'itérations est exécuté par les deux types de procédures. Enfin, il y a une phase d'échange d'information et quelques opérations supplémentaires exécutées par une procédure parallèle avec recherches multiples pour choisir la meilleure solution. Cependant cette phase ne se produit qu'une seule fois dans le traitement, les messages sont envoyés en parallèle (startup = t_s) et l'information transmise n'implique que quelques octets, on considère donc que ces sources de dégradation sont non significatives dans le temps total T_p de la procédure p-RM.

4.8.4 Procédures p-chemins et partage synchrone

Deux problématiques ont été étudiées en détail relativement aux trois stratégies de p-chemins et partage synchrone p-KS: la fréquence des points d'interaction et l'utilisation de l'information partagée.

Les tests préliminaires que nous avons effectués indiquent que les performances des stratégies p-KS tendent à s'améliorer lorsque les procédures interagissent plus souvent. Par contre, des points d'interaction trop fréquents peuvent changer la nature même de la méthode tabou conçue originalement. Par exemple, si δ (le nombre d'itérations entre deux points d'interaction) est inférieur au nombre d'itérations add/drop sans amélioration (ce paramètre peut signaler une phase d'intensification), la procédure n'exécutera jamais de phases d'intensification ou de diversification. Évidemment le temps parallèle des procédures p-KS s'accroît avec le nombre de points d'interaction dû au temps d'attente pour la synchronisation et au coût de transfert des données sur le réseau associées à chaque point d'interaction. En tenant compte de ces facteurs et étant donné le type et la dimension des problèmes que nous avons utilisés pour nos tests, nous avons fixé à $\delta = 25$, les résultats rapportés dans les tableaux 4.4, 4.5 et 4.6 correspondent à cette valeur de δ.

Concernant la problématique de l'utilisation de l'information partagée entre les processus aux points d'interaction, nous avons testé deux façons d'utiliser cette information. Dans les deux variantes, les solutions améliorantes visitées par toutes les procédures durant chaque étape sont triées, et les p meilleures sont distribuées parmi les p procédures du traitement parallèle. La première variante implante une forme simple de partage de l'information: chaque procédure accepte la solution partagée qui lui est soumise comme étant une nouvelle solution initiale, réinitialisant toutes les mémoires et listes tabous avant d'entrer dans la prochaine étape de traitement. Nous identifions cette variante (semblable à celle utilisée dans [67]) par *partage simple*.

Définition 4.6 *Nous identifierons par* **meilleure solution à moyen terme** *la*

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	p-RM SPDS						
P1	0	98	0	17	0	17	0
P2	0.07 (0.16)	86	0.07 (0.16)	86	0.07 (0.16)	86	0.42 (1.48)
P3	0.01	158	0.01	158	0 (0.32)	271	0.01 (0.8)
P4	0.74 (0.81)	225	0.27 (0.67)	105	0.05 (0.73)	185	0.90 (1.1)
P5	0.77 (1.45)	147	0.002 (0.12)	296	0.002 (0.12)	296	0.77 (1.45)
P6	0.28 (1.89)	118	0 (0.63)	71	0 (0.63)	71	0.42 (3.67)
P7	0	138	0	138	0	88	0.07
P8	1.68	50	0	255	0	255	2.15
P9	0	82	0	82	0	82	0
P10	1.23 (1.27)	25	1.02	264	0.60	208	1.49
P11	0.61	55	0.61	38	0	203	1.52
P12	0.11	17	0.11	17	0.11	17	0.11
P13	0 (0.12)	161	0 (0.12)	161	0 (0.12)	161	0.002 (0.07)
P14	0.02 (0.59)	299	0.02 (0.56)	299	0.02 (0.56)	299	0.02 (0.59)
P15	0.26 (0.29)	274	0.26 (0.29)	274	0.26 (0.29)	274	1.23 (1.54)
P16	0.52	278	0.52	278	0.23	291	0.89 (9.21)
	p-KS SPDS						
P1	0	273	0	116	0	116	0
P2	0.14 (0.66)	299	0.14 (0.54)	273	0.14	249	0.42 (1.48)
P3	0.57	221	0.57	221	0.01	220	0.01 (0.8)
P4	0.21 (0.55)	249	0.21 (0.79)	244	0.12 (0.41)	294	0.90 (1.1)
P5	0.04 (1.25)	204	0.997 (1.00)	227	0.34 (0.49)	254	0.77 (1.45)
P6	0.05 (0.66)	228	0	190	0	115	0.42 (3.67)
P7	0.07	240	0.07	240	0.07	240	0.07
P8	0	226	0	165	0	165	2.15
P9	0	82	0	82	0	82	0
P10	2.27	28	2.27	28	1.49	161	1.49
P11	2.22	185	2.22	185	2.22	185	1.52
P12	0	220	0	169	0	88	0.11
P13	0 (0.02)	249	0 (0.02)	249	0 (0.02)	249	0.002 (0.07)
P14	0.02 (0.13)	224	0.02 (0.10)	282	0.02 (0.05)	175	0.02 (0.59)
P15	1.23	278	0	126	0.29	286	1.23 (1.54)
P16	0.02	275	0.02 (0.10)	282	0.02	140	0.89 (9.21)

Tableau 4.4 p processus de recherche - Résultats de SPDS

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	p-RM MPSS						
P1	0	18	0	18	0	18	0
P2	0.14 (1.18)	105	0.14 (1.18)	105	0.14 (0.42)	78	0.42 (1.48)
P3	0	103	0	103	0	103	0.01 (0.8)
P4	0.08 (1.07)	192	0.08 (1.07)	192	0.08 (1.07)	192	0.90 (1.1)
P5	0.34 (0.98)	217	0.04 (1.54)	146	0.03 (1.91)	222	0.77 (1.45)
P6	0 (2.20)	156	0 (2.20)	156	0 (2.20)	156	0.42 (3.67)
P7	0.07	294	0	263	0	263	0.07
P8	0	215	0	215	0	98	2.15
P9	0	218	0	123	0	123	0
P10	0.37 (1.98)	180	0.37 (1.98)	180	0	118	1.49
P11	0	246	0	141	0	79	1.52
P12	0	264	0	264	0	17	0.11
P13	0.002 (0.10)	299	0.002 (0.10)	299	0 (0.03)	299	0.002 (0.07)
P14	0 (3.42)	204	0 (3.42)	204	0 (3.42)	204	0.02 (0.59)
P15	1.23 (1.35)	245	1.23 (1.35)	245	1.23 (1.35)	245	1.23 (1.54)
P16	0.89 (3.62)	268	0.89 (3.62)	268	0.89 (3.62)	268	0.89 (9.21)
	p-KS MPSS						
P1	0	65	0	65	0	65	0
P2	0.79 (1.05)	223	0.5 (1.49)	293	0.64 (1.76)	273	0.42 (1.48)
P3	0.01	275	0.01	182	0.01	112	0.01 (0.8)
P4	0.33 (0.59)	146	0.42 (1.97)	297	0.14 (1.57)	299	0.90 (1.1)
P5	0.38 (0.58)	127	0.38 (0.59)	233	0.38	232	0.77 (1.45)
P6	0 (0.43)	158	0 (0.56)	291	0 (1.09)	173	0.42 (3.67)
P7	0	52	0	139	0	274	0.07
P8	0	290	0	231	0.85 (4.15)	272	2.15
P9	0	289	0	296	0	174	0
P10	2.04	289	1.13 (46.09)	70	1.42 (5.51)	299	1.49
P11	0.61	271	0.61	205	0.61 (17.62)	36	1.52
P12	0	220	0.67 (12.05)	268	0	199	0.11
P13	0.007 (0.04)	275	0.01 (0.04)	273	0.007 (0.02)	271	0.002 (0.07)
P14	0.04 (0.24)	295	0 (0.26)	239	0 (0.02)	299	0.02 (0.59)
P15	0	299	0	233	0	226	1.23 (1.54)
P16	0.44	298	0.25	277	0.02	244	0.89 (9.21)

Tableau 4.5 p processus de recherche – Résultats de MPSS

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	p-RM MPDS						
P1	0	21	0	21	0	18	0
P2	0.21 (1.87)	165	0.21 (0.54)	179	0.14 (1.57)	120	0.42 (1.48)
P3	0 (0.83)	89	0	236	0	178	0.01 (0.8)
P4	0.35 (1.04)	250	0.32 (1.28)	213	0.12 (1.69)	235	0.90 (1.1)
P5	0.38 (0.58)	37	0 (1.04)	54	0 (1.04)	54	0.77 (1.45)
P6	0 (1.57)	121	0 (1.15)	290	0 (1.15)	290	0.42 (3.67)
P7	0.07	182	0	264	0	264	0.07
P8	0	232	0	232	0	127	2.15
P9	0	259	0	76	0	76	0
P10	0.37	91	0.37	91	0.37	91	1.49
P11	0	121	0	121	0	121	1.52
P12	0.11 (2.27)	202	0 (0.30)	170	0 (0.30)	170	0.11
P13	0.002 (0.02)	22	0 (0.12)	71	0 (0.12)	71	0.002 (0.07)
P14	0 (0.48)	117	0 (0.48)	117	0 (0.48)	117	0.02 (0.59)
P15	0 (0.34)	133	0 (0.34)	133	0 (0.34)	133	1.23 (1.54)
P16	0.89 (3.62)	102	0.02 (11.22)	187	0.02	150	0.89 (9.21)
	p-KS MPDS						
P1	0	80	0	265	0	113	0
P2	0.23 (1.52)	284	0.33 (1.21)	197	0.21 (1.02)	287	0.42 (1.48)
P3	0	177	0.01	252	0 (0.32)	229	0.01 (0.8)
P4	0.38 (1.23)	279	0.38 (1.50)	299	0.38 (1.60)	269	0.90 (1.1)
P5	0.87 (1.00)	184	0 (.04)	255	0 (0.78)	226	0.77 (1.45)
P6	0.05 (0.48)	228	0 (.60)	95	0 (0.62)	140	0.42 (3.67)
P7	0.07	126	0	240	0.66	7	0.07
P8	1.32	79	2.15	236	1.32 (6.47)	177	2.15
P9	0.82	201	0.82	77	0.82	140	0
P10	2.22 (3.37)	151	1.08 (2.37)	151	1.45	160	1.49
P11	0	286	0 (3.14)	290	0.62	128	1.52
P12	1.25	299	0	254	0 (0.30)	240	0.11
P13	0.001 (0.01)	299	0 (0.04)	205	0 (0.08)	263	0.002 (0.07)
P14	0.06 (0.3)	225	0.05 (0.26)	299	0.02 (0.18)	284	0.02 (0.59)
P15	1.23 (1.35)	276	0	234	0	254	1.23 (1.54)
P16	0.44	263	0.44	227	0	239	0.89 (9.21)

Tableau 4.6 p processus de recherche – Résultats de MPDS

meilleure solution de la boucle externe d'une procédure séquentielle (voir section 4.5).

Pour la deuxième variante que l'on identifie par *partage conditionnelle*, chaque procédure teste d'abord la solution partagée qui lui est soumise en la comparant avec la meilleure solution à moyen terme et accepte la nouvelle solution seulement si elle est meilleure. Notons que dans ce cas, les listes tabous à court terme sont réinitialisées alors que les mémoires à long terme ne le sont pas. Les résultats des tests ont été concluants: on a pu observer que la stratégie de partage conditionnelle donne constamment soit des résultats similaires ou meilleurs que la variante de partage simple.

Les conclusions portant sur les limitations pratiques du degré réel de concurrence des stratégies de parallélisation avec recherches multiples p-RM s'appliquent également aux stratégies de parallélisation p-KS. Le fait que les procédures séquentielles des stratégies parallèles p-KS partagent leurs connaissances sur l'espace de solutions ne change en rien ces conclusions. Par contre il y a deux facteurs au niveau du partage synchrone de la connaissance qui font que le temps parallèle $T_p(p-KS)$ de l'approche p-KS ne sera pas le même que le temps parallèle $T_p(p-RM)$ des approches p-RM, étant donné un même ensemble initial de stratégies de recherche et un même nombre d'itérations.

Le premier facteur est relatif à l'impact du partage des historiques de recherche sur l'évolution du chemin de chaque procédure. À chaque point d'interaction, les procédures disposent, en plus de leur historique de recherche locale, de toute l'information accumulée par l'historique de recherche des autres procédures lors de la dernière étape de traitement. Comme l'information accumulée joue un rôle important pour la méthode tabou, tout changement au niveau de ce paramètre a un impact sur l'évolution que prend le chemin de recherche d'une procédure. En changeant le chemin de recherche des procédures par le partage des historiques de recherche, la proportion de swaps et de add/drop n'est plus la même au niveau de chaque

procédure et conséquemment le temps parallèle $T_p(p-KS)$ risque de changer.

Le deuxième facteur est responsable d'un accroissement substantiel du temps d'exécution des procédures p-KS par rapport aux procédures p-RM. Ici, une analyse similaire à celle faite pour les sources de dégradation des performances de chaque procédure séquentielle des stratégies p-RM s'applique "pour chaque étape" (entre deux points d'interaction) des stratégies de type p-KS. En effet, bien que chaque étape de chacune des procédures exécute un nombre égal de δ itérations entre deux points d'interaction, les phases de la méthode tabou qui sont exécutées d'une procédure à l'autre ne sont pas nécessairement les mêmes. Cette observation implique que l'intervalle de temps entre deux points d'interaction n'est pas le même entre les différentes procédures. Soit t_{ij} le temps écoulé entre les deux points d'interaction de la procédure i à l'étape j. Pour chaque étape j le temps parallèle écoulé $T_p(p-KS)^j$ entre deux points d'interaction sera borné supérieurement par la procédure exécutant le plus grand nombre d'itérations de swap, c'est-à-dire $T_p(p-KS)^j = \max(t_{ij})\ i = 1,\ldots,p$ où p est le nombre de procédures de l'exécution parallèle p-KS. Sur l'ensemble du traitement, la borne inférieure du temps écoulé correspond à la somme des temps écoulés à chaque étape par la procédure ayant exécutée le plus de swaps, c'est-à-dire $T_p(p-KS) = \sum_{j=1}^{j=\frac{t}{k}} \max(t_{ij})$ où t est le critère d'arrêt de chaque procédure. Or cette sommation correspond à un nombre de swaps supérieur ou égal à celui de la procédure ayant exécuté le plus de swaps entre le début et la fin du traitement. C'est ce qui fait que que la dégradation des performances d'une stratégie p-KS est au moins égale mais généralement supérieure (et de beaucoup) à celle des recherches multiples. Ce temps de "synchronisation" est le coût qu'il faut payer à chaque étape pour obtenir un partage synchrone de la connaissance. En comparaison, la dégradation attribuable au temps de communication est insignifiante puisqu'on peut le gérer de manière efficace.

4.8.5 Comparaison entre les différentes approches synchrones

Quelques conclusions générales émergent dans l'ensemble des approches synchrones:

- Toutes les versions parallèles sont plus performantes, et la plupart de manière significative, que la procédure séquentielle. En fait, sur tous les tests, les implantations parallèles ont un gap moyen de 0.26% (avec un intervalle allant de 0.04% à 0.56%) comparé à la moyenne séquentielle de 0.63%. Ainsi, pour une allocation donnée de temps de calcul, la parallélisation permet d'obtenir de meilleures solutions.

- Les procédures parallèles de recherche tabou aident à atteindre des solutions de hautes qualités. En effet, sauf pour quelques exceptions, soit que la solution optimale fut trouvée ou qu'un gap très mince existe entre la solution optimale et celle trouvée par la recherche tabou parallèle.

- L'accroissement du nombre de processeurs semble améliorer les performances, mais jusqu'à un certain point seulement.

- La fréquence d'échange d'information entre les processus et la façon dont cette information est traitée peut affecter de manière significative les performances d'une recherche parallèle tabou synchrone.

L'évolution du gap moyen pour les différents tests est illustrée dans la figure 4.4. Il est clair que l'implantation parallèle améliore les performances de la méthode tabou. D'un autre côté, on observe que, pour un nombre constant d'itérations, accroître le nombre de processeurs améliore les performances de la recherche tabou, mais jusqu'à un certain point. En fait, les améliorations sont significatives lorsqu'on passe de 4 à 8 processeurs, mais la courbe tend à devenir horizontale lorsqu'on passe à 16 processeurs. De ce point de vue, la recherche tabou synchrone semble se comporter comme beaucoup d'autre méthodes d'optimisation dans un environnement parallèle. De plus, des tests additionnels faits sur un sous-ensemble de problèmes

suggèrent qu'au delà d'un certain seuil (certainement relié à la taille du problème: $p = 8$, dans le cas des exemples de problèmes considérés dans ce travail) il est plus profitable d'accroître le nombre d'itérations de la recherche que le nombre de processeurs. On peut expliquer ce phénomène par une combinaison de deux facteurs. Premièrement, lorsque le nombre de processeurs s'accroît, plusieurs d'entre eux explorerons des régions non intéressantes de l'espace de solutions dû, soit à de mauvaises solutions initiales, soit à des stratégies d'exploration moins efficaces, ou à une combinaison des deux. Deuxièmement, en accroissant le nombre d'itérations, on permet aux fonctions de la mémoire à long terme de la recherche tabou d'être utilisées pleinement, et de guider correctement la recherche.

Il semble que, même si l'échange d'information concernant les meilleures solutions trouvées lors de la recherche soit bénéfique comparé au traitement séquentiel, l'injection à tout moment d'une solution partagée dans une trajectoire de recherche tabou peut avoir un effet négatif sur les performances. En fait, si on respecte la logique de la recherche tabou, les solutions partagées devraient être utilisées par le processus récepteur qu'à des phases bien particulières de la recherche tabou, la diversification apparaissant comme l'une des phases les plus appropriées. Conséquemment, soit que l'acceptation d'une solution partagée est reportée jusqu'à ce que des conditions précises soient rencontrées, ou soit que l'on modifie le critère qui définit une étape (par exemple en ayant les points d'interaction à l'arrivée d'une phase de diversification). Il faut noter également que pour exploiter pleinement les approches de parallélisation p-KS, la procédure de recherche tabou devrait être calibrée de nouveau. Ce n'est pas cependant l'objectif de la présente étude qui vise uniquement à évaluer les procédures en fonction d'un ensemble "objectif" de paramètres. Dans ce contexte, si une stratégie de parallélisation synchrone est envisagée, il semble que l'approche simple consistant à exécuter plusieurs procédures différentes de recherche à partir de solutions initiales différentes puisse offrir les meilleures chances de succès.

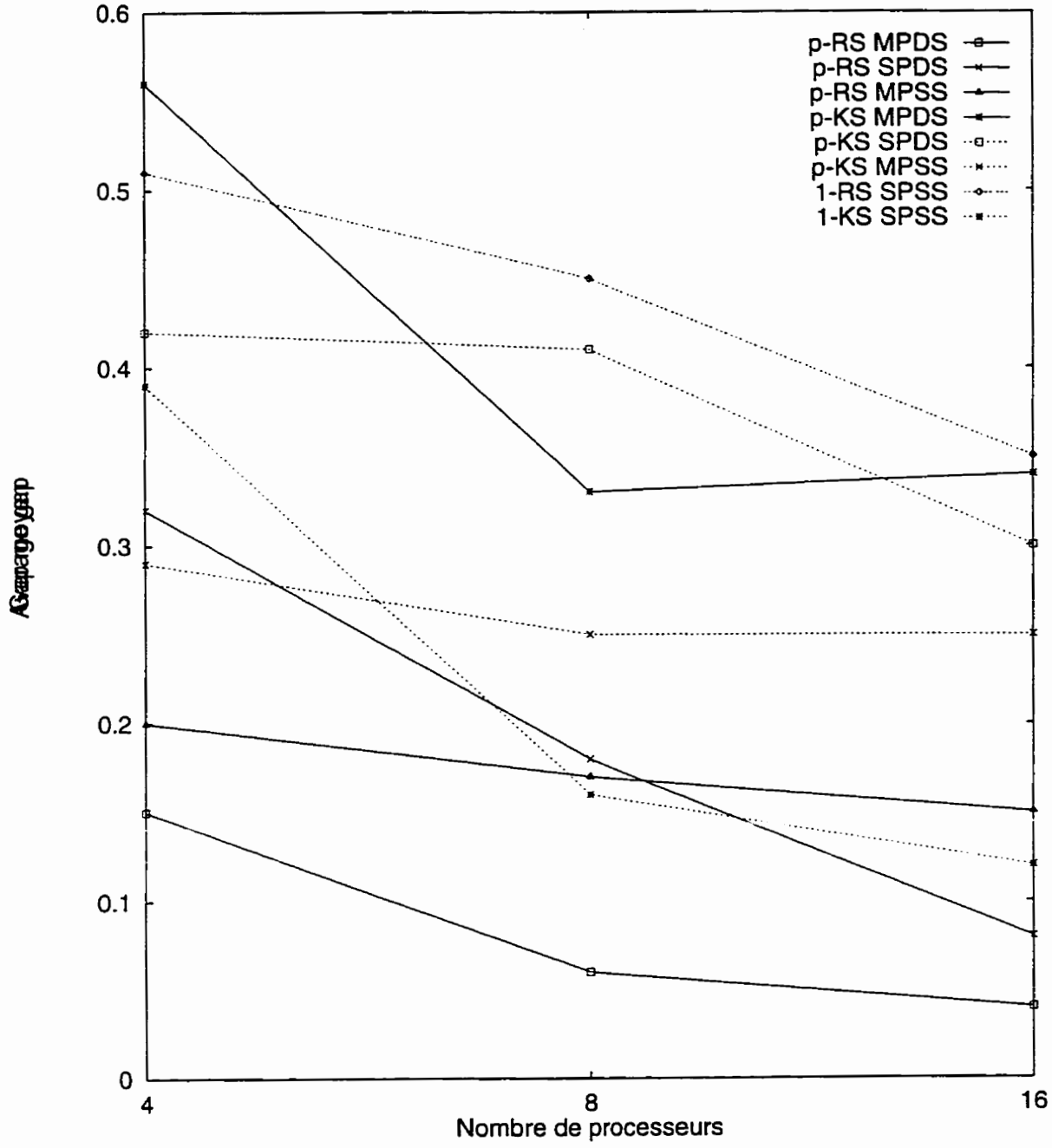


Figure 4.4 Comparaison des gaps

Relativement au temps de calcul, la mesure de performance la plus adéquate pour les stratégies synchrones est certainement le temps "écoulé" pour compléter une exécution. Cependant, dans un environnement de traitement partagé comme le nôtre où il est difficile d'isoler complètement une application d'interactions avec d'autres applications, il est hasardeux de faire des comparaisons au niveau du temps de traitement uniquement sur la base de problèmes individuels. Pour cette raison, les discussions sur le temps écoulé portent sur des temps moyens observés pour les différentes stratégies sur l'ensemble complet des tests.

Comme prévu dès le départ, le temps d'exécution s'accroît de manière significative avec le niveau d'échange d'information requis pour les différentes stratégies. Les approches avec recherches multiples, où les communications ne sont faites qu'à la fin du traitement, s'exécutent plus rapidement que les méthodes p-KS où les points d'interaction s'effectuent à chaque 25 itérations. Le temps écoulé des procédures parallèles avec recherches multiples est similaire à celui du temps d'exécution de la plus longue procédure séquentielle impliquée dans une procédure avec recherches multiples. Par contre, les approches p-KS demandent environ la moitié du temps requis par les approches 1-RS, pour ces dernières le processus maître doit interagir avec les processus esclaves à chaque itération contrairement à des séquences de 25 itérations pour les stratégies p-KS. Le temps de calcul s'accroît également en fonction du nombre de processeurs, puisque le temps pour la synchronisation aux points d'interaction dépend du processus le plus long et il y a plus de messages qui doivent être échangés lorsque plus de processeurs sont impliqués dans le traitement. Ceci est particulièrement significatif dans le cas des stratégies p-KS où les temps de calcul pour $p = 16$ sont environ 20% supérieur à ceux de $p = 4$.

4.9 Procédures asynchrones de recherche tabou

La présente section a pour objectif d'illustrer la taxonomie par le biais des procédures parallèles asynchrones de la méthode tabou. Nous dirons qu'une procédure parallèle

est asynchrone si le partage de l'information contenue dans l'historique de recherche se fait à partir de la logique interne de chaque procédure sans coordination avec les autres procédures du traitement parallèle. Ce mode de partage de l'information fait que les procédures parallèles asynchrones de la recherche tabou possèdent des caractéristiques particulières au niveau des sources de dégradation des performances, du modèle algorithmique, de la cardinalité de la recherche et du parcours de l'espace de solutions.

En effet, le taux de dégradation des performances dû à l'échange d'information est minime comparativement à celui des procédures synchrones parce que les points d'interaction se font par le biais d'une mémoire centrale, ce qui élimine la dégradation des performances due à la synchronisation des procédures entre elles. Cependant, comme c'est le cas en général pour les algorithmes chaotiques, ce partage asynchrone de l'information provoque un comportement non déterministe de la procédure parallèle. D'une exécution à l'autre, l'exploration de l'espace de solutions n'est pas la même parce que des événements reliés au système physique ou à l'activité d'autres applications qui partagent les mêmes processeurs font varier d'une exécution à l'autre les accès aux variables partagées par les procédures. Une telle variation ne serait-ce qu'infime, affecte l'échange d'information et finalement les décisions qui dépendent fortement de l'historique de la recherche sont affectées à leur tour, il y a alors un effet d'entraînement fort complexe qui se produit.

Le partage asynchrone de l'information et le non-déterminisme qu'il entraîne signifie qu'il n'est pas possible de garantir que la sémantique de la procédure parallèle asynchrone sera la même que la procédure séquentielle. En effet, pour les procédures parallèles synchrones de la recherche tabou (c.à.d. la plupart des procédures proposées dans la littérature [21, 23, 29, 67, 92], etc.) on peut toujours reproduire par une procédure séquentielle unique, le comportement (l'exploration effectuée de l'espace de solution) adopté par la procédure parallèle. Au niveau des stratégies 1-chemin, il s'agit que la procédure parallèle et la procédure séquentielle utilisent

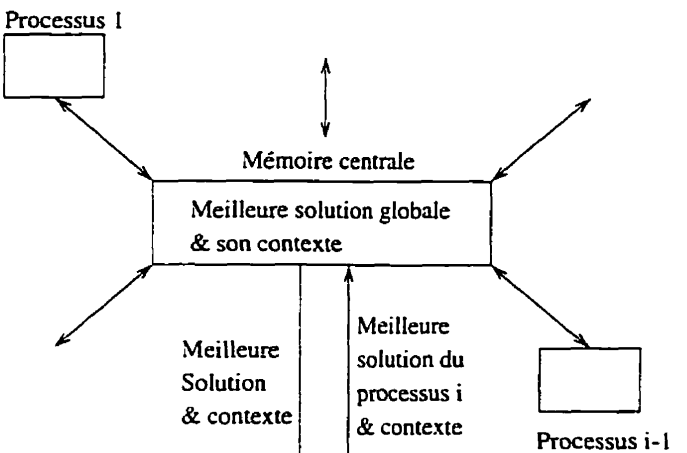
les mêmes listes candidates à chaque itération; pour les stratégies p-chemins, les p procédures et leurs étapes de synchronisation peuvent être simulées par une seule procédure séquentielle qui exécute toutes les procédures d'une étape avant de passer à l'étape suivante. Ce n'est pas nécessairement le cas pour les approches asynchrones où on ne peut que simuler par un processus stochastique l'impact du système physique et des autres applications sur la procédure tabou parallèle, il n'est donc pas possible de reproduire exactement le comportement de la procédure parallèle par une procédure séquentielle.

Le choix du voisin de la procédure séquentielle de recherche tabou que nous avons conçue porte sur celui qui minimise la valeur de la fonction objectif. C'est ce critère que nous avons appliqué aux procédures parallèles synchrones 1-chemin. L'équivalent asynchrone de ces procédures parallèles pose un problème puisqu'on ne peut plus garantir (si l'on veut être asynchrone) que le choix du voisin se portera sur celui qui minimise la fonction objectif (c'est un problème similaire à celui des stratégies de parallélisation "par erreurs" du recuit simulé). Il nous faut plutôt utiliser un critère comme "choisir la première solution améliorante lorsqu'une telle solution existe". Nous n'avons donc pas réalisé d'implantations 1-chemin asynchrones parce que cela nous aurait entraîné trop loin de notre objectif principal qui est d'utiliser les mêmes mécanismes de recherche à travers les différentes procédures pour pouvoir comparer les stratégies de parallélisation entre elles et avec la procédure séquentielle.

Enfin, comme nous le savons déjà, le caractère asynchrone de l'accès à l'information fait que chaque processus travaille avec des connaissances relatives à une sous-région de l'espace de solutions. En ce sens, l'approche asynchrone se distingue du cas p-chemins partage synchrone où l'information est globale en provenance de toutes les procédures, et confère à la parallélisation asynchrone une manière de parcourir l'espace de solutions et un mode d'acquisition de l'information tout à fait différent de celui des approches p-chemins synchrones.

4.9.1 Stratégies de parallélisation p-chemins collégiales asynchrones

Le cadre général des stratégies de parallélisation p-chemins collégiales asynchrones (p-C) de la méthode tabou est illustré dans la figure 4.5.



	$x_i = x_j \forall i,j$	$x_i \neq x_j \forall i,j$
$y_i = y_j \forall i,j$	SPSS	MPSS
$y_i \neq y_j \forall i,j$	SPDS	MPDS

Figure 4.5 Cadre du traitement parallèle asynchrone

Il s'agit de p procédures séquentielles exécutant chacune une recherche tabou. Pour chaque champs de l'historique de recherche qui est partagé, une variable globale est créée, cette variable est accédée en fonction des critères du partage de l'information de la procédure parallèle de la recherche tabou. Dans le cas d'une

implantation sur un système distribué (c'est le cas de nos implantations), nous utilisons une architecture de type blackboard où un des processeurs est responsable d'emmagasiner la valeur des champs partagés de l'historique de recherche.

Pour toutes les procédures parallèles de la présente section, les deux champs suivants de l'historique de recherche seront partagés: la meilleure solution d'une procédure et son contexte (c'est-à-dire le vecteur des variables de décisions du problème d'optimisation). Les critères d'accès et de mise à jour de la mémoire centrale sont les suivants:

- Lorsque la solution courante d'une procédure est meilleure que toutes les solutions explorées auparavant par cette procédure, celle-ci envoie sa solution courante et le contexte qui lui est associée à la mémoire centrale.
- La mémoire centrale est mise à jour si la meilleure solution courante envoyée par une procédure est meilleure que celle déjà en mémoire centrale.
- À chaque fois qu'une procédure accède la mémoire centrale pour une mise à jour, la mémoire centrale retourne à cette procédure son contenu (même si la mise à jour a échouée).
- Après un certain nombre d'itérations sans améliorer sa meilleure solution courante, une procédure demande une copie de la solution contenue en mémoire centrale.

Suite à un point d'interaction avec la mémoire centrale, un processus reprend toujours sa nouvelle étape d'exécution à partir de la configuration (mémoires et listes tabous à court terme, la meilleure solution de la recherche locale, le contexte de la solution courante, etc.) de l'étape précédente. Cependant, avant de commencer une phase de diversification, le processus compare et éventuellement remplace sa meilleure solution et son contexte par la meilleure solution de la mémoire centrale. Alors, soit que la recherche recommence à partir de la meilleure solution de la

mémoire centrale (ce qui correspond à une diversification imposée de l'extérieur), ou une diversification ordinaire est exécutée. De cette manière, on peut réconcilier le comportement de la recherche tabou basé sur les mémoires à long terme de chaque processus à l'importation d'informations exogènes.

Contrairement à ce qui se passe en mode synchrone, la même information est partagée uniquement par les procédures ayant accédées à la mémoire centrale entre deux mises à jour de celle-ci, ce qui fait qu'en théorie du moins, à tout instant de la recherche, l'information influençant les chemins de recherche de la procédure parallèle asynchrone est plus diversifiée que dans le cas synchrone.

Nous avons également testé une version différente de la mémoire centrale où plusieurs copies des champs de l'historique de recherche partagée sont créées. L'objectif de cette modification est d'obtenir un pool des meilleures solutions en mémoire centrale au lieu de n'en garder qu'une seule. On a aussi introduit un certain degré de non-déterminisme au niveau de l'information qui est retournée aux procédures à partir de la mémoire centrale. Les deux principales différences entre cette stratégie de *pool de solutions* et la précédente sont:

- La mémoire centrale garde et met à jour une liste des s meilleures solutions (et le contexte qui leur est associé) trouvées par les p procédures de recherche; la meilleure solution de la mémoire centrale de toute évidence appartient à cette liste.

- Toute solution retournée à une procédure à partir de la mémoire centrale est sélectionnée sur la base d'un choix aléatoire parmi les s solutions dans la liste.

Notons que dans le cadre d'un pool de solutions, une procédure peut recevoir plusieurs fois la même solution. En conséquence, lorsqu'une procédure reçoit une solution de la mémoire centrale, elle accepte cette solution seulement si elle est différente des solutions précédentes reçues de la mémoire centrale. Enfin, aucune solution non réalisable n'est échangée entre les procédures et la mémoire centrale pour les deux versions que nous avons implantées.

Pour ces deux types de mémoires centrales, nous avons conçu des procédures parallèles p-chemins collégiales asynchrones en combinaison avec les stratégies de différentiation de la troisième dimension. Comme pour le cas synchrone et pour les mêmes raisons, nous n'avons pas implanté de procédures parallèles p-chemins collégiales asynchrones SPSS. Nous avons donc conçu des procédures parallèles pour les stratégies de différentiation SPDS, MPSS et MPDS.

Il est évident que dans le cadre d'une procédure parallèle collégiale asynchrone tel qu'illustré ci-haut, à tout moment, une procédure dispose au mieux de l'information reliée à son propre historique de recherche, la valeur et le contexte d'une solution globale qui n'est pas nécessairement à jour, sans aucune indication de l'évolution globale de la recherche parallèle. En ce sens, nous n'obtenons pas une image globale de l'effet combiné des procédures de recherche et nous perdons, du moins partiellement, l'effet d'apprentissage et les mécanismes de mémoire centrale à la méthode tabou. Les stratégies que nous identifions par p-chemins réflexifs asynchrones (p-KC) ont pour objectif de s'aborder cette problématique.

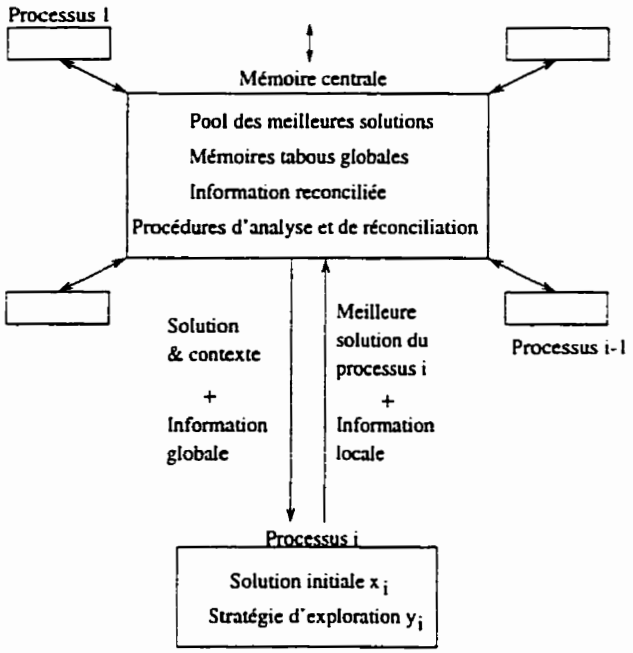
4.9.2 Stratégies de p-chemins réflexifs asynchrones

L'usage principal de l'échange d'information pour les procédures parallèles que nous avons décrit jusqu'à maintenant consiste, soit à détourner les procédures séquentielles de régions peu prometteuses de l'espace de solutions déjà explorées par d'autres procédures, ou au contraire à diriger l'exploration de certaines procédures séquentielles vers des régions plus prometteuses. On pourrait cependant faire un tout autre usage de cette information. En effet, on peut utiliser la diversité de l'information pour créer de nouvelles solutions et ainsi ouvrir de nouvelles régions de l'espace de solutions à l'exploration. On peut, comme pour le cas des algorithmes génétiques, utiliser l'information contenue au niveau de l'historique des procédures concurrentes pour créer graduellement une image de plus en plus précise du patron (pattern) de la solution optimale, utilisant ensuite cette image pour guider l'exploration des

procédures tabous concurrentes.

La catégorie des stratégies de parallélisation p-chemins réflexifs asynchrones regroupe les procédures parallèles asynchrones de la recherche tabou combinant la recherche tabou avec des méthodes d'analyse et de réconciliation des connaissances acquises par les procédures séquentielles. La fonction de ces méthodes est d'analyser les connaissances diverses accumulées et échangées entre les procédures séquentielles et d'implanter de nouveaux mécanismes de guidage du chemin de recherche de chaque processus en se basant sur cette analyse et les nouvelles connaissances qui en découlent.

La figure 4.6 illustre une version simple de la stratégie p-chemins réflexifs asynchrones.



	$x_i = x_j \forall i,j$	$x_i \neq x_j \forall i,j$
$y_i = y_j \forall i,j$	SPSS	MPSS
$y_i \neq y_j \forall i,j$	SPDS	MPDS

Figure 4.6 Stratégie p-chemins réflexifs asynchrones

Ce qui distingue cette figure de la précédente, c'est la plus grande quantité d'information qui est échangée entre les procédures et la mémoire centrale, et par conséquent la définition de nouvelles structures de données au niveau de la mémoire centrale. Dans cette figure, deux nouvelles structures de données sont définies, l'une pour l'emmagasinage de connaissances induites à partir de l'information analysée au niveau de la mémoire centrale par les procédures "d'analyse et de réconciliation" qui s'y trouvent et l'autre pour l'emmagasinage des informations utilisées par les

nouveaux mécanismes de guidage des procédures tabous concurrentes (dans le cas présent des listes tabous globales à toutes les procédures). Les différentes mémoires de la recherche tabou offrent beaucoup de potentiel pour obtenir, par une analyse comparative, de nouvelles connaissances sur l'espace de solutions. Les bases de comparaisons peuvent porter sur la distribution de certains attributs des solutions trouvées comme les optima locaux, la fréquence de changements d'états de certaines variables de décisions au niveau des bonnes ou mauvaises solutions trouvées par les processus, l'effet en moyenne du changement d'état d'une variable de décision sur la valeur de la fonction objectif, etc.

Nous avons conçu une procédure parallèle pour cette catégorie de stratégies de parallélisation où la mémoire centrale emmagasine l'information produite par la comparaison du contexte des solutions strictement améliorantes par rapport à la meilleure solution de la mémoire centrale (c'est-à-dire les solutions reçues qui forcent une mise à jour de la solution de la mémoire centrale). La nouvelle information emmagasinée enregistre la fréquence des changements de statut des variables de décisions à partir du contexte des solutions améliorantes de la mémoire centrale. Ainsi, à long terme, cette mémoire de *cohérence des solutions améliorantes* tend à faire apparaître deux catégories de variables de décisions: celles dont le statut ne change pas à travers les bonnes solutions rapportées à la mémoire centrale, c'est-à-dire les variables de décisions qui semblent définitivement fixées et celles dont le statut ne semble pas défini de ce point de vue. Cette mémoire de cohérence des solutions améliorantes sert à constituer une liste tabou globale qui est ajoutée par la mémoire centrale à chaque message visant à retourner une solution à une procédure de recherche. Sur réception, les procédures utilisent cette liste tabou globale pour influencer leur chemin de recherche par exemple en interdisant l'exploration de solutions qui impliqueraient le changement de statut d'une variable qui semble stable selon la mémoire de cohérence des solutions améliorantes. Au niveau de la présente

implantation, ceci est accompli via un mécanisme tabou similaire à celui de la diversification: le statut tabou des solutions candidates est modifié pendant la recherche locale selon les valeurs du vecteur.

4.9.3 Résultats expérimentaux des stratégies asynchrones

Les tests présentés dans cette section ne portent que sur les approches p-chemins collégiales asynchrones, nous reviendrons dans les chapitres suivants sur l'étude des stratégies de type réflexif asynchrone. Ces tests ont pour objectif d'explorer les alternatives de conception impliquées par les structures de traitement collégiales asynchrones de l'information décrites dans la section précédente, et d'examiner l'impact sur le comportement des procédures parallèles de quelques paramètres de conception tel que le nombre de processeurs, la taille du pool, etc. Ces résultats sont aussi utilisés pour déterminer quelle stratégie de parallélisation semble obtenir les meilleures performances, et comparer les approches asynchrones de parallélisation avec des approches plus classiques fondées sur la synchronisation des procédures.

Les tableaux 4.7, 4.8 et 4.9 résument les résultats obtenus pour les trois stratégies de différentiation, SPDS, MPSS, et MPDS, en utilisant un des deux mécanismes d'échange d'information: l'approche de la mémoire centrale avec une seule solution et l'approche avec le pool de solutions. Dans ce dernier cas, la taille s du pool correspond au nombre p de processeurs.

À partir des tests effectués sur les stratégies de parallélisation asynchrones de la méthode tabou, on peut tirer des conclusions générales assez similaires à celles que nous avions fait pour les approches synchrones de parallélisation. En effet, les versions parallèles asynchrones obtiennent généralement de meilleures solutions que la procédure séquentielle, et la plupart d'entre elles de manière significative. Par exemple, les gaps moyens de 0.17%, 0.12% et 0.03% pour l'approche SPDS (sans pool), avec respectivement 4, 8 et 16 processeurs se comparent avantageusement avec le gap moyen de 0.63% de la procédure séquentielle.

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	Mécanisme de base de la mémoire centrale						
P1	0	18	0	47	0	165	0
P2	0.54	286	0.14 (0.62)	290	0.20 (0.37)	299	0.42 (1.48)
P3	0.01	113	0.01	45	0	8	0.01 (0.8)
P4	0 (0.13)	290	0 (0.06)	299	0 (0.01)	290	0.9 (1.1)
P5	0.33	233	0.64	285	0	93	0.77 (1.45)
P6	0.01 (1.01)	297	0	278	0.01	241	0.42 (3.67)
P7	0	247	0	182	0	31	0.07
P8	0.85	137	0	185	0	228	2.15
P9	0	243	0	101	0	151	0
P10	0.3	242	0.17	15	0	158	1.49
P11	0	175	0	127	0	110	1.52
P12	0.11	13	0.11	127	0	10	0.11
P13	0.002 (0.12)	297	0.002 (0.04)	296	0 (0.03)	217	0.002 (0.07)
P14	0.02 (0.21)	266	0.04 (2.04)	223	0.02 (0.2)	172	0.02 (0.59)
P15	0.59 (0.71)	290	0.73 (1.06)	295	0.26 (0.6)	270	1.23 (1.54)
P16	0.23	226	0.89	241	0.14	250	0.89 (9.21)
	Stratégie avec pool $s = p$						
P1	0	109	0	125	0	18	0
P2	0.43 (0.67)	219	0.14 (0.95)	290	0.16 (0.29)	145	0.42 (1.48)
P3	0	301	0	198	0	270	0.01 (0.8)
P4	0.29	247	0.11 (0.75)	287	0.14 (0.79)	232	0.9 (1.1)
P5	1.0 (1.39)	141	0	238	0 (1.73)	287	0.77 (1.45)
P6	0.39 (1.16)	142	0 (0.63)	158	0 (0.91)	204	0.42 (3.67)
P7	0	190	0	242	0	94	0.07
P8	0	188	0	80	0	213	2.15
P9	0	194	0	253	0	130	0
P10	0.37	298	0.38	228	0.14 (3.08)	226	1.49
P11	0	268	0	135	0	233	1.52
P12	0.11	205	0.11	17	0	242	0.11
P13	0.002 (0.07)	261	0.002 (0.09)	298	0.002 (0.12)	75	0.002 (0.07)
P14	0.06 (0.76)	298	0.06 (0.18)	252	0 (1.12)	225	0.02 (0.59)
P15	0.19 (0.52)	240	0.73 (0.75)	287	1.22 (5.78)	293	1.23 (1.54)
P16	0.23	267	0.11	285	0.89 (9.1)	297	0.89 (9.21)

Tableau 4.7 Résultats p-chemins SPDS

Les procédures de recherche tabou parallèles asynchrones aident à trouver des solutions de très grande qualité. En effet, même avec un réglage simple des paramètres et sans exécution de la phase de post-optimisation, soit que la solution optimale est trouvée ou qu'un très faible gap existe entre la solution optimale et la solution trouvée par la procédure parallèle asynchrone. De plus, il faut noter la robustesse de l'approche parallèle asynchrone: toutes les procédures, généralement, trouvent de meilleures solutions que la version séquentielle, et habituellement, dans un nombre d'itérations moindre.

Avant de procéder plus en avant pour analyser les résultats et comparer les diverses stratégies, quelques remarques doivent être faites relativement aux implantations. La première concerne la définition des stratégies SPDS.

4.9.4 Stratégies collégiales asynchrones SPDS

Il est facile d'appliquer la stratégie SPDS dans le cadre synchrone où après chaque point d'interaction, toutes les procédures redémarrent en même temps et avec la même solution. Lorsque les implantations asynchrones sont envisagées, il n'est pas évident qu'un paradigme strict SPDS puisse être facilement appliqué, spécialement lorsqu'une stratégie de pool est utilisée. En fait, puisque les procédures ne s'attendent pas les unes les autres lorsqu'elles interagissent avec la mémoire centrale, on ne peut pas être assuré que toutes les procédures partagent une même solution initiale comme l'exige une différentiation de type SPDS. Pour remédier partiellement à ce problème, nous avons modifié la méthode de telle sorte que lorsqu'une solution améliorante en provenance de la mémoire centrale est acceptée par une procédure individuelle, elle remplace la meilleure solution locale immédiatement par cette solution de la mémoire centrale. Nous avons ainsi une sorte de cadre SPDS local pour un sous-ensemble de procédures qui communiquent avec la mémoire centrale dans une même fenêtre de temps à condition de ne considérer que des fenêtres de temps relativement brèves.

Une implantation plus stricte de la stratégie SPDS peut être obtenue si, par

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	Mécanisme de base de la mémoire centrale						
P1	0	20	0	20	0	19	0
P2	0.21 (0.23)	297	0.39	298	0.14 (0.55)	20	0.42 (1.48)
P3	0 (0.01)	298	0	145	0	191	0.01 (0.8)
P4	0.23 (0.30)	278	0 (0.06)	192	0.27 (0.41)	298	0.9 (1.1)
P5	0.33 (0.76)	288	0.33	274	0.33	277	0.77 (1.45)
P6	0.15 (0.19)	278	0.15	300	0.15 (0.19)	295	0.42 (3.67)
P7	0	184	0.07	28	0	49	0.07
P8	0	225	0	284	0	134	2.15
P9	0	146	0	235	0	178	0
P10	1.96	296	0.37	137	0.74	200	1.49
P11	0	117	0	45	0	47	1.52
P12	0.11	85	0	234	0	17	0.11
P13	0.002 (0.49)	255	0 (0.09)	299	0.002 (0.03)	250	0.002 (0.07)
P14	0 (1.5)	135	0 (1.5)	235	0.02 (0.21)	299	0.02 (0.59)
P15	0.59 (0.71)	275	0.26 (0.29)	290	0.59	299	1.23 (1.54)
P16	0.89	260	0.89	261	0.44	281	0.89 (9.21)
	Stratégie avec pool $s = p$						
P1	0	181	0	90	0	17	0
P2	0.44 (1.49)	210	0 (1.0)	299	0.16 (0.27)	264	0.42 (1.48)
P3	0 (0.32)	295	0	114	0	79	0.01 (0.8)
P4	0.5 (1.02)	300	0.35 (0.68)	219	0 (0.43)	258	0.9 (1.1)
P5	0.38 (0.58)	243	0	236	0	63	0.77 (1.45)
P6	0.15 (0.17)	288	0.05 (1.19)	275	0.01 (0.75)	295	0.42 (3.67)
P7	0	227	0	231	0	275	0.07
P8	0	93	0	234	0	85	2.15
P9	0	277	0	159	0	153	0
P10	1.16 (2.76)	202	0.14 (1.46)	257	0.30	83	1.49
P11	0	205	0	150	0	141	1.52
P12	0.11 (1.34)	227	0 (0.23)	295	0	17	0.11
P13	0 (0.48)	202	0.002 (0.44)	277	0.002 (0.49)	297	0.002 (0.07)
P14	0.06 (0.75)	201	0.04 (0.32)	183	0 (0.39)	128	0.02 (0.59)
P15	0 (0.07)	296	0	284	0.03 (0.54) (0.03)	296	1.23 (1.54)
P16	0.02 (3.08)	278	0.44 (3.37)	267	0.89	256	0.89 (9.21)

Tableau 4.8 Résultats p-chemins MPSS

exemple, lorsqu'une solution améliorante est trouvée par une procédure individuelle, cette même solution est gardée inchangée en mémoire centrale pour les $p-1$ points d'interaction suivants avec la mémoire centrale. Cette approche présente quelques difficultés d'implantation évidentes: par exemple, on doit porter attention au cas où la même procédure communique plusieurs fois consécutivement avec la mémoire centrale, autrement toutes les procédures n'auront pas la même solution initiale. Plus important encore, de telles approches contredisent l'objectif de la stratégie du pool de solutions qui vise à diversifier la recherche en envoyant des solutions différentes mais de bonne qualité aux différentes procédures. De plus, même s'il y a de bonnes raisons pour démarrer toutes les procédures à partir de la même solution initiale (par exemple, pour les problèmes P13 à P16, la seule solution initiale qui semble fonctionner consiste à partir avec tous les dépôts ouverts), de contraindre les procédures de recherche asynchrones d'accepter la même solution en utilisant un de ces mécanismes, est équivalent à réintroduire une forme subtile de synchronisation. Ceci se reflète dans le fait que la stratégie du pool semble obtenir de moins bonnes performances dans un contexte SPDS.

4.9.5 Stratégies collégiales asynchrones MPSS et MPDS

Une deuxième remarque concerne le comportement des stratégies avec multiples solutions initiales, MPSS et MPDS. En général, ces paradigmes sont relativement faciles à maintenir dans un environnement parallèle asynchrone, puisque les procédures font des requêtes à la mémoire centrale indépendamment les unes des autres. C'est d'autant plus vrai dans le contexte des stratégies du pool de solutions où l'ensemble des solutions disponibles est modifié de manière imprévisible et où le choix dans ce pool de la solution retournée à une procédure s'exécute de façon aléatoire. La procédure de base des stratégies asynchrones sans pool peut poser problème puisque seulement une solution, la meilleure solution courante, est disponible en tout temps, et n'est mise à jour que lorsqu'un procédure de recherche trouve une solution

améliorante. Conséquemment, parce que la solution de la mémoire centrale est mise à jour moins fréquemment au fur et à mesure que la recherche globale progresse, il est possible que les recherches MPSS et MPDS deviennent éventuellement des recherches SPSS et SPDS respectivement, ce qui pourrait constituer une importante source de dégradation des performances. Le fait que (lorsque 16 processeurs sont utilisés) les gaps moyens soient les mêmes pour les stratégies MPDS et SPDS, semble nous orienter vers cette conclusion.

Deux observations doivent cependant être faites relativement à ce dernier point. Notons d'abord que dans le cas de l'approche MPDS, une stratégie différente de recherche adoptée par chaque procédure assure une exploration plus diversifiée de l'espace de solutions. C'est probablement ce qui explique la supériorité, apparente dans les figures 4.7 et 4.8, de MPDS sur MPSS. D'autre part, lorsqu'on compare les solutions MPDS et SPDS, lorsque la solution optimale n'est pas trouvée pour un problème donné, la meilleure solution obtenue par chaque procédure est généralement différente et provient d'une itération différente de celle obtenue par les autres procédures. Ceci est spécialement intéressant lorsqu'on considère le cas des problèmes P13 à P16, puisqu'ici toutes les procédures de recherche démarrent avec la même solution initiale, le paradigme des solutions initiales différentes (MPSS et MPDS) n'étant implantées qu'à travers la gestion de la mémoire centrale.

Nous pouvons donc conclure que les chemins de recherche des deux stratégies de différentiation MPDS et SPDS ne convergent pas vers un chemin unique. De même, le phénomène de dégradation où plusieurs solutions initiales différentes MPSS et MPDS se transforment en une stratégie de type SPSS ou SPDS, n'est pas tellement présent dans nos tests. Cela démontre qu'une conception adéquate et le caractère non prévisible des communications inhérent à la parallélisation asynchrone peut produire une stratégie de parallélisation qui donne une plus large exploration de l'espace de solutions.

Prob	p = 4		p = 8		p = 16		Seq
	Gap	Iter	Gap	Iter	Gap	Iter	
	Mécanisme de base de la mémoire centrale						
P1	0	119	0	119	0	33	0
P2	0.32 (0.350	275	0 (0.42)	253	0.06	217	0.42 (1.48)
P3	0	73	0	72	0	73	0.01 (0.8)
P4	0.23 (0.56)	296	0	297	0	254	0.9 (1.1)
P5	0	292	0	53	0	188	0.77 (1.45)
P6	0.15 (1.24)	243	0.05 (0.93)	256	0.01 (0.02)	297	0.42 (3.67)
P7	0.07	223	0.07	154	0	216	0.07
P8	0	65	0	272	0	81	2.15
P9	0	149	0	240	0	61	0
P10	0	99	0.14 (1.79)	286	0 (2.72)	297	1.49
P11	0	289	0	135	0	72	1.52
P12	0	33	0	33	0	33	0.11
P13	0.002 (0.02)	276	0.002 (0.07)	292	0 (0.03)	244	0.002 (0.07)
P14	0.02 (0.2)	285	0.06 (1.0)	284	0.02	290	0.02 (0.59)
P15	0.55 (0.81)	263	0.26 (0.29)	291	0.26 (0.6)	275	1.23 (1.54)
P16	0.23	226	0.23	284	0.23	223	0.89 (9.21)
	Stratégie avec pool $s = p$						
P1	0	255	0	103	0	21	0
P2	0.21 (1.87)	165	0.07 (0.15)	163	0.21 (0.3)	256	0.42 (1.48)
P3	0	201	0	106	0	75	0.01 (0.8)
P4	0.05	228	0.23 (0.3)	240	0.05 (0.55)	235	0.9 (1.1)
P5	0.38 (0.13)	210	0	122	0	116	0.77 (1.45)
P6	0.05 (0.6)	256	0 (0.12)	257	0.05 (0.69)	196	0.42 (3.67)
P7	0	255	0.07	110	0	264	0.07
P8	0	280	0	127	0	217	2.15
P9	0	266	0	54	0	283	0
P10	0.17 (1.53)	211	0.42 (1.01)	272	0	225	1.49
P11	0	281	0	186	0	240	1.52
P12	0	57	0	33	0	33	0.11
P13	0.002 (0.07)	221	0.002 (0.05)	300	0.002 (0.06)	200	0.002 (0.07)
P14	0.02 (0.6)	299	0.06 (0.44)	256	0.06 (0.29)	263	0.02 (0.59)
P15	0.59 (0.81)	253	0.26 (0.29)	266	0	290	1.23 (1.54)
P16	0.23	226	0.23	218	.23	256	0.89 (9.21)

Tableau 4.9 Résultats p-chemins MPDS

4.9.6 Comparaisons des résultats pour les stratégies asynchrones

Les résultats des tests des stratégies de parallélisation asynchrones sont graphiquement illustrés dans les figures 4.7 et 4.8 qui montrent l'évolution des gaps moyens respectivement pour les problèmes P1 à P12, et pour les problèmes P1 à P16.

Un premier indice de l'impact de la parallélisation asynchrone de la recherche tabou est visible au niveau de la qualité des solutions trouvées, plus précisément, cette qualité semble s'accroître avec le nombre de processeurs, par exemple, en général de meilleures solutions sont trouvées en utilisant 8 ou 16 processeurs au lieu de 4. Les stratégies qui semblent tirer le plus d'avantages d'un accroissement du nombre de processeurs sont MPDS, SPDS et MPDS avec le mécanisme de pool de solutions. Ceci indiquerait que le succès de la parallélisation dépend de la capacité à différentier le chemin de chaque procédure. Nous avons aussi exécuté les stratégies MPDS pour 600 itérations sur 4 processeurs, mais les performances ne sont pas meilleures que l'exécution de 300 itérations sur 8 processeurs. D'autres recherches pourront mieux clarifier cette problématique, mais nos tests initiaux semblent indiquer qu'un accroissement du nombre de processeurs est plus bénéfique qu'un accroissement du nombre d'itérations.

Cependant, le même phénomène de saturation dans la capacité de diversifier la recherche avec plusieurs procédures semble se manifester au niveau des parallélisations asynchrones comme pour les stratégies synchrones. Nous avons pu observer qu'il y a une limite sur l'efficacité d'ajouter plus de processeurs pour la résolution d'un problème. Généralement, soit que le gap moyen diminue légèrement ou même s'accroît, lorsqu'on passe de 8 à 16 processeurs. Ce phénomène de saturation provient de deux sources essentiellement: la redondance des chemins d'exploration et l'inefficacité de certaines stratégies d'exploration. Étant donné la taille des problèmes considérés (16 procédures différentes) ceci peut impliquer que certaines solutions initiales et stratégies d'exploration sont très loin de celles identifiées comme "bonnes" pour la recherche tabou séquentielle et peuvent entraîner de manière

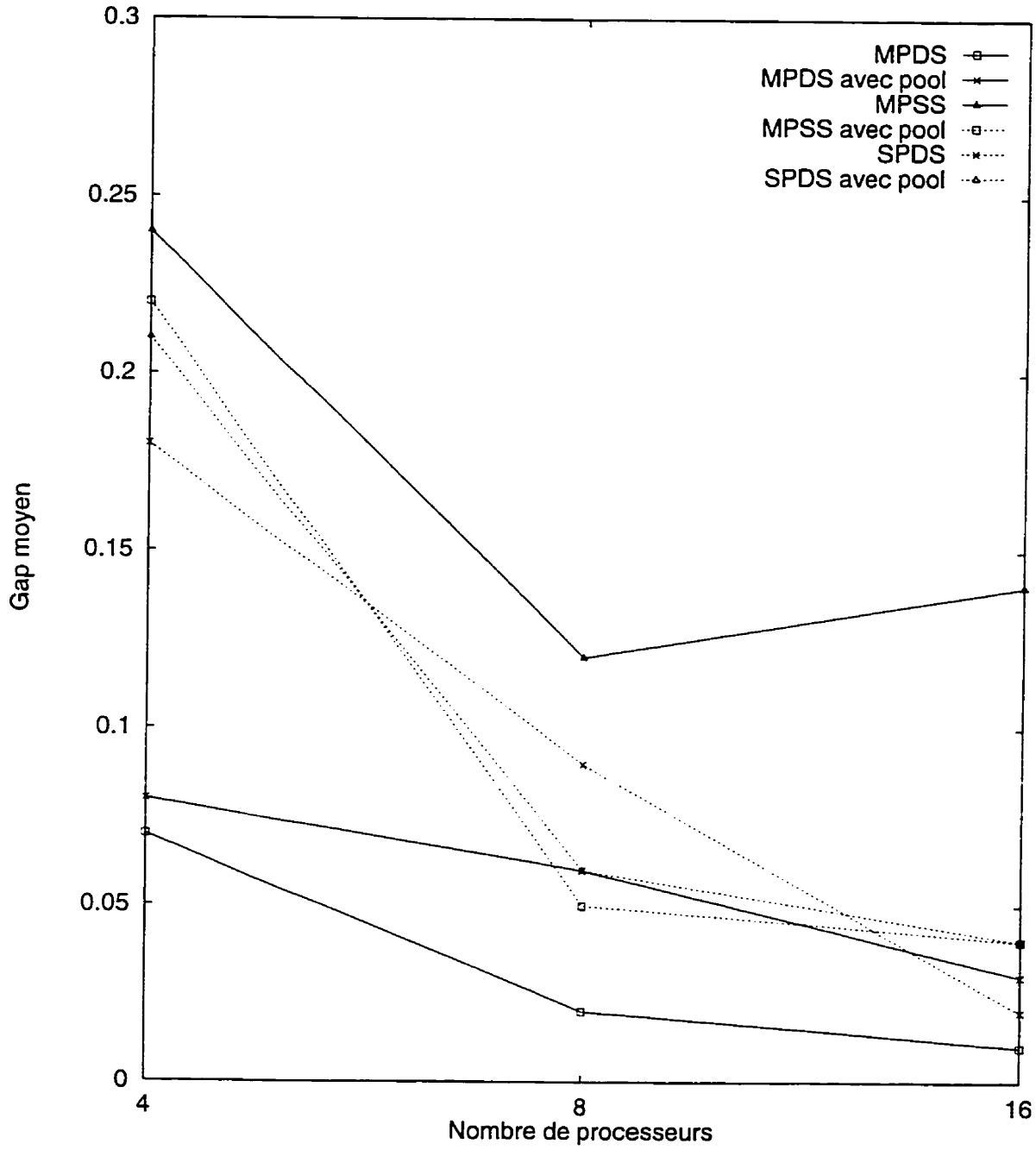


Figure 4.7 Comparaisons des gaps pour les problèmes P1 à P12

systématique les chemins de recherche dans des régions de l'espace de solutions qui n'aident pas à la découverte de bonnes solutions. La redondance des chemins provient du manque de différentiation au niveau des stratégies de recherche, ce qui fait que les mêmes solutions sont explorées par plusieurs procédures différentes. La redondance a un effet pernicieux au niveau de l'exploration globale. En effet, en visitant plusieurs fois la même région de l'espace de solutions, beaucoup d'informations s'accumulent dans les mémoires des différentes procédures sur cette région, lui donnant un pouvoir d'attraction qui ne devrait exister que pour les vallées profondes. Par conséquent, il semble que 8 processeurs correspond au nombre adéquat de processeurs pour la taille des problèmes que nous avons traités. Cette conclusion est aussi corroborée par le comportement du nombre moyen d'itérations requis pour atteindre la meilleure solution, où aller de 8 à 16 processeurs est difficilement justifié.

Concernant la qualité et la gestion de l'information gardée dans la mémoire centrale et échangée entre les procédures, nos résultats suggèrent qu'en utilisant un pool des p meilleures solutions déjà trouvées, et en allouant de manière aléatoire une de ces solutions à une procédure en particulier, cela accroît le nombre de régions différentes dans l'espace de solutions qui sont explorées et améliore la qualité des solutions. C'est intéressant, puisqu'aucun réglage n'a été effectué sur les procédures et que nous avons utilisé un mécanisme très simple d'acceptation des solutions dans le pool: une solution est acceptée si elle est meilleure que la moins bonne solution dans le pool. Finalement, les procédures semblent très robustes relativement à la taille du pool gardé en mémoire centrale. Nous avons fait un certain nombre d'autres tests avec des tailles de pool différentes de p, bien que les performances peuvent varier pour quelques problèmes individuels, le comportement général des procédures ne varie pas.

Parmi les stratégies qui ont été testées, MPDS et SPDS semblent avoir de meilleures performances, relativement au gap moyen et à l'effort requis pour trouver la meilleure solution, avec MPDS légèrement meilleure lorsque 8 processeurs sont

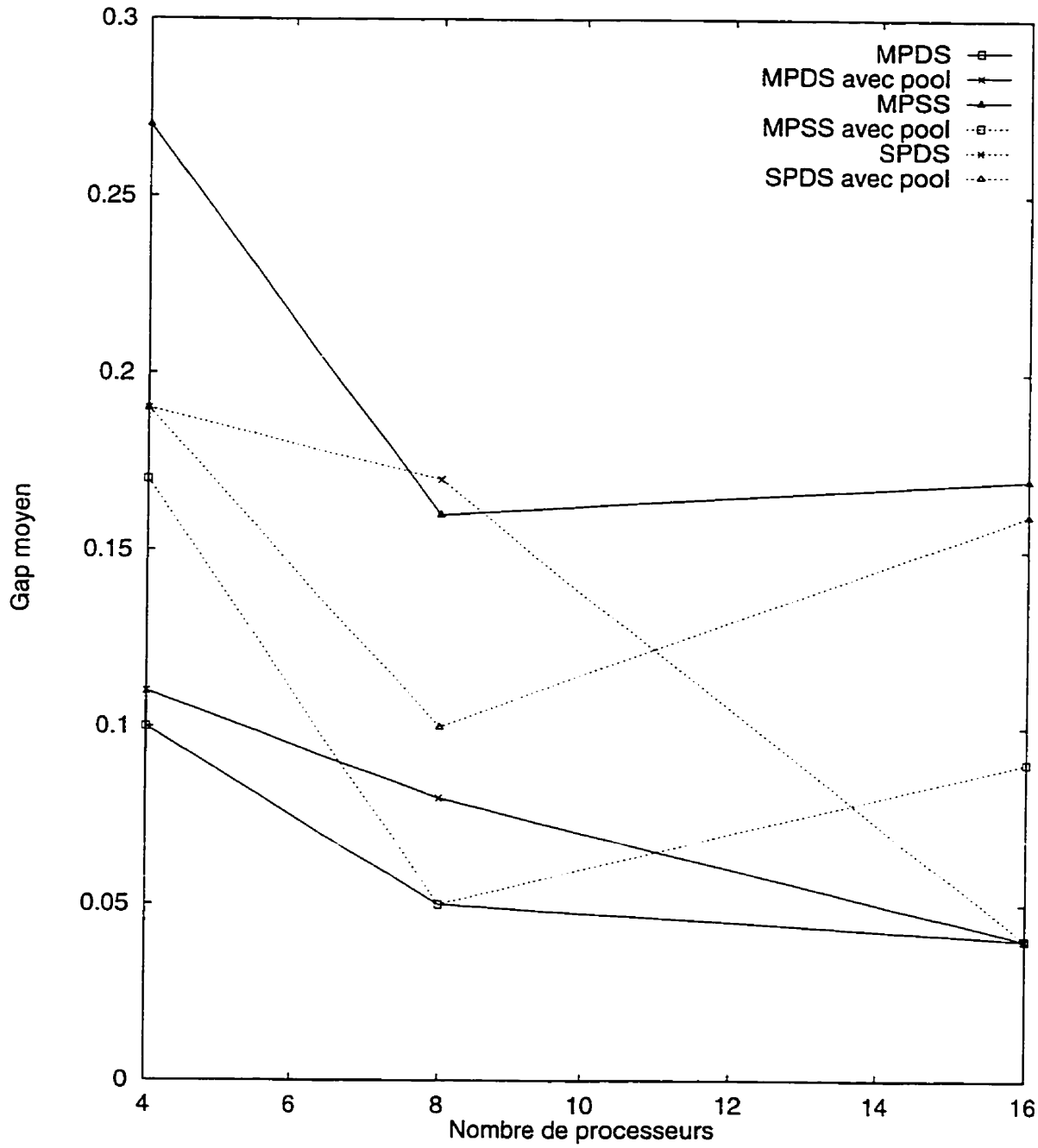


Figure 4.8 Comparaisons des gaps entre les implantations asynchrones pour tous les problèmes

utilisés. En fait, il est intéressant de noter que les recherches tabous parallèles asynchrones apparaissent remarquablement robustes dans leur capacité à atteindre une solution de bonne qualité pour un vaste éventail de stratégies algorithmiques et de réglages de paramètres.

En ce qui concerne l'analyse des dégradations de performances, elle est la même pour toutes les approches asynchrones. Il y a essentiellement trois sources de dégradation dans les parallélisations asynchrones de la recherche tabou:

- la procédure qui exécute le plus grand nombre de swaps détermine le temps total d'exécution de la procédure parallèle (c'est le même cas que les stratégies p-RM);
- les instructions qu'il faut exécuter pour envoyer, recevoir les messages et implanter les nouvelles procédures de guidage du chemin dans le cas de la catégorie réflexif asynchrone;
- la gestion de la mémoire centrale et la congestion pour accéder à cette mémoire.

Les deux premières sources de dégradation ont été décrites au niveau des stratégies synchrones de parallélisation, les stratégies de type réflexif asynchrone ne faisant qu'augmenter la part de dégradation attribuable aux instructions supplémentaires. La troisième source s'accroît en importance avec la quantité du travail effectuée au niveau de la mémoire centrale. Cette source de dégradation est infime lorsqu'une seule solution est conservée en mémoire centrale, elle devient un peu plus importante lorsqu'on considère les stratégies réflexives asynchrones.

4.9.7 Conclusions

La taxonomie a permis d'imaginer de nouvelles stratégies de parallélisation de la méthode tabou en comparaison avec celles que l'on connaît actuellement, et offre un cadre général pour des études comparatives. Il semble, par exemple, que les stratégies de parallélisation des recherches multiples soient très performantes et

donnent de meilleurs résultats que les autres stratégies synchrones. Cependant, si cette approche est choisie, il semble qu'il soit bénéfique de varier non seulement la solution initiale, mais aussi les stratégies d'exploration. Par contre, les stratégies asynchrones suggérées par la taxonomie semblent encore plus prometteuses. En laissant les procédures communiquer entre elles sans imposer de synchronisation, ces stratégies trouvent de meilleures solutions pour la classe de problèmes étudiés et surpassent légèrement les stratégies avec recherches multiples. La taxonomie suggère aussi que des stratégies de parallélisation améliorées peuvent être obtenues en extrayant de la connaissance à partir de l'échange d'information entre les procédures.

Il est intéressant de comparer la performance des stratégies asynchrones avec celles des stratégies synchrones, ces comparaisons sont illustrées dans la figure 4.9. On y montre l'évolution du gap moyen obtenu sur tous les problèmes en fonction du nombre de processeurs. On compare les gaps entre la procédure séquentielle et trois procédures synchrones représentatives de leur classe et offrant les meilleures performances: p-RM MPDS, p-KS MPSS, et 1-KS SPSS. Les performances des algorithmes synchrones sont comparées à celles des versions asynchrones de MPDS, MPDS avec pool, et SPDS avec pool. Les résultats illustrés dans cette figure confirment que sur tout le spectre des stratégies de parallélisation, les approches asynchrones sont meilleures. Une exploration plus minutieuse de l'espace de solutions comparée à celle réalisée par les méthodes séquentielles et parallèles synchrones, due au fractionnement de la connaissance qui résulte de l'implantation asynchrone de la recherche est la cause la plus probable de ces résultats. Comme prévu, la stratégie de parallélisation p-RM MPDS, laquelle utilise p procédures de recherche non coordonnées pour explorer l'espace de solutions, obtient de très bonnes solutions. Ce qu'il faut noter cependant, c'est le fait que, même dans le présent contexte où les améliorations possibles sont très restreintes (les gaps sont généralement très minces), une implantation simple d'une recherche parallèle asynchrone peut encore dépasser les meilleures méthodes synchrones.

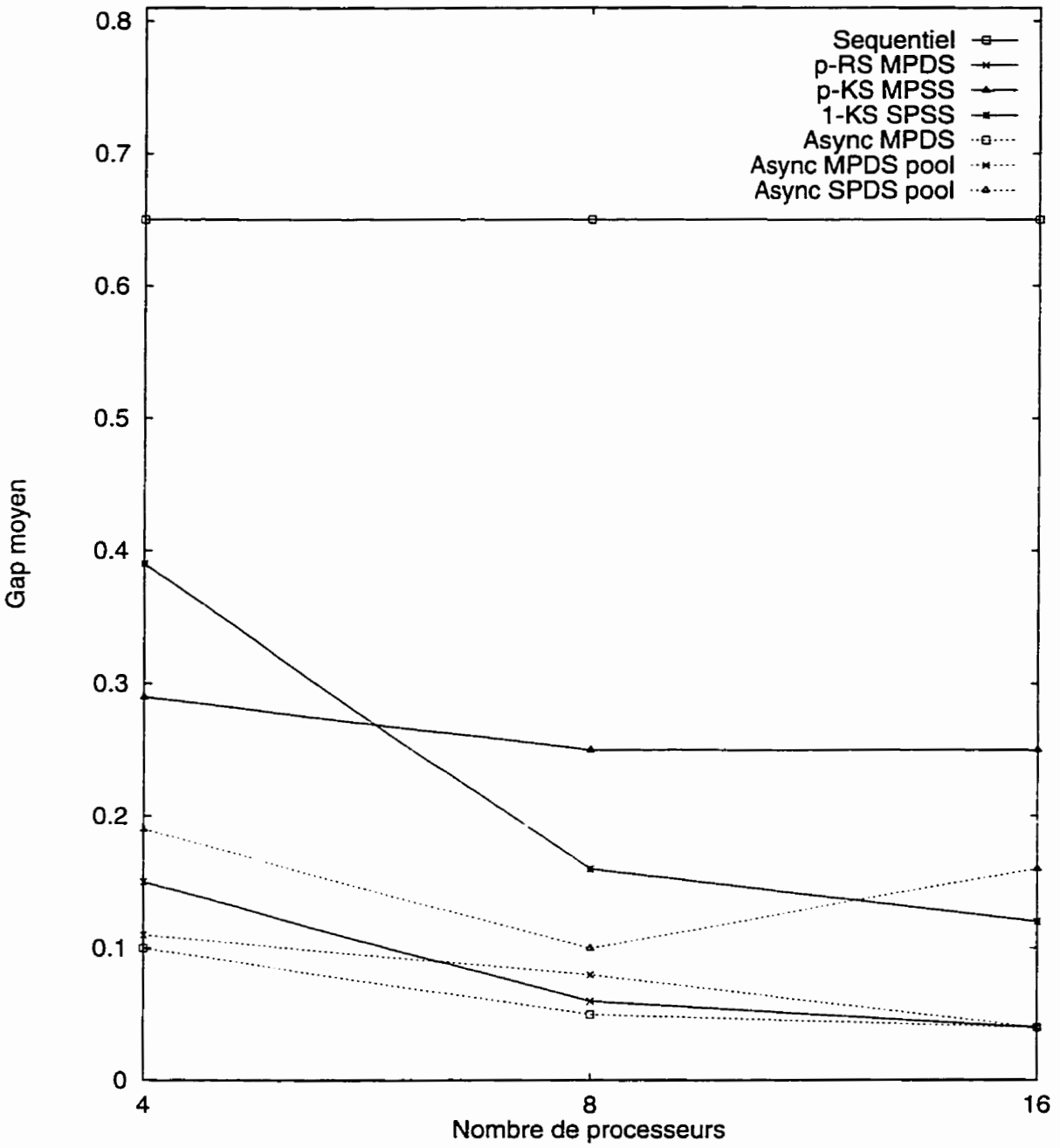


Figure 4.9 Comparaisons des gaps – Meilleures procédures asynchrones et synchrones

Les tableaux 4.10 et 4.11 montrent les gaps moyens, calculés sur l'ensemble des problèmes, obtenus par chaque implantation parallèle pour 4, 8 et 16 processeurs, de même que le gap moyen de la procédure séquentielle. L'évolution des gaps moyens avec le nombre de processeurs est aussi illustrée dans les figures 4.10 et 4.11 pour les procédures asynchrones et synchrones respectivement. Ces résultats supportent la conclusion selon laquelle l'utilisation du parallélisme améliore les performances des procédures de recherche tabou. Les résultats rapportés sur plus de 672 exécutions des implantations synchrones et asynchrones du tabou parallèle montrent que les procédures parallèles s'arrêtent avec la même solution que la procédure séquentielle dans 25% des cas, tandis qu'il y a amélioration dans 68% des autres. En particulier, la solution optimale est trouvée dans 48% du temps, comparé à 12% pour la version séquentielle.

p	SEQ	1-RS	1-KS	p-RM			p-KS		
		SPSS	SPSS	SPDS	MPSS	MPDS	SPDS	MPSS	MPDS
4	0.63	0.51	0.39	0.32	0.20	0.15	0.53	0.65	0.61
8	0.63	0.45	0.16	0.18	0.17	0.06	0.49	0.50	0.32
16	0.63	0.35	0.12	0.08	0.15	0.04	0.44	0.41	0.21

Tableau 4.10 Pourcentage des gaps (%) - Procédures synchrones

p	SEQ	p-C			p-KC		
		SPDS	MPSS	MPDS	SPDS	MPSS	MPDS
4	0.63	0.19	0.28	0.10	0.25	0.44	0.15
8	0.63	0.17	0.15	0.05	0.31	0.36	0.20
16	0.63	0.04	0.17	0.04	0.13	0.13	0.19

Tableau 4.11 Pourcentage des gaps (%) - Procédures asynchrones

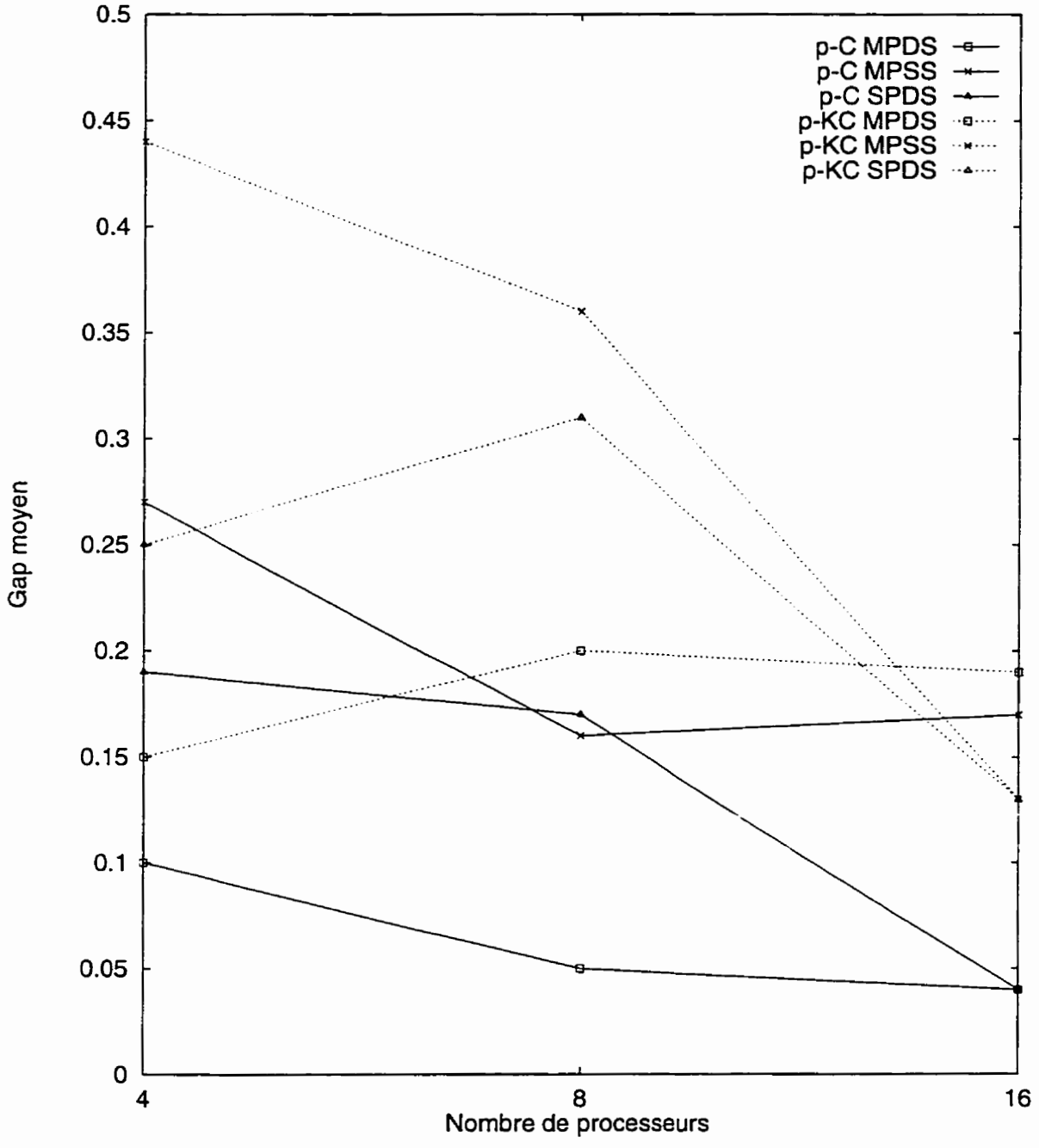


Figure 4.10 Comparaisons des gaps– Implantations asynchrones

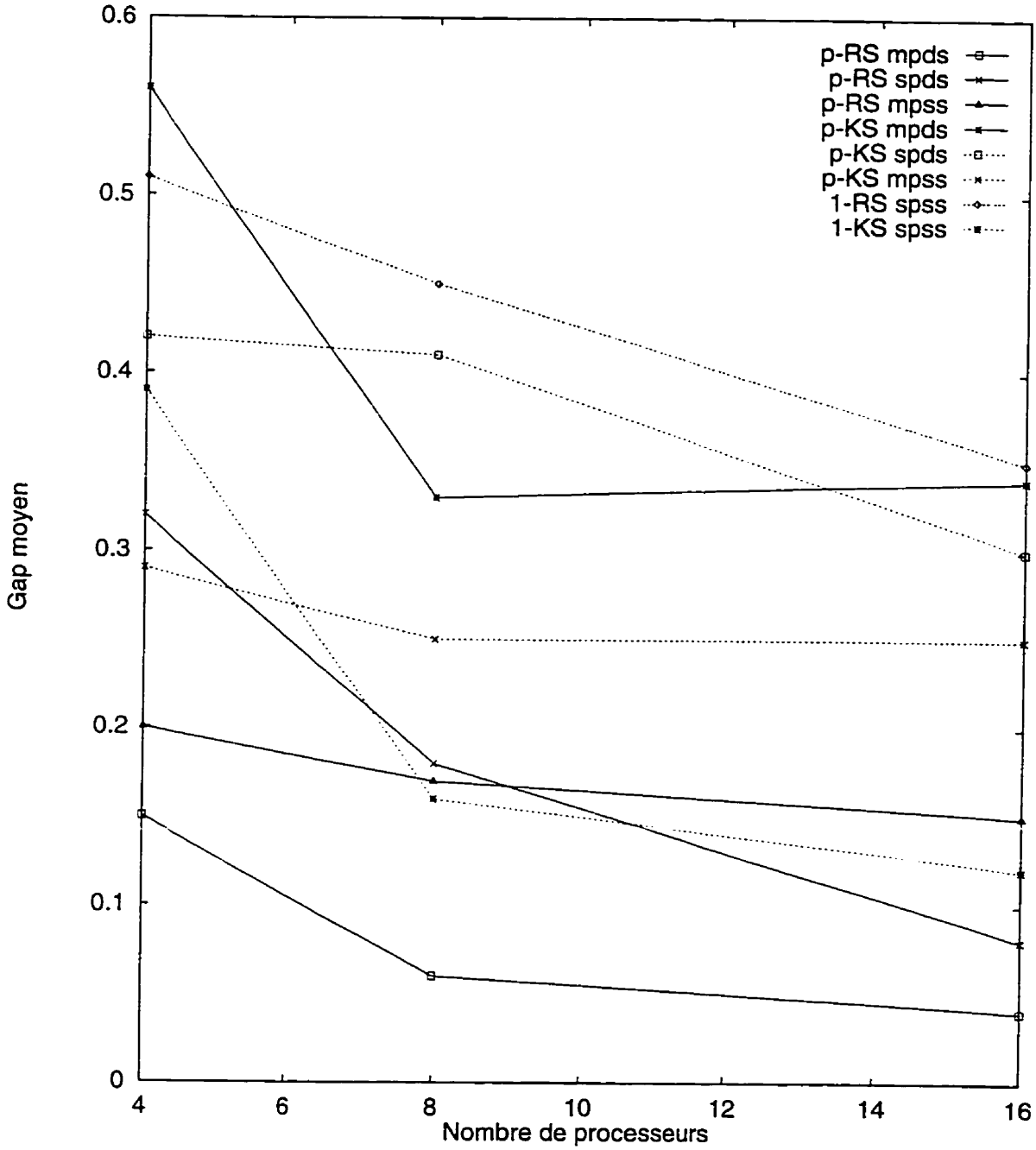


Figure 4.11 Comparaisons des gaps– Implantations synchrones

Chapitre 5

Parallélisation par recherches multiples coopérantes

5.1 Introduction

Au chapitre 4, nous avons implanté plusieurs procédures parallèles dont les stratégies de parallélisation appartiennent aux recherches multiples coopérantes. Lors de la conception de ces procédures parallèles, nous avons fait plusieurs hypothèses sur l'information qui devait être échangée (les meilleures solutions), la fréquence des échanges (25 itérations ou communications asynchrones) et sur les procédures séquentielles visées par un échange d'information (toutes les procédures dans le cas p-KS ou indéterminées pour p-KC et collégiales asynchrones). Ces décisions portant sur l'échange d'information ont constitué une part importante du travail de conception de ces procédures parallèles.

Cette phase de conception visant à spécifier comment l'information doit être échangée entre les procédures séquentielles est particulière aux recherches multiples coopérantes. On ne la retrouve pas dans la conception de procédures parallèles exploitant le parallélisme obligatoire ou le parallélisme par décomposition du domaine du traitement spéculatif. La problématique portant sur "comment faire coopérer entre elles les procédures séquentielles" de recherches multiples est relativement nouvelle en ce qui concerne la parallélisation de la méthode tabou. Cependant, cette problématique a fait l'objet de plusieurs travaux de recherche depuis près d'une décennie pour des méthodes de recherche comme les algorithmes génétiques, le recuit simulé, de même que certaines méthodes spécialisées. Il n'en demeure pas

moins que cette phase de conception que l'on appelle parfois "l'engineering du traitement parallèle coopératif" est encore mal définie. La compréhension des effets du partage de l'information sur le chemin d'exploration de chaque procédure séquentielle reste intuitive et incomplète, et une méthode systématique portant sur la manière de mener à bien cette étape de conception fait toujours défaut.

Le présent chapitre traite de la phase de conception de l'échange d'information entre les procédures séquentielles d'une stratégie de parallélisation par recherches multiples coopérantes. Nous allons proposer une démarche d'analyse de la méthode tabou destinée à guider le développement de cette phase de conception. Plus spécifiquement, nous proposerons un cadre général où la configuration de cette phase de conception est bien définie. Nous allons également énoncer certains critères permettant d'évaluer les choix qui s'offrent pour réaliser l'échange d'information pour chaque aspect spécifique du cadre général. Chaque critère sera motivé par un ensemble de considérations provenant de notre expérience avec la parallélisation de la méthode tabou et de travaux de chercheurs portant sur d'autres méthodes de recherche. Le concepteur pourra alors être en mesure de comparer le contexte de développement de sa procédure parallèle avec les hypothèses théoriques et pratiques qui ont conduit à l'énoncé de ce cadre général et des critères qui l'accompagnent.

5.2 Travaux sur les recherches multiples coopérantes

Comme mentionné dans l'introduction, il existe déjà certains travaux portant sur des stratégies de parallélisation par recherches multiples coopérantes appliquées à des méthodes de recherche autres que la méthode tabou. Nous allons maintenant introduire quelques-uns de ces travaux, nous reviendrons sur des aspects plus particuliers de ces recherches pour étayer certaines considérations sur l'échange d'information entre les procédures séquentielles de la méthode tabou.

5.2.1 Méthodes de recherche spécialisées

Les travaux de Clearwater, Huberman & Hogg [25] portent sur l'utilisation de stratégies de parallélisation par recherches multiples coopérantes pour la résolution en parallèle d'un problème d'arithmétique cryptographique. Il s'agit d'un problème de satisfaction de contraintes consistant à trouver une affectation unique d'un nombre à chaque caractère telle que les nombres composés par des mots différents puissent s'additionner correctement. Dans la version séquentielle de cette méthode de recherche, chaque itération effectue une affectation aléatoire unique de nombres aux caractères. Dans la version parallèle avec recherches multiples coopérantes, les procédures séquentielles s'échangent de l'information qui correspond à des affectations partielles de nombre à des caractères s'additionnant correctement. La version parallèle consiste donc à remplacer au moins une partie de l'affectation aléatoire faite à chaque itération des procédures séquentielles par une affectation partielle contenue dans une mémoire centrale (architecture blackboard). Les procédures échangent ces solutions partielles par la lecture et l'écriture de manière asynchrone de la mémoire centrale. La mémoire centrale n'est pas limitée (c.a.d. qu'il n'y a pas de limites au nombre de messages contenus dans cette mémoire) et le choix du message en mémoire centrale par une procédure séquentielle se fait par un tirage aléatoire. Les résultats de cette recherche montrent une amélioration sensible des performances de la stratégie avec recherches multiples coopérantes par rapport à la stratégie par recherches multiples.

Hogg et Williams [55] appliquent une stratégie de parallélisation par recherches multiples coopérantes à un problème de coloriage de graphe en utilisant deux méthodes différentes de recherche. La première, est une recherche en profondeur avec backtracking (une sorte de B&B) qui tente d'abord de faire une affectation de couleurs au niveau des noeuds ayant le plus de contraintes (le plus de noeuds adjacents). La seconde méthode est une heuristique qui, à partir de configurations initiales générées de manière aléatoire, essaie de produire une solution en changeant

l'affectation des couleurs des noeuds afin de réduire le nombre de contraintes violées dans le problème. Ces deux méthodes sont très différentes, l'une procède avec des solutions partielles et converge vers une solution réalisable tandis que l'autre commence avec une solution complète non réalisable et tente de l'améliorer sans garantie de convergence. En programmant deux méthodes différentes, les auteurs cherchaient à obtenir une plus grande diversité du comportement des procédures séquentielles engagées dans une stratégie de parallélisation par recherches multiples coopérantes. Comme pour le cas précédent, l'implantation de l'interaction se fait par l'intermédiaire d'une architecture blackboard centralisée où les procédures séquentielles vont écrire et lire des messages. Les messages sont des solutions partielles, c'est-à-dire des sous-ensembles de noeuds colorés de manière cohérente. Un nombre limité de messages sont gardés dans la mémoire centrale, lorsque celle-ci est pleine, les plus anciens sont éliminés. Des tests effectués avec 10 procédures séquentielles avec interaction ont montré de meilleurs résultats par rapport à un même ensemble de recherches sans interaction.

5.2.2 Méthodes de recherche générales

Plusieurs implantations distribuées d'algorithmes génétiques à gros grain, Pettey, Leuze & Grefenstette [78], Tanase [95] et Gordon & Whitley [46] et à grain fin, Manderick & Spiessens [68], Mühlenbein, Gorges-Schleuter & Kramer [74] et Gordon, Whitley & Bohm [47] suivent le schéma d'une stratégie de parallélisation par recherches multiples coopérantes. Il en va de même pour des algorithmes distribués de recuit simulé, Greening [51] et Graffigné [48]. Il serait trop long de décrire et d'introduire tous ces travaux. Dans les trois prochaines sections, nous décrirons plus en détail certains aspects de ces recherches.

5.2.3 Modèle probabiliste d'une stratégie de parallélisation par recherches multiples coopérantes

Huberman [56] a introduit un modèle général visant à expliquer comment l'échange d'information entre les recherches multiples affecte: (1) le parcours du chemin d'exploration des procédures séquentielles; (2) les performances de la procédure parallèle. Ce modèle est basé sur une approche probabiliste. Selon ce modèle, les meilleurs résultats obtenus par des recherches multiples coopérantes proviennent du fait que l'échange d'information entre les procédures séquentielles, bien que néfaste à certaines d'entre elles, est aussi plus bénéfique aux procédures séquentielles obtenant les meilleures solutions. L'échange d'information provoquerait une plus grande variation entre les solutions trouvées par les recherches qui coopèrent en comparaison avec les solutions trouvées par les recherches multiples sans coopération. La figure 5.1 montre la distribution des meilleures solutions trouvées par les mêmes procédures séquentielles avec et sans coopération. Cette figure montre que la plus grande variation des solutions obtenues par recherches multiples coopérantes se traduit par une distribution lognormale contrairement à une distribution normale pour la parallélisation par recherches multiples. La courbe lognormale se caractérise par une longue traînée de chaque côté de la moyenne. Conséquemment, la stratégie de parallélisation par recherches multiples coopérantes aura quelques procédures séquentielles dans la traînée représentant les solutions de plus grande qualité. La seule solution importante dans une approche par recherches multiples correspond à la meilleure solution obtenue parmi les p procédures séquentielles. La coopération entre les recherches multiples améliorerait les résultats d'une procédure parallèle en obtenant de meilleurs résultats du sous-groupe des procédures séquentielles les plus performantes.

Ce modèle n'est pas très opérationnel en ce sens qu'il ne nous donne pas beaucoup d'indication sur la façon de concevoir l'échange d'information entre les procédures séquentielles. De plus, comme nous le verrons au prochain chapitre,

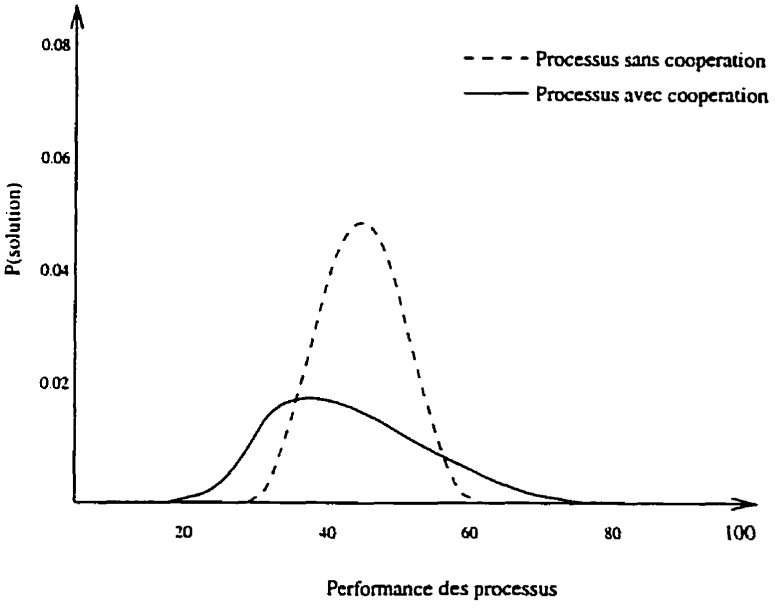


Figure 5.1 Distribution des solutions avec et sans coopération

ce modèle repose sur une hypothèse d'indépendance des procédures séquentielles qui n'est pas satisfaite évidemment lorsqu'il y a échange d'information entre ces procédures.

5.2.4 Cadre général: la notion de couplage entre les procédures séquentielles

Il n'existe pas de phase de conception de l'échange d'information pour les stratégies de parallélisation obligatoire. La définition des tâches parallèles repose en effet sur la décomposition de l'algorithme logique ou séquentiel. Par exemple, pour un algorithme logique prenant la forme d'un graphe $G = \{N, A\}$ (voir la section 2.2.2), l'information échangée entre deux tâches parallèles X et Y est donnée par les arcs (u, v) où $u \in \{X, Y\}$, $v \in \{X, Y\}$ et $(u, v) \notin X$, $(u, v) \notin Y$. Il n'est donc pas nécessaire d'avoir une phase de conception puisque l'échange d'information du parallélisme obligatoire dérive directement des dépendances de données existant au

niveau de l'algorithme logique (ou de l'algorithme séquentiel), de la granularité de la parallélisation et du type de décomposition. Une situation similaire existe pour les stratégies de parallélisation basées sur une décomposition du domaine du traitement spéculatif. S'il y a échange d'information heuristique au niveau de la procédure séquentielle, le même type d'information sera échangé par la procédure parallèle.

Nous avons emprunté au domaine du génie logiciel et de la théorie des systèmes complexe le concept de *couplage* comme mesure de l'interaction entre les procédures séquentielles des parallélisations avec recherches multiples coopérantes.

Le couplage fait référence au degré d'inter-dépendance entre les composantes d'un système ou les modules d'un programme. Par exemple, on peut utiliser cette notion pour décrire le niveau d'inter-dépendance des tâches d'une parallélisation obligatoire en montrant comment les tâches interagissent entre elles, c'est-à-dire quelles données sont échangées, quand et entre quelles tâches l'échange se fait. Nous utiliserons cette notion de couplage comme cadre général visant à obtenir une approche de développement plus structurée pour la phase de conception de l'échange d'information des recherches multiples coopérantes. L'idée de couplage synthétise bien les principaux aspects d'une interaction efficace entre les procédures séquentielles, à savoir, la définition de *quelle* information doit être partagée, *quand* cette information doit être partagée, et *où*, c'est-à-dire entre quelles procédures séquentielles l'information doit être partagée (nous référerons à ce dernier aspect du couplage par la notion *structure de voisinage* entre les procédures séquentielles). L'énoncé explicite de ces trois paramètres dans la conception d'une stratégie de parallélisation par recherches multiples coopérantes basée sur la méthode tabou constitue la spécification du couplage entre les procédures séquentielles. Chacune des trois sections qui suivent est consacrée à discuter un aspect particulier du couplage.

5.3 Quelle information doit être échangée

Parmi les trois aspects du couplage des procédures de recherche dont nous discutons dans ce chapitre, l'aspect concernant le choix de l'information à partager est celui qui est façonné le plus directement par la méthode de recherche utilisée.

Échange d'information entre les sous-populations d'un algorithme génétique

Pour les algorithmes génétiques, l'échange d'information a pour but d'améliorer la qualité des solutions trouvées et de maintenir la diversité entre les sous-populations. La qualité et la quantité des individus qui se trouvent dans chaque message entre les sous-populations sont les deux paramètres qui réduisent ou accentuent la pression sur la sélectivité. Le choix de l'information à échanger entre les sous-populations se limite donc à ces deux seuls paramètres. Lorsqu'une sous-population reçoit constamment des individus similaires et de très bonne qualité, ces individus risquent de devenir prédominants dans la sous-population ce qui accentue la sélectivité et a un impact négatif sur la diversité. Par contre, lorsque les individus sont choisis au hasard pour l'échange entre les sous-populations, la pression sur la sélectivité est réduite et la diversité est peut-être maintenue, mais la qualité des solutions trouvées par les sous-populations risque de ne pas s'améliorer. Les critères pour déterminer la qualité et la quantité des individus dans un message sont donc fortement influencés par ces deux dernières considérations.

Différents travaux de recherche sur les algorithmes génétiques distribués ont donc été effectués pour connaître l'incidence du partage d'information sur les sous-populations. Pour ne mentionner que deux exemples particuliers parmi d'autres, Belding [12] a fait des tests avec des taux d'échange de 10%, 20% et 50% des meilleurs individus des sous-population, tandis que Levine [66] a testé deux stratégies de sélection des individus migrants: une première stratégie consistant à faire émigrer le meilleur individu vers des sous-populations voisines tandis que pour la deuxième stratégie les individus étaient choisis selon un modèle probabiliste avec un biais

de 0.6 en faveur des meilleurs individus, dans le but de réduire la pression sur la sélectivité.

En ce qui concerne le recuit simulé, le choix de l'information à échanger entre les procédures séquentielles se limite presqu'uniquement à celui de la meilleure solution trouvée. Au niveau des procédures de fouille que nous avons examinées dans la section 5.2, des solutions partielles sont utilisées comme information échangée.

Stratégies d'échange d'information de la méthode tabou

Le choix de l'information à être partagée entre les procédures séquentielles de la méthode tabou s'effectue à partir du contenu des nombreuses mémoires constituant l'historique de recherche de chaque procédure séquentielle. Rappelons que la méthode tabou se base sur des règles de transition qui tiennent compte d'information heuristique en provenance de l'historique de recherche (les différentes mémoires de la méthode tabou). Trois stratégies de partage d'information s'offrent pour définir une procédure parallèle par recherches multiples coopérantes:

1. Concevoir la phase d'échange d'information à partir des mémoires existantes des procédures séquentielles (les procédures parallèles p-KS et collégiales asynchrones ont été conçues à partir de cette stratégie);

2. Associer la conception de la méthode tabou avec celle de la phase relative à l'échange d'information;

3. Enrichir les règles de transition d'une information qui ne provient pas directement de l'exploration de l'espace de solutions mais dérive plutôt de la manipulation de l'information échangée entre les procédures (p-KC).

Dans le premier cas, la conception relative au choix de l'information se fait sans modifier la méthode tabou utilisée. L'objectif visé est de permettre aux règles de transition d'accéder les mémoires de plusieurs procédures séquentielles et conséquemment d'accroître l'information dont chaque règle de transition dispose. Dans

la deuxième situation, de nouvelles règles de transition peuvent être créées pour exploiter l'information de nouvelles mémoires ajoutées à la méthode tabou originale et ainsi profiter du fait qu'il existe un couplage entre les procédures séquentielles. Il y a alors interaction entre la conception de la méthode tabou et celle de l'échange d'information entre les procédures. Enfin, la troisième alternative vise à définir des procédures de traitement du contenu des mémoires partagées pour en déduire de l'information additionnelle relative à l'espace de solutions qui soit utile aux règles de transition existantes.

Dans les sections qui suivent, nous décrirons plus en détails les enjeux relatifs à chacun de ces trois choix stratégiques.

5.3.1 Partage des mémoires de la méthode tabou

Pour certaines règles de transition, leur efficacité peut croître en fonction du nombre d'itérations exécutées par la procédure séquentielle. C'est le cas, par exemple pour la borne supérieure de la méthode d'énumération implicite par évaluation et séparation, dont l'efficacité est meilleure au fur et a mesure que la recherche progresse. Ce type de règles devrait donc pouvoir bénéficier d'un schéma de recherches multiples coopérantes en ayant accès à l'information heuristique de p procédures séquentielles. Cette approximation de l'impact de la coopération entre les recherches multiples sur l'augmentation de l'efficacité des règles de transition doit cependant être nuancée. Certains facteurs peuvent affecter de manière négative le bénéfice global de la coopération entre les recherches multiples.

On peut considérer par exemple le coût des communications inhérent au partage des mémoires entre les procédures, coût qu'une stratégie de parallélisation par recherches multiples n'a pas à assumer. Les mémoires à court, moyen et long terme de la méthode tabou ne sont pas mises à jour selon les mêmes fréquences. Or, il existe une relation entre la fréquence de mise à jour des mémoires et le coût des communications pour le partage de ces mémoires. Ainsi, pour les mémoires qui sont mises

à jour très fréquemment comme c'est le cas pour les listes tabous à court terme, le partage d'une information qui ne soit pas désuète entre les procédures séquentielles va demander un haut débit d'échange de données au niveau des processeurs.

Critère 1 *On peut évaluer la pertinence de partager les mémoires de la recherche tabou en relation avec le coût d'utilisation du réseau de communication à partir de la relation suivante:*

$$it \leq ct \leq max\{it, \frac{s^*(it)}{s^*(ct)} * it\} \tag{5.1}$$

où it et ct correspondent au temps requis pour exécuter t itérations avec p procédures séquentielles pour les stratégies avec recherches multiples et celles par recherches multiples coopérantes respectivement. $s^*(it)$ *correspond à la meilleure solution d'une procédure parallèle par recherches multiples après t itérations, et* $s^*(ct)$ *correspond à la meilleure solution de la stratégie par recherches multiples coopérantes avec le même nombre de t d'itérations.*

La relation 5.1 exprime le fait que, plus le partage d'information améliore la qualité des solutions, plus on peut tolérer un niveau élevé des coûts de communication qui pourraient résulter de ce partage. Cette contrainte peut cependant s'avérer trop restrictive pour certaines applications, et elle peut être élargie sur la base d'observations expérimentales. Néanmoins, les recherches multiples coopérantes basées sur le partage des listes tabous à court terme ne rencontreront presque jamais cette contrainte. Par contre, les mémoires dont le contenu devient lentement désuet, telles que les mémoires à moyen et long terme, demandent moins de transfert d'information sur le réseau de communication, et seront ainsi mieux adaptées aux stratégies de parallélisation par recherches multiples coopérantes.

Un deuxième facteur est à considérer dans le choix de l'information que l'on peut échanger efficacement entre les procédures séquentielles. Il faut, en effet, considérer l'*utilité* du contenu des mémoires d'une procédure séquentielle lorsque ce

contenu est transféré dans le contexte de recherche d'une autre procédure séquentielle. Par exemple, il est peu pertinent dans une implantation normale d'un mécanisme tabou, de transférer le statut tabou des solutions d'une procédure séquentielle à l'autre puisque ce statut dépend de l'historique local de recherche de chaque procédure et peut difficilement servir à d'autres procédures. Malheureusement, la situation n'est pas toujours aussi claire et en général le degré d'intérêt du contenu des mémoires n'est pas facile à représenter à l'aide d'un modèle quantitatif. Cependant, il est parfois possible d'interpréter qualitativement le contenu des mémoires en relation avec la topologie de l'espace de solutions. C'est le cas notamment pour la méthode tabou définie à la section 4.5, où les mémoires à moyen terme sont utilisées pour identifier des régions de l'espace de solutions où une phase d'intensification peut être exécutée.

Critère 2 *Si le contenu d'une mémoire de l'historique de recherche possède une signification intrinsèque indépendamment de l'historique de la procédure séquentielle qui génère cette information, alors ce contenu peut être échangé efficacement entre les procédures séquentielles.*

C'est le cas pour les mémoires à moyen terme de la procédure tabou du chapitre 4 dont le contenu représente la valeur de la fonction objectif et a donc une signification intrinsèque. Lorsque cette situation existe, on peut utiliser le contenu de ces mémoires dans le contexte de n'importe quelle procédure séquentielle.

Comparaison d'une procédure parallèle avec et sans échange d'information

Les tableaux 5.1 et 5.2 comparent les résultats obtenus par 8 procédures séquentielles, respectivement, avec une stratégie par recherches multiples et une stratégie par recherches multiples coopérantes. On observe au niveau du tableau 5.1 des variations dans le degré de succès de chaque procédure séquentielle. Par exemple, la meilleure solution trouvée par la procédure $S6$ a un gap de 3.22% pour le problème $P11$, alors que le gap de la procédure $S2$ est de 0%. Aussi, pour le problème

Problème	Seq.	IST	S1	S2	S3	S4	S5	S6	S7	S8
P1	0	0	.41	0	0	0	.41	0	0	0
P2	.42	.21	.21	.33	.38	.99	.26	.23	.59	.29
P3	.01	0	.03	.63	.57	.57	0	.94	.03	.56
P4	.9	.32	.61	.58	.61	1.12	.32	.37	.39	.88
P5	.77	0	1.25	1.25	.57	.50	0	.90	1.10	1.06
P6	.42	0	.63	.15	.15	.05	.44	0	.05	.15
P7	.07	0	.13	3.69	.73	.73	0	.07	2.19	.07
P8	2.15	0	0	1.32	2.53	2.75	2.68	2.68	0	.85
P9	0	0	1.27	.82	.82	0	0	1.44	0	.82
P10	1.49	.37	1.58	1.59	2.50	1.16	2.33	2.37	2.47	.37
P11	1.52	0	.61	0	3.52	1.38	0	3.22	.61	1.52
P12	.11	0	2.59	1.80	1.46	1.79	0	1.09	0	.11

Tableau 5.1 Recherches multiples, $p = 8$

Problème	IST	P.C.	S1	S2	S3	S4	S5	S6	S7	S8
P1	0	0	0	0	0	0	.41	0	0	0
P2	.21	0	.39	.39	.39	0	.30	.30	.31	.39
P3	0	0	0	0	0	0	0	0	0	0
P4	.32	0	0	0	0	0	0	0	.27	0
P5	0	0	.49	.49	.49	.48	0	0	0	.14
P6	0	.05	.19	.05	.05	.05	.05	.15	.05	.05
P7	0	.07	.07	.07	.07	.07	.10	.07	.10	.07
P8	0	0	0	.79	0	.79	0	.79	.79	0
P9	0	0	0	.82	0	0	1.26	0	0	0
P10	.37	.14	.14	.75	.75	.14	.14	.14	.82	.60
P11	0	0	0	.33	0	0	0	.33	0	0
P12	0	0	.18	1.14	0	1.14	1.38	1.09	0	1.38

Tableau 5.2 Recherches multiples coopérantes, $p = 8$

$P7$, la procédure $S6$ trouve une meilleure solution à .07% de la solution optimale tandis que la procédure $S2$ se termine avec un optimum local à 3.69% de la solution optimale. Ces différences de gap mesurent en quelque sorte le degré de succès des solutions initiales et des paramètres de recherche pour les différents problèmes. Étant donné, que peu d'itérations tabou ont été alloués au traitement (300), les procédures séquentielles commençant avec des solutions initiales correspondant à des régions loin de bons optima locaux, n'ont pas été capables de surmonter leur handicap et ont conclu avec de mauvaises solutions. Par contre, comme cela apparaît dans le tableau 5.2, les solutions initiales non favorables perdent rapidement de leur influence sur la recherche des procédures séquentielles lorsque l'information en provenance de procédures ayant plus de succès prend le contrôle des stratégies peu appropriées. Ces résultats montrent que le partage de l'information se traduit par une plus grande uniformité des solutions trouvées, ce qui peut signifier que les procédures séquentielles ont concentré leur recherche dans de meilleures régions de l'espace de solutions.

5.3.2 Création de nouvelles mémoires

On peut concevoir la phase d'échange d'information en interaction avec celle de la méthode tabou. Cette stratégie de conception est appropriée lorsque le choix de l'information à partager implique des modifications à la méthode tabou pour inclure de nouvelles mémoires et de nouvelles règles de transition, ou des versions distribuées de règles de transition existantes. Des approches similaires ont été testées pour les algorithmes génétiques [74] où des transformations au niveau de la conception de ces méthodes sont introduites pour tirer avantage du couplage entre les procédures séquentielles.

Par exemple, on sait qu'une procédure parallèle avec recherches multiples coopérantes perd une certaine efficacité à cause de la redondance provenant de l'exploration de la même solution par plusieurs procédures séquentielles. L'addition

d'un mécanisme de listes tabou distribuées au niveau de chaque procédure séquentielle peut contrôler ce problème. Le concept est simple, il consiste à utiliser une liste tabou distribuée afin de rendre tabou pour toutes les procédures, toute solution ayant déjà été visitée par une des procédures séquentielles. Étant donné, qu'une interdiction complète de la redondance contraindrait de manière excessive les chemins de recherche, chaque procédure séquentielle peut diffuser périodiquement les solutions visitées, et ces solutions deviennent tabous pour une durée spéficiée par la taille de la liste tabou distribuée.

D'autres règles distribuées de transition peuvent être ainsi imaginées, chacune requérant une information particulière. Cependant, dans tous les cas, cette information devrait être considérée comme faisant partie de "quelle" information doit être partagée.

5.3.3 Création d'information relative à l'espace de solutions

On peut obtenir de l'information additionnelle ne résultant pas directement de l'exploration de l'espace de solutions en exécutant certaines opérations sur les mémoires partagées. Par exemple, certains algorithmes génétiques distribués à grain fin utilisent des méthodes de descentes pour trouver des optima locaux, et appliquent ensuite l'opérateur de crossover à ces optima locaux. On peut considérer les optima locaux trouvés par les algorithmes génétiques comme étant de l'information partagée entre les procédures séquentielles à grain fin. L'application de l'opérateur de crossover est une procédure permettant de créer indirectement de l'information additionnelle sur l'espace de solutions: les enfants. L'opérateur de crossover est une forme d'opération pouvant être exécutée sur des mémoires partagées de la méthode tabou pour obtenir de l'information additionnelle sur l'espace de solutions. D'autres types d'opérateurs peuvent être obtenus à partir de l'étude des distributions des attributs des chemins de recherche, de la fréquence des

changements de statut d'une variable ou d'un ensemble de variables dans de mauvaises ou bonnes solutions, l'effet en moyenne de certains changements de statut des variables sur la valeur de la fonction objectif, etc. Les différentes mémoires de la méthode tabou offrent beaucoup de potentiel pour obtenir un surplus d'information sur l'espace de solutions, au coût de quelques communications inter-processeurs et de quelques opérations de traitement. Ce surplus de connaissance peut être utilisé par les différentes règles de transition.

5.4 Quand l'échange d'information doit se faire

Au niveau du parallélisme obligatoire, la fréquence où il y a échange de données entre les processeurs est une source de dégradation des performances parce qu'elle fait varier le temps d'exécution de la procédure parallèle. Il était tout naturel de penser que la fréquence d'interaction entre les procédures séquentielles se limiterait à avoir ce seul impact pour les stratégies de parallélisation avec recherches multiples coopérantes. Ce n'est qu'en 1989 que Tanasé [96] a démontré, suite à une étude exhaustive sur les algorithmes génétiques distribués, que la fréquence de migration des individus ne se limitait pas à une simple modification des temps de calcul mais qu'elle avait également un impact sur le comportement d'exploration de l'espace de solutions de ces méthodes de recherche. Son travail a démontré qu'une fréquence de migration trop faible ou trop élevée pouvait créer une détérioration de la qualité des solutions trouvées et non pas simplement une variation au niveau des temps de calcul de la procédure parallèle. Munetomo, Takai & Sato [75] ont aussi mis en lumière l'importance du problème de bien choisir l'intervalle d'interaction entre les sous-populations des algorithmes génétiques distribués.

Dans la présente section nous montrerons comment, pour la méthode tabou, la fréquence d'accès aux mémoires partagées affecte le comportement des procédures parallèles par recherches multiples coopérantes. Nous allons également introduire un critère permettant de définir la fréquence d'échange d'information entre les

procédures séquentielles.

Problématique relative aux algorithmes génétiques

La problématique est la suivante, en ce qui concerne la fréquence d'échange d'information entre les sous-populations: lorsque l'intervalle de migration est trop court, les sous-populations n'ont pas le temps de générer de nouveaux individus très performants qui pourraient se qualifier pour la migration. Il en résulte que les mêmes copies des individus les mieux adaptés émigrent souvent et dans un grand nombre de sous-populations, accroissant la pression sur la sélection et réduisant la diversité ce qui provoque des problèmes de convergence prématurée. Par contre, si la fréquence est trop basse, certaines sous-populations risquent d'évoluer longtemps avec un bassin d'individus incapables de générer des solutions de bonne qualité ce qui se traduit par de mauvaises performances générales du traitement génétique. Belding [12] a réalisé des tests avec des intervalles de migration de 5, 10, 20, 50, 100 et 500 générations, mais aucune conclusion générale n'a pu s'imposer. Levine [66] a utilisé trois intervalles différents de migration en les comparant avec le cas où, aucun échange d'information n'est effectué entre les sous-populations. Fait à remarquer, la plupart des expérimentations et des systèmes génétiques choisissent un intervalle fixe de migration pour la durée de la recherche, c'est le cas par exemple pour GENITOR de Starkweather, Witley & Mathias [89].

Problématique pour le recuit simulé et les méthodes de recherche spécialisées

Au niveau du recuit simulé, les études expérimentales de Graffigne [48] sur une procédure synchrone concluent que la fréquence d'échange d'information n'a pas d'effet sur la vitesse de convergence et la qualité des solution trouvées. Dans Hogg et Williams [55] concernant le problème de coloriage des noeuds d'un graphe où le partage d'information se fait par le biais de l'architecture blackboard, le choix de la fréquence d'accès est basé sur une probabilité ρ d'utiliser un message et

$1 - \rho$ d'utiliser l'information locale à chaque itération de la méthode. Si $\rho = 0$, cela correspond à une parallélisation par recherches multiples. Dans le cas du problème d'arithmétique cryptographique de Clearwater, Huberman & Hogg [25] où la procédure avec recherches multiples coopérantes est aussi basée sur une mémoire centrale, chaque procédure choisit d'abord une solution partielle dans la mémoire partagée à chaque itération. Les procédures utilisent leur information locale uniquement s'il n'y a pas de messages dans la mémoire centrale ou si le message choisit conduit à une solution déjà explorée par la procédure. Dans les deux cas, aucune motivation n'est donnée sur le choix de la fréquence d'accès et aucune étude n'a été faite sur l'utilisation d'autres fréquences d'accès à l'information partagée.

5.4.1 Échange d'information pour la méthode tabou

Ce qui caractérise une stratégie de parallélisation par recherches multiples coopérantes appliqué à la recherche tabou c'est le partage des nombreuses mémoires de cette méthode comme mode d'échange de l'information entre les procédures. Le partage de ces mémoires augmente le volume d'information dont dispose chaque règle de transition. Cependant ce partage peut aussi réduire la diversité des chemins d'exploration en alimentant chaque procédure séquentielle avec la même information. Il faut savoir que la diversité des chemins d'exploration provient essentiellement des stratégies d'exploration qui ne sont pas les mêmes entre les procédures séquentielles. Si l'historique de recherche des procédures séquentielles tend à s'uniformiser, cela réduit la capacité des stratégies d'exploration à maintenir cette diversité.

Illustration du processus d'échange d'information entre les procédures séquentielles

Les implantations que nous avons réalisées de stratégies de parallélisation par recherches multiples coopérantes sont basées sur l'utilisation d'une mémoire centrale. Pour fin d'illustration, nous ferons les hypothèses suivantes relativement au partage

d'information entre les procédures séquentielles. Soit ML_i le champs de l'historique de recherche de la procédure séquentielle p_i dont le contenu correspond à la meilleure solution à moyen terme de p_i. Le contenu du champs ML_i est utilisé par la procédure séquentielle p_i comme solution initiale de chaque phase d'intensification. Soit MC un champs de la mémoire centrale destiné à recevoir une solution trouvée par une procédure séquentielle p_i. Le critère de mise à jour de MC est défini de la manière suivante: $MC = ML_i$ si la solution de ML_i est meilleure que celle se trouvant déjà dans MC.

Exemple 5.1 La figure 5.2 illustre trois procédures séquentielles p_1, p_2 et p_3 qui partagent entre elles le contenu du champs ML par le biais de la location MC de la mémoire centrale. Ces procédures séquentielles s'échangent de l'information en écrivant le contenu du champs ML dans la mémoire centrale MC et en lisant le contenu de MC dans ML. Notons qu'une procédure séquentielle p_i peut écrire le contenu de ML_i dans MC uniquement si l'information en provenance de ML_i rencontre le critère de mise à jour de MC. Du côté des champs ML_i, leur mise à jour peut se faire de deux façons différentes: à partir des solutions générées par l'exploration de l'espace de solutions par la procédure séquentielle p_i ou à partir de la solution en provenance de la mémoire centrale. Le cycle de mise à jour de ML_i par une procédure séquentielle p_i est illustré par les boucles courtes à l'intérieur de chaque procédure (parfois selon un critère d'admissibilité de l'information). Par contre, le cycle de mise à jour du champs ML_i dû à un accès à la mémoire centrale est illustré par les boucles qui sortent des rectangles représentant les procédures séquentielles. Souvent, un cycle d'accès à la mémoire centrale par une procédure p_i va résulter en une tentative d'écriture dans MC suivi d'une lecture de MC. L'objet de la présente section porte sur la fréquence de ce cycle de mise à jour des mémoires locales suite à une lecture de la mémoire centrale.

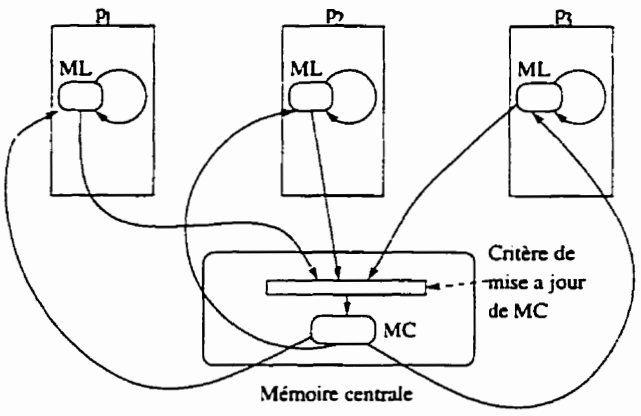


Figure 5.2 Cycle de mise à jour des mémoires partagées

Convergence prématurée créée par l'échange d'information

Lorsque pour un certain nombre de cycles, les champs partagés de plusieurs procédures séquentielles sont mis à jour avec un contenu de la mémoire centrale qui ne change pas, il en résulte une uniformisation des historiques de recherche qui tend à réduire la diversité des chemins d'exploration des procédures séquentielles. Dans le cas de l'exemple 5.1, une telle situation correspondrait à avoir le contenu du champs MC copié dans le champs ML de plusieurs procédures séquentielles. Cela reviendrait pour ces procédures séquentielles à exécuter leur phase d'intensification dans une même région de l'espace de solutions puisque la solution servant à débuter cette phase proviendrait de MC. Il en résulterait forcément une réduction de la diversité du parcours de l'espace de solutions. En fait, nous avons pu observer que, lorsque les procédures séquentielles ont un accès non restreint à l'information partagée, la procédure parallèle tabou peut éprouver de sérieux problèmes de convergence prématurée similaires à ceux des algorithmes génétiques distribués.

Ce comportement s'explique par le fait que lors de certaines phases d'une procédure parallèle avec recherches multiples coopérantes, le contenu des champs partagés des procédures séquentielles n'arrive pas à satisfaire le critère de mise à

jour de la mémoire centrale ou encore ce contenu provient d'une seule sous-région de l'espace de solutions. Conséquemment, la mémoire centrale n'est plus mise à jour ou encore les mises à jour portent sur de l'information provenant que d'une partie restreinte de l'espace de solutions. Dans ce cas, plus la fréquence d'accès à la mémoire centrale est grande, plus souvent les champs partagés seront ré-initialisés avec le même contenu dans un plus grand nombre de procédures séquentielles. Il en résulte que le cycle de mise à jour des mémoires locales par les procédures séquentielles (celui des boucles courtes dans la figure 5.2) tend à être le même à travers les procédures partageant la même information. Dans les cas extrêmes, on observe que toutes les procédures séquentielles sont figées. Les stratégies d'exploration ne disposent pas d'un nombre suffisant d'itérations entre chaque accès à la mémoire centrale pour extraire les procédures séquentielles de l'attraction de certaines régions de l'espace de solutions et pour pouvoir poursuivre une exploration efficace.

Contrôle de la fréquence de mise à jour des mémoires locales

On peut éviter que ce phénomène de convergence prématurée ne réduise l'efficacité d'une stratégie de parallélisation par recherches multiples coopérantes en contrôlant la fréquence de mise à jour des mémoires locales par la mémoire centrale. L'objectif essentiel visé de ce contrôle, est de permettre un équilibre entre l'information partagée et les stratégies d'exploration sur le déroulement de l'exploration de l'espace de solutions. Ce contrôle peut s'exécuter à l'aide de critères spéciaux de mise à jour des mémoires locales. Ces critères peuvent être implantés pour chaque règle de transition, ou encore ils peuvent être indépendants et contrôler tout un ensemble de règles de transition. De plus, les paramètres pour ces critères peuvent être, soit prédéfinis et fixés pour la durée de la recherche, ou encore varier dynamiquement selon, par exemple, la capacité d'une stratégie de parallélisation par recherches multiples coopérantes à échapper à certains attracteurs locaux.

Critère 3 *Soit ML_i un champs de l'historique de recherche de la procédure séquentielle p_i, et MC un champs de la mémoire centrale. Le critère de mise à jour du champs ML_i par le contenu de MC peut être exprimé par la relation suivante pour chaque procédure p_i:*

$$ML_i = \begin{cases} MC & si\ (\frac{i}{\alpha} * MC) < ML_i \\ ML_i & sinon. \end{cases} \tag{5.2}$$

où la constante α est fonction de l'application et i est l'indice de la procédure p_i.

La relation (5.2) signifie qu'une procédure séquentielle p_i peut admettre une information en provenance d'une autre procédure (MC), si le bénéfice pour la procédure p_i et la stratégie de parallélisation par recherches multiples coopérantes vaut le changement dans le parcours du chemin d'exploration de la procédure p_i. Des critères de mise à jour comme celui-ci sont faciles à implanter, et ont prouvé leur efficacité dans nos tests pour établir un équilibre entre l'influence des stratégies d'exploration et l'information partagée sur le parcours de l'espace de solutions des procédure séquentielles.

Enfin, soulignons que la fréquence d'échange d'information entre les procédures séquentielles peut être utilisée pour orienter le comportement global d'une stratégie de parallélisation par recherches multiples coopérantes. En effet, la modification dynamique des conditions d'accès à l'information partagée peut induire des comportements collectifs spécifiques parmi les procédures séquentielles. Par exemple, si on cherche à obtenir une convergence rapide vers un optimum local, l'accès à l'information partagée peut être élargi. Ceci va conduire plusieurs procédures séquentielles vers la même région de l'espace de solutions, et ainsi simuler une séquence d'intensification. Si au contraire une descente plus lente est souhaitée, le critère d'accès doit être resserré. Il est aussi possible d'avoir un mouvement collectif de diversification en empêchant tout accès à l'information partagée, ce qui retournera le contrôle complet de l'exploration aux stratégies d'exploration. Si les

restrictions à l'information partagée sont changées en alternance, on peut conduire l'exploration de l'espace de solutions dans des phases alternatives d'intensification et de diversification collective.

5.5 Entre quelles procédures l'information doit être échangée

L'objectif de la présente section est de définir des critères permettant de choisir entre quelles procédures séquentielles chaque échange d'information doit se faire. Dans la littérature portant sur les stratégies de parallélisation par recherches multiples coopérantes, on nous réfère à ce problème comme étant celui de la définition d'une structure de voisinage ou encore d'une *structure d'hyper voisinage* [99, 100] entre les procédures séquentielles.

Problématique reliée à la structure de voisinage pour les algorithmes génétiques

L'article de Mühlenbein [72] constitue une bonne introduction sur la façon dont ce problème de la définition d'une structure de voisinage a été abordé par la recherche portant sur les algorithmes génétiques distribués. La division de la population en sous-populations relativement isolées est une manière efficace de faire obstacle au phénomène de la convergence prématurée, l'isolement relatif de chaque sous-population étant plus propice à maintenir une plus grande diversité de la population dans son ensemble. Le rôle de la structure de voisinage est de définir entre quelles sous-populations l'échange d'information peut se faire directement.

La structure de voisinage n'est pas neutre relativement à l'impact quelle peut avoir sur la diversité de la population, car elle influence la vitesse de propagation des individus les mieux adaptés à travers les sous-populations. Une structure de voisinage avec un degré élevé de connectivité entre les sous-populations va permettre

aux bonnes solutions de se répandre plus rapidement et de prendre le contrôle de l'ensemble de la population. Au contraire, une faible connectivité va ralentir la propagation des bonnes solutions et favoriser l'apparition de solutions différentes.

On constate cependant dans la littérature sur les algorithmes génétiques, que la structure de voisinage emprunte souvent celle du réseau d'interconnexion de l'ordinateur parallèle utilisé pour les tests. C'est le cas pour les travaux suivants: [9, 26, 46, 85, 87, 96]. Quelques recherches sur les algorithmes génétiques à grain fin [9, 20, 87] définissent la structure de voisinage en fonction de son diamètre, c'est-à-dire le nombre de voisins de chaque sous-population. Les résultats de ces études semblent favoriser un faible diamètre où les voisinages s'entrecroisent. Une autre catégorie de structure de voisinage consiste, pour chaque message, à choisir par un tirage aléatoire, deux sous-populations entre lesquelles l'échange d'information s'exécute. Belding [12] a effectué plusieurs tests sur ces structures dynamiques de voisinage.

Recuit simulé et méthodes de recherche spécialisées

Il n'existe pas d'études comparatives au niveau du recuit simulé quant à l'impact de la structure de voisinage sur les performances de cette méthode. L'étude de Graffigne [48] se base sur une structure de voisinage qui force un ordonnancement séquentiel des procédures. Chaque procédure envoie à son voisin de droite la meilleure des deux solutions suivantes: celle reçue du voisin de gauche ou sa propre solution. La structure de voisinage utilisée pour le recuit simulé asynchrone [51] est essentiellement la même que celle de la procédure parallèle collégiale asynchrone du chapitre 4. Enfin, Clearwater, Hogg et Huberman [24] analysent le problème de la structure de voisinage en fonction de son impact sur la congestion du réseau physique de communication et selon une organisation hiérarchique de l'architecture blackboard. Dans leur étude, ils ont remarqué que de meilleurs résultats sont obtenus lorsque des liens de voisinage sont créés dynamiquement entre les procédures séquentielles (similaire

à la structure de voisinage de la procédure parallèle collégiale asynchrone).

Structure de voisinage et la méthode tabou

Comme mentionné à la section 5.4, l'information échangée entre les procédures séquentielles peut avoir un effet néfaste sur l'efficacité des stratégies d'exploration. Tout comme pour les algorithmes génétiques, la structure de voisinage des recherches multiples coopérantes de la méthode tabou réduit ou accélère la propagation de l'information partagée. La problématique est donc assez similaire à celle des algorithmes génétiques. Nous allons d'abord explorer de manière schématique comment l'information partagés et les stratégies d'exploration s'influencent mutuellement au niveau des procédures parallèles p-KS (voir section 4.7.4). Par la suite nous pourrons montrer comment la structure de voisinage affecte les stratégies de parallélisation.

5.5.1 Information partagée et stratégies d'exploration des procédures p-KS

La structure de voisinage des procédures parallèles p-KS correspond à une connexité complète entre les procédures séquentielles et l'échange de messages s'effectue en mode synchrone. Lorsqu'une procédure séquentielle p_i atteint un point d'interaction, elle entre dans un cycle d'accès à la mémoire centrale. Ce cycle comprend une phase d'écriture si l'information en provenance de p_i rencontre le critère de mise à jour de la mémoire centrale, suivi d'une phase d'attente que toutes les procédures séquentielles aient signalé leur arrivée au point d'interaction, et finalement d'une phase de lecture du contenu de la mémoire centrale. Ce type d'interaction entre les procédures séquentielles est équivalent à celui où chaque procédure p_i fait une diffusion (broadcast) du même message à toutes les autres procédures séquentielles suivi d'une étape où chaque procédure séquentielle choisit un message parmi les $p - 1$ messages reçus (la littérature portant sur les algorithmes génétiques et le recuit simulé associe généralement le traitement synchrone à un accès, par toutes

les procédures, à une même information partagée).

Exemple 5.2 Soit MC et ML_i les variables définies à la section 5.4. Soit δ le nombre d'itérations exécutées par chaque procédure séquentielle entre deux points d'interaction et $m = \frac{t}{\delta}$ le nombre de points d'interaction. Soit $s_0, s_1, \ldots, s_{p-1}$ l'ensemble de stratégies d'exploration de la procédure p-KS et l_i une structure de données locale représentant la liste tabou à court terme de la procédure séquentielle p_i. Le critère de mise à jour du champs ML_i pour chaque procédure est contraint par la relation (5.2). Lorsque les procédures arrivent à un point d'interaction k, chaque procédure p_i écrit le contenu de ML_i dans la mémoire centrale MC si $ML < MC$ (critère de mise à jour de la mémoire centrale). Avant de poursuivre leur exécution, chaque procédure p_i écrit le contenu de la mémoire centrale MC dans le champs partagé ML_i si le critère de mise à jour 5.2 de la mémoire locale est respecté. Le contenu du champs ML_i est utilisé comme solution initiale des phases de diversification des procédures séquentielles.

Avant l'arrivée au premier point d'interaction, l'exploration de l'espace de solutions par une procédure p-KS s'exécute de la même manière qu'une procédure parallèle avec recherches multiples. Entre les points d'interaction 1 et 2, dépendant des paramètres locaux de recherche qui contrôlent la phase de diversification, un sous-ensemble D, des procédures séquentielles de la procédure parallèle par recherches multiples coopérantes va entrer dans une phase de diversification. En fonction de l'équation (5.2), pour un sous-ensemble $U \subseteq D$, l'exploration de l'espace de solutions va redémarrer à partir de la même solution, c'est-à-dire celle en provenance de la mémoire centrale MC. Pour les autres procédures $(D \setminus U)$, l'exploration de l'espace de solutions se fera à partir de ML_i, la meilleure solution locale de la procédure p_i. Pour $p_i \in U$, l'itération tabou qui suit la phase de diversification est identique pour chaque procédure séquentielle:

1. $ML_i = MC$,

2. La liste tabou à court terme l_i va enregistrer la solution en provenance de MC comme étant une solution tabou,

3. Les paramètres locaux de recherche vont tous être appliqués à la même solution MC.

Lorsque la recherche progresse vers le deuxième point d'interaction, les chemins d'exploration des procédures séquentielles dans U vont prendre des directions différentes sous l'influence des stratégies d'exploration de chaque procédure séquentielle et du contenu des mémoires locales qui n'ont pas été affectées par la solution MC. Pour les procédure $p_i \in T \setminus U$, ils vont continuer de se comporter comme des procédures séquentielles indépendantes au moins jusqu'au prochain point d'interaction. Arrivé au deuxième point d'interaction, le même processus que celui que nous venons de décrire après l'arrivée au point d'interaction se répète.

Proposition 5.1 *Le parcours de l'espace de solutions de chaque procédure séquentielle p_i d'une procédure parallèle p-KS est déterminé par l'interaction entre la stratégie d'exploration s_i et l'information partagée.*

Preuve: Il est clair qu'à chaque point d'interaction, le contenu de MC est déterminé par celui des champs ML_i, ce qui implique que l'information partagée est fonction des stratégies d'exploration. Donc s'il s'agit d'un problème de minimisation, $MC = \min\{ML_0, ML_1, \ldots, ML_{p-1}\}$. D'un autre côté, avant le début de la phase qui suit un point d'interaction, pour les procédures séquentielles p_i où le critère de mise à jour 5.2 est satisfait, $ML_i = MC = \min\{ML_0, ML_1, \ldots, ML_{p-1}\}$. Conséquemment, pour ces procédures p_i, le contenu du champs ML_i est différent après le point d'interaction de ce qu'il était avant l'arrivée au point d'interaction. Pour l'ensemble U des procédures séquentielles où le contenu de $ML_i = MC$ au moment d'une diversification, la solution initiale de la diversification sera différente de ce qu'elle aurait été sans le partage d'information, ce qui affecte le parcours de

l'espace de solutions de la procédure séquentielle p_i. Mais le processus de diversification fait partie de la stratégie d'exploration, on peut donc dire que le comportement de la stratégie d'exploration est modifié par l'introduction d'information partagée dans l'historique de recherche des procédures séquentielles.

Remarquons que pour les procédures séquentielles de l'ensemble U, le parcours modifié de ces procédures va affecter le contenu de leur champs ML_i, ce qui ultimement pourra affecter la valeur du champs MC (c'est-à-dire que le contenu de MC au point d'interaction $k+1$ sera différent de ce qu'il aurait été sans les modifications aux parcours de certaines procédures provoquées par la valeur de MC au point d'interaction k), etc. Ce jeux subtile entre l'information partagée et les stratégies d'exploration crée une pression vers une convergence plus rapide de la recherche globale plutôt que de stimuler l'exploration de nouvelles régions. S'il y a une trop grande uniformisation du contenu des champs partagés, l'effet peut être très dommageable à une stratégie de parallélisation par recherches multiples coopérantes.

5.5.2 Comparaison de différentes structures de voisinage

Comme nous l'avons montré à la section 4.9 du chapitre 4, l'accès à la mémoire partagée pour les procédures parallèles p-KC et collégiales asynchrones est déclenché par une logique interne de chaque procédure séquentielle sans référence a un critère global (horloge, itérations, etc.). La structure de voisinage qui en résulte n'est donc pas fixe. Elle se constitue de façon dynamique par le biais de la lecture de la mémoire centrale par une ou plusieurs procédures séquentielles entre deux modifications de cette mémoire centrale. L'accès à la mémoire centrale par chaque procédure s'effectue sur une base irrégulière et sans synchronisation avec les autres procédures. Il en résulte que le contenu des champs partagés de l'historique de recherche est beaucoup moins uniforme que pour les procédures parallèles p-KS.

Par contre, lors de l'échange d'information d'une procédure parallèle p-KS, toutes les procédures écrivent dans la mémoire centrale à chaque point d'interaction

avant de lire *le même contenu de MC dans leur mémoires locales.* Conséquemment, l'information relative à une seule sous-région de l'espace de solutions sera partagée entre deux points d'interaction, la diversité de l'information partagée est minimale, et il y a une forte uniformisation du contenu de mémoires locales. Ce n'est pas le cas pour les procédures parallèles p-KC et collégiales asynchrones, entre deux mises à jour de la mémoire centrale, un nombre limité de procédures séquentielles accèdent cette mémoire centrale. Conséquemment, l'information en provenance d'une sous-région n'est pas transmise à un grand nombre de procédures séquentielles et plusieurs sous-régions peuvent partager leur information avec différentes procédures séquentielles. Il y a donc une plus grande diversité dans l'information qui est partagée, et le contenu des mémoires locales tend à être moins uniforme. Sous l'effet de l'interaction entre l'information partagée et les stratégies d'exploration, les procédures parallèles p-KS tendent à converger vers une même région de l'espace de solutions, ce phénomène est moins important pour les deux autres types de procédures parallèles.

On s'attend donc a un comportement général différent de la méthode tabou par recherche multiples coopérantes dans ces deux cas extrêmes de modes d'échange d'information, les résultats des tests du chapitre 4 le démontrent bien. Étant donné que le mode asynchrone a produit de meilleurs résultats, nous avons voulu tester une hypothèse selon laquelle le plus faible diamètre du voisinage des procédures parallèles collégiales asynchrones serait un facteur déterminant pour ces meilleures performances. Un diamètre plus faible signifie une plus faible connexité entre les procédures séquentielles ce qui, comme pour les algorithmes génétiques, implique une diffusion plus lente du contenu des champs partagés et donc une plus grande diversité de l'exploration.

Nous avons donc défini une structure de voisinage où le partage direct d'une même information est limité à des sous-ensembles de faible cardinalité de procédures séquentielles. Les sous-ensembles se recoupent entre eux pour permettre à une information de circuler entre les sous-ensemble mais à une vitesse beaucoup moindre.

Globalement, cette stratégie de parallélisation par recherches multiples coopérantes est synchrone, l'échange d'information se fait à intervalle régulier à des points d'interaction. Cependant le partage direct d'information ne se fait qu'entre les procédures séquentielles appartenant au même voisinage.

Définition 5.1 *Le voisinage $\mathcal{N}_{p_i}$ d'une procédure séquentielle p_i correspond à l'ensemble des procédures séquentielles p_j dont le champs ML_j est partagé avec celui de ML_i.*

Chaque procédure séquentielle ne possède qu'un seul voisinage mais elle peut être impliquée dans le voisinage de plusieurs autres procédures séquentielles.

Définition 5.2 *Une procédure p_i ne possède qu'un seul voisinage en ce sens qu'au moment d'échanger de l'information à un point d'interaction, le contenu de $ML_i = ML_j$ ssi $ML_j < ML_l \; \forall \; p_l \in \mathcal{N}_{p_i}$ et $ML_j < ML_i$.*

Définition 5.3 *Une procédure p_i est impliquée dans plusieurs voisinage en ce sens que $p_i \in \mathcal{N}_{p_m}$ et conséquemment le champs ML_i peut être partagé avec le champs ML_m avant que le contenu de ML_i ne devienne égale à celui de ML_j à un point d'interaction.*

Puisque les voisinages se chevauchent les uns les autres, cette nouvelle structure d'échange permet la propagation d'une même information vers toutes les procédures.

Définition 5.4 *Si au point d'interaction k, $ML_i = ML_j$, au point d'interaction $k+1$, les voisinages $\mathcal{N}_{p_j}$ qui ont une intersection non vide avec la procédure séquentielle p_i seront tel que si $ML_i < ML_l \; \forall \; p_l \in \mathcal{N}_{p_j}$, alors $ML_j = ML_i$.*

L'information partagée doit traverser plusieurs couches de voisinages avant qu'elle puisse atteindre toutes les procédures, bien souvent elle ne les atteindra pas toutes.

Les performances obtenues en utilisant cette structure de voisinage ont été meilleures que celles de la procédure parallèle collégiale asynchrone, la solution

optimale fut trouvée pour les problèmes dans le tableau 5.1. Les conclusions qui semblent se dessiner à partir de ces tests et des expériences faites au niveau des algorithmes génétiques nous conduisent à énoncer le critère suivant pour la structure de voisinage:

Critère 4 *Un voisinage de faible diamètre devrait constituer un premier choix pour l'implantation de stratégies de parallélisation par recherches multiples coopérantes avec la méthode tabou.*

Cependant, le caractère informel de l'argumentation développée dans la présente section ne nous permet pas de bien mesurer toute la complexité de la dynamique d'interaction provoquée par les différentes structures de voisinage entre les procédures séquentielles. Le modèle introduit au chapitre 7 sur la dynamique d'interaction entre les stratégies d'exploration nous permettra de pousser l'analyse sur la structure de voisinage un peu plus en profondeur.

5.6 Conclusion

La conception d'une procédure parallèle basée sur une stratégie de parallélisation obligatoire ne demande pas que soit spécifié l'échange d'information entre les tâches parallèles. Cet échange est défini de manière implicite par le graphe de flot de données de l'algorithme séquentiel, la granularité de la parallélisation et le type de décomposition. Ce n'est pas le cas pour les stratégies de parallélisation par recherches multiples coopérantes puisque ces stratégies ne sont pas basées sur la décomposition d'une procédure séquentielle.

Le concepteur d'une procédure parallèle par recherches multiples coopérantes doit donc spécifier lui-même comment l'échange d'information doit se faire entre les procédures séquentielles. Dans ce chapitre, nous avons introduit un cadre général en vue de guider cette phase de conception d'échange d'information. D'un point de vue purement de conception, l'échange d'information entre les procédures séquentielles

des recherches multiples coopérantes consiste à spécifier de manière explicite, ce qui est fait de manière implicite au niveau du traitement parallèle obligatoire (c'est-à-dire quelles donnés sont échangées, entre quelles procédures et à quel moment). C'est la raison pour laquelle ce cadre général s'inspire de la notion de couplage. Le couplage entre les procédures séquentielles de la méthode tabou spécifie les mémoires locales impliquées dans le partage de l'information, les critères de mise à jour des mémoires locales qui contrôlent la fréquence d'échange d'information, et la structure de voisinage qui contrôle la vitesse de propagation de l'information.

Cependant, les critères spécifiques du couplage sont basés sur des ensembles de considérations découlant de la problématique du partage d'information d'une stratégie de parallélisation par recherches multiples coopérantes. C'est le cas, par exemple, concernant l'impact de l'échange d'information entre les procédures séquentielles sur le comportement de la méthode tabou. Nous savons que la méthode tabou utilise les connaissances acquises par l'exploration de l'espace de solutions à travers le processus de recherche. Si nous arrêtons l'exécution d'une procédure tabou p_i et nous remplaçons les connaissances acquises de p_i par celles d'une autre procédure p_j, il est clair que le parcours de l'espace de solutions par la procédure p_i sera affecté par cette substitution. L'échange d'information d'une procédure par recherches multiples coopérantes a un effet similaire sur le comportement des procédures séquentielles, cette échange modifie constamment les connaissances acquises de chaque procédure séquentielle. Les exemples 5.1 et 5.2 montrent que le partage d'information entre les procédures séquentielles entraîne cependant un effet secondaire, celui d'uniformiser les connaissances acquises. Nous avons vu à travers les sections 5.4 et 5.5 portant sur la fréquence d'échange d'information et la structure de voisinage, que l'uniformisation des connaissances acquises peut réduire l'efficacité des stratégies d'exploration. Il est donc apparu important qu'un des objectifs de la phase de conception de l'échange d'information entre les procédures séquentielles soit d'empêcher qu'il y ait une trop grande uniformisation des connaissances acquises.

Dans les deux cas, les critères énoncés visent à maîtriser ce phénomène d'uniformisation en contrôlant la vitesse de propagation de l'information partagée.

Chapitre 6
Dynamiques de réactions en chaîne

6.1 Introduction

Au chapitre précédent, l'uniformisation des connaissances acquises au niveau de plusieurs procédures séquentielles est apparue comme un effet secondaire important de l'échange d'information entre les procédures séquentielles. Dans ce chapitre nous allons aborder l'étude d'une deuxième catégorie d'effets secondaires résultant de la coopération entre les procédures séquentielles. Nous avons signalé pour la première fois l'existence de cette catégorie d'effets secondaires par la proposition 5.1 du chapitre précédent, montrant la dépendance du parcours de l'espace de solutions sur plusieurs niveaux d'interaction entre l'information partagée et les stratégies d'exploration.

Intuitivement, il est facile de comprendre que le parcours de l'espace de solutions d'une procédure de recherche tabou change si on modifie l'ensemble ou un sous-ensemble des connaissances acquises par cette procédure. Cependant, si l'on réfléchit bien, il nous faut aussi admettre que le changement au parcours de l'espace de solutions d'une procédure séquentielle (quelque soit l'origine de ce changement), provoque un changement des connaissances acquises par la procédure puisque ces connaissances proviennent de l'exploration de l'espace de solutions. On peut ensuite reprendre l'argument selon lequel les changements aux connaissances acquises modifient le parcours de l'espace de solutions, et montrer qu'un changement aux connaissances acquises provoqué par un changement de parcours entraîne ultérieurement un nouveau changement de parcours, etc. Une modification aux connaissances acquises ne se traduit donc pas seulement par un changement immédiat

du parcours de l'espace de solutions, elle prend également la forme d'une chaîne de réactions à long terme entre les connaissances acquises et le chemin d'exploration d'une procédure de recherche tabou.

Ce phénomène se traduit par un autre type de comportement global lorsqu'on l'observe à l'échelle de l'ensemble des procédures séquentielles. En effet, si le changement des connaissances acquises d'une procédure p_i peut par réaction changer le parcours de p_i, lorsqu'il y a échange d'information entre les procédures séquentielles, le même changement à p_i peut par réaction provoquer un changement de parcours d'une autre procédure p_j.

Définition 6.1 *Soit l'***ensemble global des connaissances acquises** *correspondant à la somme des connaissances acquises par chaque procédure séquentielle.*

Tout changement dans les connaissances acquises par une procédure p_i constitue un changement dans l'ensemble global des connaissances acquises. Puisqu'il y a partage des connaissances acquises entre les procédures séquentielles, un changement dans l'ensemble global des connaissances acquises pourra se répercuter par réaction sur n'importe quelle autre procédure séquentielle p_j, lequel changement va affecter le parcours de p_j, qui va encore modifier l'ensemble des connaissances acquises, etc. Le phénomène de réactions en chaîne qui se manifeste dans le temps au niveau d'une procédure séquentielle de la méthode tabou s'étend donc dans l'espace et dans le temps en ce qui concerne les procédures par recherches multiples coopérantes.

Dans ce chapitre nous étudierons ce phénomène global des recherches multiples coopérantes où un changement dans une procédure séquentielle se traduit par une série de perturbations au niveau d'un certain nombre des autres procédures séquentielles de la procédure parallèle. Nous identifierons par *comportement dynamique de réactions en chaîne* ce comportement global des procédures par recherches multiples coopérantes. À cette étape il nous être très précis, nous ne chercherons pas, contrairement au chapitre précédent, à comprendre comment un contenu

particulier d'une mémoire partagée se propage à travers les procédures séquentielles, ce qui effectivement donne naissance au phénomène d'uniformisation des connaissances acquises. Nous chercherons plutôt à comprendre comment les effets du comportement d'une procédure séquentielle sur l'ensemble global des connaissances acquises se propage sur le comportement des autres procédures séquentielles, sachant que cette propagation peut s'effectuer à travers l'échange de messages fort différents les uns des autres entre les procédures séquentielles.

On peut voir l'ensemble global des connaissances acquises comme un medium par lequel les procédures séquentielles se transmettent des signaux qui affectent leur comportement d'exploration de l'espace de solutions. En ce sens, nous ferons plusieurs expériences visant à isoler certaines procédures de ce réseau de signaux et nous étudierons les changements au comportement dynamique de réactions en chaîne qu'entraînent ces modifications. Nous pourrons ainsi montrer l'existence de deux patrons différents de réactions en chaîne entre les procédures séquentielles.

Le chapitre est organisé de la manière suivante. Dans la prochaine section, nous montrerons comment l'échange d'information des recherches multiples coopérantes interfère avec le déroulement normal de la méthode tabou. Dans la section suivante, nous montrerons l'existence de chaînes de réactions au comportement des procédures séquentielles au niveau des recherches multiples coopérantes. Nous identifierons deux patrons d'interactions entre les procédures séquentielles provoquées par l'échange d'information. Enfin, nous montrerons l'impact considérable que ces dynamiques ont sur les recherches multiples coopérantes et l'inter-dépendance complexe qu'elles supposent entre le comportement de chaque procédure séquentielle. Nous montrerons que ces dynamiques laissent supposer une dépendance importante du comportement d'exploration de l'espace de solutions d'une procédure par recherches multiples coopérantes sur le nombre de procédures séquentielles, la structure de voisinage et l'information échangée.

6.2 Interférence de l'échange d'information sur la méthode tabou

Dans la présente section, nous montrerons comment les stratégies d'exploration des procédures séquentielles sont affectées lorsqu'elles s'exécutent dans un contexte d'échange d'information. Par souci de clarté et sans perte de généralité, l'argumentation du présent chapitre se limite aux recherches multiples coopérantes où la phase de conception de l'échange d'information est développée à partir des mémoires existantes de la méthode tabou (voir la section 5.3).

Soit $s_0, s_1, \ldots, s_{p-1}$ les p stratégies d'exploration d'une procédure parallèle basée sur une décomposition fonctionnelle du traitement spéculatif (p-KC, p-KS, collégiales asynchrones ou recherches multiples). Dans le contexte de ce chapitre, lorsque $i \neq j$, la stratégie d'exploration s_i diffère de la stratégie d'exploration s_j. Rappelons que des stratégies d'exploration différentes, pour une procédure parallèle basée sur une décomposition fonctionnelle du traitement spéculatif, sont obtenues à partir des stratégies de différentiation MPDS, SPDS et MPSS.

Définition 6.2 *Nous dirons que l'ensemble formé par les paramètres de recherche, la stratégie de différentiation et la solution initiale constitue la stratégie d'exploration "virtuelle" d'une méthode tabou. La stratégie d'exploration virtuelle combinée avec l'historique de recherche pour une instance d'un problème donné d'une procédure tabou séquentielle constitue la stratégie d'exploration "actuelle" d'une méthode tabou.*

Ce sont les données qui se trouvent dans les diverses mémoires d'une procédure tabou, c'est-à-dire l'historique de recherche, qui forment les connaissances acquises d'une procédure de recherche tabou. Ces données lorsque combinées avec une stratégie d'exploration virtuelle, déterminent, pour un problème donné, la trajectoire d'exploration de l'espace de solutions d'une procédure de recherche tabou séquentielle. Le choix, à chaque itération d'une solution dans le voisinage de la solution courante, dépend de l'information contenue dans l'historique de recherche.

Une stratégie d'exploration devient actuelle lorsqu'elle accède à ces données pour définir le parcours précis de l'espace de solutions effectué par une procédure de recherche tabou séquentielle.

Comme le mentionne Verhoeven [100], chaque règle de transition d'une méthode de recherche impose un graphe orienté sur l'espace de solutions. Les noeuds du *graphe de transitions* correspondent aux solutions et il y a un arc $(a \rightarrow b)$ dans ce graphe si la solution b est un voisin de la solution a. Un chemin d'exploration est un parcours dans ce graphe orienté où à chaque itération k, la solution courante est une solution dans le voisinage de la solution courante à l'itération $k-1$. D'après Verhoeven, les performances d'une méthode de recherche séquentielle dépendent à la fois de la structure du graphe de transitions qui elle-même dépend de la règle de transition utilisée et de la façon dont le graphe de transitions est parcouru qui à son tour dépend de la stratégie d'exploration (le type de recherche locale dans [100]).

Nous avons pu observer au chapitre 4 que pour un même ensemble de stratégies d'exploration, le parcours du graphe de transitions de chaque procédure séquentielle et les performances des procédures parallèles peuvent varier de manière importante d'une stratégie de parallélisation à l'autre. Cependant, étant donné que l'ensemble des stratégies d'exploration est le même, les variations de performances ne peuvent provenir que de l'actualisation des stratégies d'exploration. Les procédures par recherches multiples coopérantes sont conçues effectivement à partir d'un changement à la manière d'actualiser les stratégies d'explorations des procédures séquentielles des recherches multiples. Contrairement aux procédures parallèles par recherche multiples, l'actualisation d'une stratégie d'exploration s_i des recherches multiples coopérantes peut s'effectuer à partir d'information en provenance de l'historique de recherche de procédures séquentielles autres que p_i (par échange d'information). Cette façon d'actualiser des stratégies d'exploration vise à améliorer les performances des recherches multiples coopérantes, en accroissant le degré d'information dont dispose les stratégies d'exploration pour guider le parcours du

graphe de transitions accompli par les procédures séquentielles.

L'échange d'information entre des procédures séquentielles de la méthode tabou pose cependant un problème similaire à celui que nous avons pu observer au niveau de la parallélisation des méthodes de recherche par énumération implicite (voir la section 3.3.3). Une procédure séquentielle d'énumération implicite et sa version parallèle ne visitent pas nécessairement le même ensemble de solutions. La décomposition de l'espace de solutions de même que l'échange d'information entre les tâches parallèles font que l'information heuristique diffère entre la procédure parallèle et la procédure séquentielle lors la génération des noeuds dans l'arbre d'exploration. L'information contenue dans l'historique de recherche d'une procédure tabou est de nature heuristique, comme celle qui guide une procédure par énumération implicite. Un changement au niveau de cette information heuristique entraîne parfois la stratégie d'exploration à faire des choix différents dans le voisinage de la solution courante, ce qui évidemment correspond à une modification du parcours du graphe de transitions en comparaison au parcours effectué par une procédure de recherche tabou séquentielle.

Définition 6.3 *Soit h_{ik} le contenu des mémoires de l'historique de recherche de la procédure séquentielle p_i à l'itération k, et $\bar{h}_{ik}$ le contenu des mémoires du même historique de recherche lorsque de l'information d'historiques de recherche en provenance d'autres procédures séquentielles est prise en compte.*

Définition 6.4 *Soit x' la solution choisie dans le voisinage de la solution courante x par la stratégie d'exploration actuelle s_i à l'itération k à partir de l'historique de recherche h_{ik} et $\bar{x}'$ le choix effectué par la stratégie d'exploration actuelle s_i à l'itération k à partir de $\bar{h}_{ik}$.*

L'actualisation des stratégies d'exploration à partir de l'historique de recherche $\bar{h}_{ik}$ équivaut à modifier les connaissances acquises de la procédure p_i. Il s'agit d'une première source d'interférence des recherches multiples coopérantes sur le

déroulement normale de la méthode tabou. Lorsque pour une procédure séquentielle p_i, $\bar{x}' \neq x'$ à l'itération k, le parcours du graphe de transitions exécuté par p_i se trouve changé à cause du mode d'actualisation de la stratégie d'exploration s_i en interaction avec les autres procédures séquentielles. Le parcours de p_i est modifié suite à un changement au niveau des connaissances acquises de p_i.

Chaque actualisation à partir de $\bar{h}_{ik}$ est le point d'origine d'une réaction en chaîne à l'intérieur de la procédure p_i. En effet, à moyen terme, l'historique de recherche d'une procédure séquentielle est affecté par des choix de solutions $\bar{x}'$ provenant de $\bar{h}_{ik}$. Les attributs des solutions $\bar{x}'$ finissent par laisser des traces dans les mémoires de la méthode tabou différentes de celles qu'auraient laissées les solutions x' (par exemple la mémoire à court terme enregistre $\bar{x}'$ comme solution tabou et non pas x'). À long terme, cette information emmagasinée dans les mémoires de l'historique de recherche affecte à son tour, de manière substantielle, le parcours du graphe de transitions exécuté par une procédure de recherche tabou faisant partie d'une procédure par recherches multiples coopérantes. Même lorsque l'actualisation s'effectue à partir de h_{ik}, le choix de x' n'est pas le même puisque h_{ik} n'est plus le même historique de recherche que celui d'une procédure séquentielle dans le contexte des recherches multiples.

Donc, si on se limite à expliquer le comportement des procédures parallèles par recherches multiples coopérantes strictement à partir de celui des procédures séquentielles, on constate qu'il existe au moins deux sources potentielles d'interférence des recherches multiples coopérantes avec la méthode tabou pouvant changer le comportement d'exploration des procédures séquentielles: l'actualisation des stratégies d'exploration s_i à partir d'information sur l'espace de solutions ne provenant pas du parcours exécuté par la procédure séquentielle p_i, et l'altération des mémoires de la méthode tabou consécutive à l'inclusion d'information exogènes. L'actualisation des stratégies d'exploration à partir de $\bar{h}_{ik}$ interfère directement avec la méthode tabou. Elle est l'origine d'un phénomène de réaction en chaîne entre le parcours

de l'espace de solution et les connaissances acquises, phénomène qui à son tour interfère avec l'exécution normale de la stratégie d'exploration de la procédure de recherche tabou. Fait important à noter, les réactions en chaîne au niveau de chaque procédure séquentielle changent de manière importante les connaissances acquises par la procédure, ce qui se traduit évidemment par un impact important sur l'ensemble global des connaissances acquises.

6.3 Comportement dynamique de réactions en chaîne

À la section précédente, nous avons montré comment les stratégies d'exploration sont affectées par leur interaction avec les autres procédures séquentielles du traitement par recherches multiples coopérantes. Dans cette section, nous étudierons la dynamique qui fait que le comportement d'une procédure séquentielle se propage et affecte le comportement d'autres procédures séquentielles des recherches multiples coopérantes. Notre approche sera faite dans une perspective tout à fait différente de celle de la section précédente, car elle fera porter le poids des modifications au parcours de l'espace de solutions sur l'interaction entre les procédures séquentielles et non pas sur des altérations de la méthode tabou. Un des résultats important de la présente section sera de montrer l'inter-dépendance complexe existant entre les stratégies d'exploration des procédures séquentielles du traitement par recherches multiples coopérantes.

Mais, tout d'abord, nous allons introduire une procédure par recherches multiples coopérantes conçue dans le but de faire ressortir la dynamique d'interaction entre les stratégies d'exploration. Nous utiliserons cette procédure pour valider de façon expérimentale certaines hypothèses de cette section.

6.3.1 Procédure par recherches multiples coopérantes

Soit $T3$ une procédure par recherches multiples coopérantes pour le problème d'optimisation de la section 4.5. L'espace de configurations est donné par $\mathcal{C} \subseteq B^n$ et

inclut des solutions non réalisables pour le problème d'optimisation. Les règles de transition de chaque procédure séquentielle sont les mêmes que celles de la méthode tabou de la section 4.5. MPSS sera la stratégie de différentiation de cette procédure parallèle. Nous supposerons que l'exploration de l'espace de solutions est effectuée par p procédures séquentielles en t itérations.

Le couplage entre les procédures séquentielles est défini de la manière suivante: M est l'ensemble des messages pouvant être échangés entre les procédures séquentielles, $M \subseteq B^n$, $|M| \leq 2^n - 1$, $m_k \in M$, m_0 est un message spécial correspondant au message vide ($\emptyset$), c'est-à-dire pas de message. L'ensemble M est un sous-ensemble de C et correspond aux contextes des solutions réalisables du problème d'optimisation combinatoire. L'échange d'information de la procédure parallèle $T3$ se fait de manière synchrone. Nous ferons l'hypothèse que l'intervalle d'échange d'information entre les procédures séquentielles est égal à 1, c'est-à-dire qu'il y a un point d'interaction et échange d'information à chaque itération des procédures séquentielles. Si une procédure séquentielle trouve une solution améliorante (voir la définition 4.5), le contexte de cette solution est envoyé comme message, dans le cas contraire le message m_0 est envoyé pour signifier "pas de nouvelle information à échanger". À chaque itération k, la stratégie d'exploration s_i de la procédure séquentielle p_i tente d'abord de s'actualiser à partir de $\bar{h}_{ik}$ ($\bar{h}_{ik} = h_{ik} \bigcup m$, le message reçu à l'itération $k-1$). Si le message m_0 est reçu à l'itération $k-1$, l'actualisation de la stratégie d'exploration s_i se fait à partir de l'historique de recherche h_{ik} et la solution x' dans le voisinage de la solution courante x est choisie. La structure de voisinage entre les procédures séquentielles du traitement par recherches multiples coopérantes est définie de la manière suivante: à chaque itération k la procédure séquentielle p_i envoie un message à la procédure séquentielle $p_{(k+i \bmod p)}$. Cette procédure parallèle $T3$ ne fait pas appel à une mémoire centrale comme les procédures parallèles basées sur une décomposition fonctionnelle du traitement spéculatif au chapitre 4.

Définition 6.5 *L'état d'une procédure séquentielle p_i à l'itération k est donné par l'ensemble de variables $\{x, x', m_s, m_r\}$ où x est la solution courante, x' est la solution choisie dans le voisinage de la solution courante x, m_r est le message reçu par p_i à l'itération k et m_s est le message envoyé par p_i à l'itération k.*

Définition 6.6 *La* **trajectoire** *d'une procédure séquentielle p_i est donnée par le vecteur d'états EPS_i de taille t où chaque entrée contient l'état de l'itération correspondante de p_i.*

Le parcours dans le graphe de transition par une procédure séquentielle p_i consiste à prendre la valeur du champs x des t entrées du vecteur EPS_i.

Définition 6.7 *L'état de la procédure parallèle $T3$ à l'itération k est donné par un vecteur E de p entrées où chaque entrée $E(i)$ de ce vecteur contient la valeur de x_{ik}, c'est-à-dire la valeur de la variable x de la procédure séquentielle p_i à l'itération k.*

Définition 6.8 *Le parcours du graphe de transitions accomplit par la procédure parallèle $T3$ est donnée par le tableau TPC de dimension $t \times p$, où chaque entrée $(k, i) \in TPC$ contient le champs x de l'entrée $EPS_i(k)$.*

Il ne faut pas oublier que l'objectif de la présente série de tests est d'analyser la trajectoire des procédures séquentielles (particulièrement le parcours du graphe de transitions) et le parcours de la procédure parallèle $T3$. Par conséquent, ce que nous voulons mesurer ce sont les différences au niveau de l'exploration de l'espace de solutions et non pas la valeur de la fonction objectif. Nous allons donc identifier de manière unique le contexte de chaque solution et de chaque message échangé entre les procédures séquentielles. Pour ce faire, nous utiliserons la représentation numérique de chaque contexte. Le nombre de variables de décisions des exemplaires du problème que nous allons tester est 45, conséquemment les contextes de configuration sont des vecteurs binaires de taille 45. Puisque 2^{45} n'est pas une valeur numérique pouvant être traitée par nos stations SunSparc, chaque

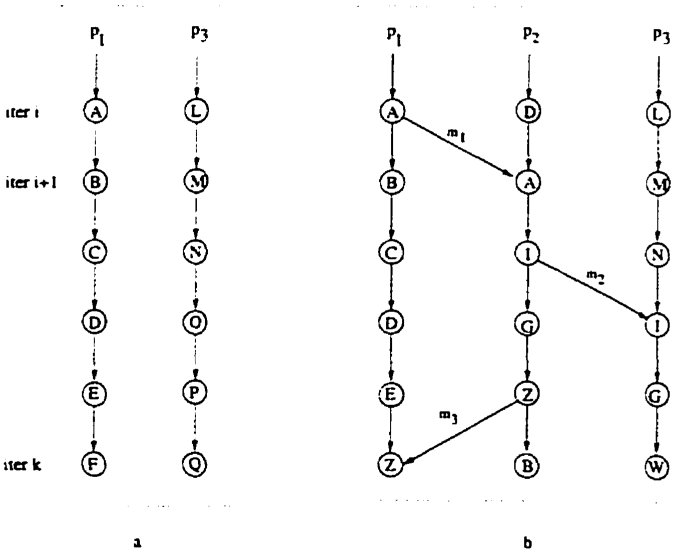


Figure 6.1 Comportement de réactions en chaîne entre procédures séquentielles

contexte faisant partie d'un chemin d'exploration ou d'un message échangé entre deux procédures séquentielles sera converti dans sa représentation hexadécimale. Pour chaque valeur hexadécimale différente nous associerons un élément différent de l'ensemble $\{0,\ldots,1999\}$ (c'est-à-dire le nombre approximatif de toutes les solutions et messages différents échangés lors de tous les tests effectués avec $T3$ dans la présente section). C'est ainsi que nous pourrons identifier toutes les solutions et messages de la trajectoire d'une procédure séquentielle.

6.3.2 Comportements dynamiques entre les stratégies d'exploration

Nous avons identifié deux patrons du comportement dynamique de réactions en chaîne entre les stratégies d'exploration: l'interaction en chaîne et l'interaction récursive. La figure 6.1a illustre schématiquement une parallélisation par recherche multiples où le parcours entre les itérations $i, i+1, \ldots, k$ de deux procédures séquentielles p_1 et p_3 passe respectivement par les solutions (A,B,C,D,E,F) et (L,M,N,O,P,Q). La figure 6.1b illustre schématiquement une parallélisation par

recherches multiples coopérantes et montre les deux patrons de comportement dynamique par réaction en chaîne entre les procédures séquentielles p_1, p_2 et p_3. La procédure séquentielle p_1 envoie un message m_1 qui modifie le parcours de la procédure p_2, ce changement de parcours de p_2 provoque l'envoi d'un message:

1. m_2 à la procédure séquentielle p_3 modifiant le parcours de cette procédure;
2. m_3 à la procédure séquentielle p_1 modifiant le parcours de p_1 à l'itération k.

Le comportement (1), mettant en cause les procédures séquentielles p_1, p_2 et p_3, illustre une dynamique provoquée par l'envoi d'un message de p_1 vers p_2 à partir duquel s'actualise la stratégie d'exploration s_2 ce qui provoque un changement de parcours chez p_2, lequel changement provoque l'envoi d'un nouveau message de p_2 vers p_3, lequel provoque un changement de parcours chez p_3, etc. La réception par p_2 du message en provenance de p_1 a eu pour effet de changer le parcours de p_2. Si l'on reprend la terminologie utilisée dans l'introduction de ce chapitre, la réception du message par p_2 change les connaissances acquises par p_2, lesquelles provoquent un changement dans le parcours de p_2, lequel changement se traduit par une nouvelle modification des connaissances acquises par p_2, ce qui veut dire une modification de l'ensemble global des connaissances acquises. Cette modification de l'ensemble global des connaissances acquises se répercute par réactions sur la procédure p_3 dont le parcours est modifié. En réalité, le changement de parcours de la procédure p_2 provoque l'envoi de nouveaux messages, dont l'un pour p_3, message qui modifie le comportement de p_3. On voit à travers ce schéma comment le comportement d'une procédure séquentielle peut affecter le comportement de plusieurs autres procédures. Il s'agit donc d'une dynamique d'interaction en chaîne entre les stratégies d'exploration qui se propage de manière séquentielle à travers les procédures séquentielles des recherches multiples coopérantes.

Le comportement (2) met en cause les procédures séquentielles p_1 et p_2, et illustre une dynamique où la procédure p_1 ayant déclenché une interaction en chaîne,

se retrouve sur le chemin de cette dynamique d'interaction en chaîne, ce qui entraîne p_1 dans un cycle de modifications récursif de son propre parcours du graphe de transitions. Encore une fois, dans les termes de l'introduction, une modification des connaissances acquises globalement engendrée par la procédure p_1 provoque une réaction en chaîne dans l'espace qui affecte le comportement de plusieurs procédures séquentielles dont la procédure p_1. Nous montrerons maintenant de manière expérimentale que ces descriptions schématiques correspondent à un phénomène global observable au niveau des recherches multiples coopérantes.

6.3.3 Dynamique d'interaction en chaîne

Hypothèse 6.1 *Il existe un lien d'interaction entre les stratégies d'exploration des procédures par recherches multiples coopérantes basé sur une dynamique d'interaction en chaîne.*

Soit une procédure parallèle $T3$ où $p = 4$ et $t = 100$. Le tableau 6.1 représente le parcours de la procédure séquentielle p_0 lors de l'exécution de $T3$. Nous allons maintenant effectuer un premier test visant à confirmer l'hypothèse 6.1. Dans la prochaine expérimentation nous isolerons la procédure p_0 des signaux en provenance des autres procédures séquentielles. Nous empêcherons les procédures séquentielles de $T3$ d'interagir avec la stratégie d'exploration de p_0 en ne permettant pas à la procédure séquentielle p_0 de lire ses messages. Lorsque dans une procédure par recherches multiples coopérantes, une procédure séquentielle p_i ne peut pas lire ses messages, la stratégie d'exploration s_i de la procédure p_i s'actualise uniquement à partir de son propre historique de recherche. Conséquemment l'exploration de l'espace de solutions par p_i est identique à celui d'une procédure tabou séquentielle avec la stratégie d'exploration s_i. Bien que p_0 ne lise pas ses messages, cette procédure continue d'envoyer des messages aux autres procédures séquentielles de $T3$.

Le tableau 6.2 montre la trajectoire de p_0 suite à cette modification. On y observe des changements importants des variables x, x', m_s, m_r par rapport au tableau 6.1. La colonne x, correspondant au parcours du graphe de transitions effectué par p_0, est très différente entre ces deux tableaux, ce qui témoigne de l'impact pour p_0 de l'interférence des recherches multiples coopérantes sur la recherche tabou. Les changements de la colonne m_s des messages envoyés aux autres procédures séquentielles par p_0 découlent naturellement du fait que les messages générés par une procédure séquentielle ne sont pas les mêmes lorsque le parcours du graphe de transitions n'est pas le même (parce que l'ensemble des solutions améliorantes n'est plus le même).

La différence au niveau des colonnes de messages reçus m_r est très révélatrice. En effet, à l'exception de l'absence de la lecture des messages par p_0, tous les paramètres de conception de la procédure parallèle $T3$ sont identiques pour les deux tableaux, y compris la structure de voisinage (toutes les procédures séquentielles continuent d'envoyer et de recevoir leurs messages des mêmes procédures voisines). Le seul changement dans ce test pouvant se répercuter sur les autres procédures séquentielles, provient de la colonne m_s des messages envoyés par p_0, puisque la séquence d'envoi des messages n'est plus la même. C'est dire que la seule explication possible au fait que la colonne m_r de p_0 puisse différer, provient des modifications dans le parcours de l'espace de solutions des autres procédures séquentielles suite au changement dans la séquence d'envoi de messages par p_0.

Cette différence de la colonne m_r de p_0 constitue un premier indice de l'existence d'une dynamique d'interaction en chaîne provoquée par l'envoi de messages par p_0 et de la dépendance du comportement d'exploration des procédures séquentielles sur cette dynamique. Le fait que p_0 n'envoie pas la même séquence de messages aux autres procédures séquentielles, se traduit de manière ultime par un changement dans les séquences de messages reçues par p_0 en provenance des autres procédures séquentielles. C'est donc dire qu'au minimum, il existe une chaîne d'interaction

$p_0, p_i, i \neq 0, p_0$ dont la dynamique fut modifiée par les changements dans p_0. Nous allons maintenant montrer que la dynamique d'interaction en chaîne s'étend à plus d'une procédure séquentielle.

Les tableaux 6.3 et 6.4 montrent de manière plus directe l'impact sur la procédure parallèle $T3$ de la modification effectuée à p_0. Les colonnes du tableau 6.3 montrent la trajectoire effectuée par $T3$ sur 100 itérations avant la modification de la procédure p_0, tandis que le tableau 6.4 donne la même information après le changement à p_0. On constate que cette simple modification de p_0 a eu un impact considérable sur le parcours de $T3$, puisque $T3$ n'explore pratiquement pas les mêmes solutions avant et après les modifications à p_0. Dans les deux cas la procédure $T3$ commence avec les mêmes solutions initiales pour p_0, p_1, p_2 et p_3 (itération 1), et visite les mêmes solutions à l'itération 2. La procédure p_0 est la première à modifier son parcours (à l'itération 3) ce qui est normal puisqu'elle ne lit pas ses messages en provenance des autres procédures. Les changements du parcours de p_0 finissent par affecter les autres procédures par le biais de la nouvelle séquence de messages envoyée par p_0. La procédure p_1 est la première touchée, à l'itération 4 elle reçoit le message "9" de p_0 généré à l'itération 3. Ensuite, les procédures p_2 et p_3 sont affectées à l'itération 5 suite à la réception par p_2 du message "202" en provenance de p_1 et à la réception par p_3 du message "9" en provenance de p_1, etc. On constate qu'après l'itération 17, la procédure parallèle de $T3$ modifiée ne visite aucune des solutions ayant été visitées par la procédure $T3$ originale. Ces séquences de modifications et le fait que les deux ensembles de solutions visitées par les deux procédures $T3$ soient totalement différents, montrent bien que la dynamique d'interaction en chaîne peut s'étendre sur une longue séquence de procédures séquentielles.

Mentionnons que les nombres qui n'apparaissent pas dans les tableaux 6.3 et 6.4 (par exemple les solutions 10, 11, 16, 17, 18, etc.) correspondent à des solutions x' générées dans le voisinage de la solution courante x d'une procédure p_i qui (1) ne sont pas devenues la solution courante x d'une procédure p_i à l'itération suivante

parce qu'elles ont été éclipsées par l'arrivée d'un message en provenance d'une autre procédure et (2) elles ne se sont pas qualifiées pour l'envoi d'un message m_s par p_i parce qu'elles n'étaient pas des solutions améliorantes.

On peut conclure de ce premier test qu'il y a bien un effet de propagation du comportement d'une procédure séquentielle sur celui des autres procédures séquentielles des recherches multiples coopérantes. Cette dynamique détermine de manière importante le parcours du graphe de transitions effectué par les procédures séquentielles. Nous n'avons qu'à modifier légèrement la dynamique de cette interaction (en empêchant p_0 de lire ses messages ce qui a provoqué une nouvelle séquence d'envoi de messages en provenance de p_0) pour induire un bouleversement presque complet de l'exploration de l'espace de solutions.

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	98	67	∅	116
2	5	9	∅	6	52	116	101	∅	∅
3	6	13	6	∅	53	101	99	∅	∅
4	13	17	13	7	54	99	98	∅	∅
5	7	14	7	∅	55	98	67	∅	∅
6	14	24	14	∅	56	67	101	∅	∅
7	24	28	24	∅	57	97	125	∅	∅
8	28	32	∅	∅	58	125	128	125	∅
9	32	36	32	∅	59	128	130	∅	∅
10	36	40	36	∅	60	125	132	∅	∅
11	40	42	∅	∅	61	132	134	132	133
12	36	46	∅	∅	62	133	136	∅	∅
13	46	49	∅	∅	63	136	140	∅	∅
14	49	52	49	∅	64	132	143	∅	∅
15	52	54	52	∅	65	143	134	∅	∅
16	54	56	∅	∅	66	134	125	∅	∅
17	56	59	∅	57	67	125	151	∅	∅
18	57	53	57	12	68	151	153	∅	∅
19	12	65	∅	63	69	153	134	∅	155
20	63	69	∅	∅	70	155	138	∅	∅
21	57	53	∅	∅	71	138	151	∅	∅
22	53	60	∅	∅	72	132	161	∅	∅
23	60	66	∅	∅	73	161	163	161	∅
24	66	53	∅	76	74	163	165	∅	∅
25	76	60	∅	74	75	161	167	∅	∅
26	74	62	∅	∅	76	167	169	∅	∅
27	62	82	∅	∅	77	169	167	169	∅
28	82	66	∅	∅	78	167	161	∅	∅
29	57	67	∅	∅	79	161	172	∅	∅
30	67	87	67	∅	80	169	173	∅	∅
31	87	90	∅	∅	81	173	167	∅	175
32	67	92	∅	∅	82	175	178	∅	∅
33	92	94	∅	∅	83	178	161	∅	∅
34	94	96	∅	71	84	161	173	∅	∅
35	67	97	∅	∅	85	173	167	∅	∅
36	97	98	97	∅	86	167	178	∅	∅
37	98	99	∅	∅	87	178	161	∅	∅
38	99	101	∅	∅	88	169	185	∅	97
39	101	67	∅	∅	89	97	186	97	∅
40	67	98	∅	∅	90	97	188	∅	∅
41	98	99	∅	∅	91	97	99	∅	∅
42	99	103	∅	∅	92	99	190	∅	∅
43	103	98	∅	∅	93	190	191	190	∅
44	98	105	∅	97	94	191	194	∅	∅
45	97	106	∅	∅	95	194	195	∅	∅
46	97	98	∅	∅	96	190	99	∅	∅
47	98	110	∅	∅	97	99	191	∅	∅
48	110	112	∅	∅	98	191	97	∅	∅
49	97	99	∅	∅	99	97	199	∅	∅
50	99	98	∅	∅	100	199	99	∅	∅

Tableau 6.1 EPS_0 pour $T3$

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	0	0	51	306	316	∅	∅
2	5	9	∅	6	52	316	312	∅	∅
3	9	202	9	∅	53	312	318	∅	∅
4	202	203	202	7	54	318	314	∅	320
5	203	205	203	∅	55	314	316	∅	∅
6	205	209	205	∅	56	316	312	∅	∅
7	209	210	209	∅	57	312	327	∅	∅
8	210	214	210	∅	58	327	314	∅	∅
v 9	214	218	214	∅	59	306	289	∅	330
10	218	221	∅	∅	60	289	335	289	∅
11	221	224	221	∅	61	335	339	∅	∅
12	224	226	224	∅	62	289	342	∅	∅
13	226	230	226	∅	63	342	293	∅	∅
14	230	231	∅	∅	64	293	292	293	∅
15	226	233	∅	∅	65	292	293	292	∅
16	233	235	∅	∅	66	293	295	∅	∅
17	235	236	∅	∅	67	295	351	∅	∅
18	226	239	∅	238	68	292	294	∅	∅
19	239	242	∅	240	69	294	293	∅	355
20	242	233	∅	∅	70	293	359	∅	∅
21	233	247	∅	∅	71	359	362	359	∅
22	247	239	∅	∅	72	362	366	∅	359
23	239	242	∅	251	73	366	367	∅	∅
24	242	233	∅	256	74	367	370	367	∅
25	233	247	∅	∅	75	370	373	∅	∅
26	226	230	∅	∅	76	373	359	∅	∅
27	230	266	230	∅	77	359	380	∅	∅
28	266	270	266	263	78	380	370	∅	∅
29	270	273	∅	∅	79	370	373	∅	∅
30	266	276	∅	∅	80	373	359	∅	∅
31	276	280	∅	278	81	359	380	∅	∅
32	280	283	∅	∅	82	380	390	∅	∅
33	266	286	∅	∅	83	390	392	∅	∅
34	286	276	∅	∅	84	367	376	∅	∅
35	276	289	∅	278	85	376	362	∅	∅
36	289	292	289	∅	86	362	395	∅	∅
37	292	293	292	∅	87	367	380	∅	∅
38	293	294	∅	∅	88	380	359	∅	∅
39	294	295	∅	∅	89	359	293	∅	∅
40	295	289	∅	∅	90	293	370	∅	∅
41	289	293	∅	∅	91	370	373	∅	∅
42	293	294	∅	∅	92	373	359	∅	∅
43	294	295	∅	∅	93	359	380	∅	∅
44	295	289	∅	∅	94	380	370	∅	278
45	289	306	∅	∅	95	367	409	∅	∅
46	306	307	306	∅	96	409	410	409	∅
47	307	309	∅	308	97	410	411	∅	∅
48	306	312	∅	∅	98	409	412	∅	∅
49	312	314	∅	∅	99	412	413	∅	∅
50	314	306	∅	∅	100	413	414	∅	∅

Tableau 6.2 EPS_0 sans lecture des messages de m_r

it	p_0	p_1	p_2	p_3	it	p_0	p_1	p_2	p_3
1	1	2	3	4	51	98	115	89	116
2	5	6	7	8	52	116	117	74	118
3	6	7	8	12	53	101	111	76	116
4	13	6	7	8	54	99	121	77	88
5	7	8	13	6	55	98	122	71	112
6	14	21	7	8	56	67	123	97	116
7	24	14	21	27	57	97	97	87	124
8	28	21	24	31	58	125	126	97	88
9	32	33	7	8	59	128	122	125	129
10	36	32	38	39	60	125	123	131	112
11	40	21	36	27	61	132	126	133	124
12	36	43	44	45	62	133	111	132	88
13	46	15	47	48	63	136	137	138	139
14	49	50	38	15	64	132	121	133	116
15	52	53	26	49	65	143	126	144	104
16	54	52	53	27	66	134	145	104	147
17	56	57	7	8	67	125	148	149	104
18	57	60	61	12	68	151	145	152	148
19	12	57	63	61	69	153	126	154	155
20	63	61	67	68	70	155	148	149	148
21	57	67	57	61	71	138	157	158	159
22	53	71	72	57	72	132	155	152	155
23	60	74	57	71	73	161	162	155	126
24	66	76	71	76	74	163	164	161	148
25	76	77	74	78	75	161	148	166	121
26	74	79	80	74	76	167	126	104	162
27	62	76	81	76	77	169	162	149	164
28	82	83	74	85	78	167	164	170	169
29	57	74	86	71	79	161	148	104	126
30	67	76	71	74	80	169	126	152	155
31	87	67	78	89	81	173	174	149	175
32	67	71	77	77	82	175	155	154	177
33	92	77	93	76	83	178	148	152	179
34	94	95	71	74	84	161	145	179	180
35	67	71	76	89	85	173	155	154	179
36	97	79	74	79	86	167	182	152	174
37	98	97	89	76	87	178	126	104	183
38	99	57	79	100	88	169	148	97	184
39	101	74	76	71	89	97	184	87	183
40	67	76	74	74	90	97	162	97	97
41	98	79	89	80	91	97	164	98	184
42	99	74	77	71	92	99	148	110	189
43	103	71	101	79	93	190	155	97	174
44	98	97	71	74	94	191	190	99	193
45	97	104	79	57	95	194	87	98	190
46	97	107	85	76	96	190	190	67	184
47	98	108	71	107	97	99	196	101	193
48	110	111	74	89	98	191	197	99	174
49	97	113	111	77	99	97	190	98	184
50	99	111	77	71	100	199	99	67	179

Tableau 6.3 Évolution de $T3$ avec lecture des messages par p_0

it	p_0	p_1	p_2	p_3	it	p_0	p_1	p_2	p_3
1	1	2	3	4	51	306	315	287	311
2	5	6	7	8	52	316	281	278	285
3	9	7	8	12	53	312	302	317	282
4	202	9	7	8	54	318	317	319	320
5	203	8	202	9	55	314	315	317	322
6	205	206	7	8	56	316	281	323	320
7	209	205	206	51	57	312	302	325	326
8	210	211	209	205	58	327	308	317	329
9	214	205	7	8	59	306	330	331	320
10	218	214	220	58	60	289	333	323	334
11	221	222	47	223	61	335	330	337	338
12	224	205	38	221	62	289	340	341	326
13	226	224	228	229	63	342	341	323	344
14	230	219	226	27	64	293	345	346	334
15	226	232	220	51	65	292	347	293	345
16	233	234	38	45	66	293	348	349	292
17	235	217	7	8	67	295	345	323	320
18	226	227	237	238	68	292	352	353	354
19	239	219	240	237	69	294	347	354	355
20	242	205	243	244	70	293	356	357	358
21	233	243	245	237	71	359	360	263	355
22	247	248	243	250	72	362	363	364	359
23	239	251	252	248	73	366	347	308	364
24	242	252	255	256	74	367	364	364	369
25	233	251	258	255	75	370	345	369	367
26	226	260	261	262	76	373	374	375	376
27	230	263	255	256	77	359	377	364	369
28	266	230	263	269	78	380	374	371	382
29	270	269	266	272	79	370	383	369	372
30	266	263	274	275	80	373	384	385	364
31	276	277	278	279	81	359	374	364	385
32	280	267	281	282	82	380	388	371	389
33	266	260	282	285	83	390	391	369	387
34	286	277	263	279	84	367	383	385	393
35	276	278	287	288	85	376	388	364	369
36	289	281	290	291	86	362	394	263	389
37	292	289	281	285	87	367	263	396	397
38	293	263	292	279	88	380	383	263	371
39	294	287	263	288	89	359	374	267	399
40	295	281	296	291	90	293	345	400	364
41	289	284	278	298	91	370	402	263	387
42	293	263	284	282	92	373	345	277	405
43	294	290	300	285	93	359	347	267	364
44	295	301	278	303	94	380	407	278	369
45	289	278	287	282	95	367	345	281	385
46	306	281	281	279	96	409	363	287	387
47	307	308	306	285	97	410	409	263	371
48	306	310	263	311	98	409	352	284	369
49	312	313	290	288	99	412	360	281	385
50	314	308	281	279	100	413	356	290	389

Tableau 6.4 Évolution de $T3$ sans lecture des messages par p_0

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	88	124	∅	504
2	5	9	∅	6	52	504	112	∅	∅
3	6	13	∅	∅	53	112	139	∅	∅
4	13	17	∅	7	54	139	88	∅	∅
5	7	14	∅	8	55	88	129	∅	∅
6	8	53	∅	∅	56	129	112	∅	∅
7	53	421	∅	∅	57	112	124	∅	∅
8	421	424	∅	∅	58	124	515	∅	∅
9	424	428	∅	∅	59	515	517	∅	∅
10	428	430	∅	∅	60	116	88	∅	∅
11	424	432	∅	∅	61	88	112	∅	∅
12	432	434	∅	∅	62	112	110	∅	∅
13	434	424	∅	∅	63	116	124	∅	∅
14	424	436	∅	∅	64	124	88	∅	∅
15	436	432	∅	∅	65	88	139	∅	∅
16	432	439	∅	∅	66	139	112	∅	∅
17	439	434	∅	∅	67	112	124	∅	∅
18	434	436	∅	12	68	124	88	∅	∅
19	12	432	∅	63	69	88	139	∅	∅
20	63	439	∅	∅	70	139	112	∅	∅
21	439	434	∅	∅	71	116	118	∅	∅
22	424	421	∅	∅	72	118	120	∅	∅
23	421	451	∅	∅	73	120	532	∅	∅
24	451	455	∅	∅	74	118	535	∅	161
25	421	458	∅	69	75	161	535	∅	∅
26	69	460	∅	53	76	535	161	∅	∅
27	53	57	∅	∅	77	161	167	∅	∅
28	57	66	∅	464	78	167	169	∅	∅
29	464	468	∅	∅	79	169	167	∅	∅
30	468	471	∅	∅	80	167	161	∅	∅
31	57	53	∅	∅	81	161	172	∅	543
32	53	60	∅	∅	82	169	173	∅	544
33	60	66	∅	475	83	544	167	∅	∅
34	475	82	∅	∅	84	167	178	∅	∅
35	82	478	∅	∅	85	178	173	∅	∅
36	478	66	∅	479	86	173	553	∅	∅
37	479	60	∅	∅	87	553	167	∅	∅
38	60	478	∅	∅	88	167	173	∅	∅
39	57	139	∅	∅	89	173	161	∅	∅
40	139	118	∅	∅	90	169	185	∅	∅
41	118	120	∅	∅	91	185	560	∅	∅
42	139	490	∅	∅	92	560	564	∅	∅
43	490	492	∅	∅	93	185	566	∅	∅
44	492	494	∅	∅	94	566	567	∅	∅
45	139	421	∅	∅	95	567	569	∅	∅
46	421	490	∅	∅	96	185	572	∅	571
47	490	499	∅	∅	97	571	574	∅	∅
48	499	492	∅	∅	98	574	578	∅	∅
49	492	116	∅	∅	99	578	580	∅	∅
50	116	88	∅	∅	100	580	582	∅	∅

Tableau 6.5 EPS_0 sans envoi de messages aux autres procédures séquentielles

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	2	3	4	51	98	115	89	116
2	5	6	7	8	52	116	117	74	118
3	6	7	8	12	53	101	111	76	116
4	13	6	7	8	54	99	121	77	88
5	7	8	13	6	55	98	122	71	112
6	14	21	7	8	56	67	123	97	116
7	24	14	21	27	57	97	97	87	124
8	28	21	24	31	58	125	126	97	88
9	32	33	7	8	59	128	122	125	129
10	36	32	38	39	60	125	123	131	112
11	40	21	36	27	61	132	126	133	124
12	36	43	44	45	62	133	111	132	88
13	46	15	47	48	63	136	137	138	139
14	49	50	38	15	64	132	121	133	116
15	52	53	26	49	65	143	126	144	104
16	54	52	53	27	66	134	145	104	147
17	56	57	7	8	67	125	148	149	104
18	57	60	61	12	68	151	145	152	148
19	12	57	63	61	69	153	126	154	155
20	63	61	67	68	70	155	148	149	148
21	57	67	57	61	71	138	157	158	159
22	53	71	72	57	72	132	155	152	155
23	60	74	57	71	73	161	162	155	126
24	66	76	71	76	74	163	164	161	148
25	76	77	74	78	75	161	148	166	121
26	74	79	80	74	76	167	126	104	162
27	62	76	81	76	77	169	162	149	164
28	82	83	74	85	78	167	164	170	169
29	57	74	86	71	79	161	148	104	126
30	67	76	71	74	80	169	126	152	155
31	87	67	78	89	81	173	174	149	175
32	67	71	77	77	82	175	155	154	177
33	92	77	93	76	83	178	148	152	179
34	94	95	71	74	84	161	145	179	180
35	67	71	76	89	85	173	155	154	179
36	97	79	74	79	86	167	182	152	174
37	98	97	89	76	87	178	126	104	183
38	99	57	79	100	88	169	148	97	184
39	101	74	76	71	89	97	184	87	183
40	67	76	74	74	90	97	162	97	97
41	98	79	89	80	91	97	164	98	184
42	99	74	77	71	92	99	148	110	189
43	103	71	101	79	93	190	155	97	174
44	98	97	71	74	94	191	190	99	193
45	97	104	79	57	95	194	87	98	190
46	97	107	85	76	96	190	190	67	184
47	98	108	71	107	97	99	196	101	193
48	110	111	74	89	98	191	197	99	174
49	97	113	111	77	99	97	190	98	184
50	99	111	77	71	100	199	99	67	179

Tableau 6.6 Évolution de $T3$ avec envoi de messages par p_0

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	2	3	4	51	88	496	464	504
2	5	6	7	8	52	504	505	151	118
3	6	7	8	12	53	112	506	507	504
4	13	14	7	8	54	139	508	132	510
5	7	8	19	14	55	88	511	153	195
6	8	21	7	8	56	129	505	146	504
7	53	25	21	27	57	112	146	512	513
8	421	21	423	31	58	124	508	514	510
9	424	425	7	8	59	515	496	146	124
10	428	429	26	45	60	116	132	153	195
11	424	21	38	27	61	88	161	132	513
12	432	15	433	58	62	112	132	132	510
13	434	50	15	48	63	116	153	519	124
14	424	435	26	45	64	124	520	153	504
15	436	437	38	27	65	88	132	512	522
16	432	438	433	58	66	139	125	522	523
17	439	50	7	8	67	112	134	146	522
18	434	435	440	12	68	124	153	525	154
19	12	437	63	440	69	88	525	526	104
20	63	440	441	442	70	139	464	525	522
21	439	443	63	440	71	116	134	528	171
22	424	15	447	448	72	118	153	529	154
23	421	435	449	450	73	120	132	525	531
24	451	452	63	440	74	118	161	533	104
25	421	15	69	457	75	161	163	528	171
26	69	53	459	448	76	535	161	537	154
27	53	459	462	463	77	161	167	538	531
28	57	55	464	465	78	167	169	533	522
29	464	466	467	457	79	169	167	169	540
30	468	469	470	448	80	167	540	528	541
31	57	461	472	473	81	161	169	525	543
32	53	474	468	440	82	169	167	544	545
33	60	466	470	475	83	544	545	169	546
34	475	57	472	476	84	167	169	547	545
35	82	476	468	57	85	178	549	495	547
36	478	60	57	479	86	173	178	547	552
37	479	480	481	482	87	553	167	552	545
38	60	482	464	190	88	167	173	554	555
39	57	60	472	482	89	173	169	547	550
40	139	66	485	486	90	169	556	557	558
41	118	82	464	488	91	185	540	556	559
42	139	480	470	482	92	560	556	562	563
43	490	491	472	79	93	185	565	554	550
44	492	493	132	491	94	566	163	557	558
45	139	132	153	102	95	567	556	552	545
46	421	496	132	497	96	185	570	562	571
47	490	498	464	496	97	571	565	547	573
48	499	496	151	102	98	574	575	576	571
49	492	501	153	486	99	578	576	579	574
50	116	502	134	482	100	580	570	576	107

Tableau 6.7 Évolution de $T3$ sans envoi de messages par p_0

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	894	899	∅	∅
2	5	9	∅	6	52	899	892	∅	∅
3	6	13	6	∅	53	892	902	∅	∅
4	13	17	13	6	54	885	911	∅	∅
5	6	808	6	∅	55	911	914	911	∅
6	808	810	∅	∅	56	914	918	914	∅
7	810	811	810	∅	57	918	920	∅	∅
8	811	814	811	∅	58	914	921	∅	∅
9	814	817	814	∅	59	921	922	∅	∅
10	817	818	∅	∅	60	922	923	∅	168
11	818	819	818	∅	61	914	926	∅	∅
12	819	821	∅	∅	62	926	929	∅	∅
13	818	822	∅	∅	63	929	921	∅	53
14	822	823	822	∅	64	53	932	∅	∅
15	823	824	∅	∅	65	932	934	∅	∅
16	824	825	∅	∅	66	934	929	∅	∅
17	825	826	825	∅	67	929	921	∅	∅
18	826	827	∅	825	68	921	932	∅	∅
19	825	830	∅	829	69	914	938	∅	∅
20	825	826	∅	442	70	938	940	938	∅
21	442	833	∅	∅	71	940	941	∅	∅
22	833	837	∅	∅	72	941	943	941	∅
23	837	841	837	∅	73	943	945	943	∅
24	841	845	841	∅	74	945	946	∅	∅
25	845	837	845	841	75	943	948	∅	∅
26	841	851	∅	∅	76	948	950	∅	∅
27	851	853	851	∅	77	950	951	950	481
28	853	856	∅	∅	78	481	953	∅	∅
29	856	859	∅	∅	79	953	955	∅	107
30	859	861	∅	∅	80	950	958	∅	∅
31	861	862	861	∅	81	958	959	∅	∅
32	862	865	∅	∅	82	959	961	∅	166
33	865	867	∅	∅	83	166	951	∅	∅
34	867	851	∅	∅	84	951	959	∅	518
35	851	862	∅	873	85	518	958	∅	∅
36	873	865	∅	∅	86	958	951	∅	∅
37	865	867	∅	∅	87	951	959	∅	∅
38	867	851	∅	∅	88	950	971	∅	544
39	851	882	∅	∅	89	544	972	544	∅
40	882	885	∅	∅	90	544	973	∅	∅
41	885	888	885	∅	91	544	968	∅	∅
42	888	891	∅	∅	92	968	518	∅	∅
43	885	892	∅	∅	93	518	978	∅	563
44	892	894	∅	∅	94	544	979	∅	∅
45	894	896	∅	∅	95	979	968	∅	∅
46	885	898	∅	∅	96	968	974	∅	∅
47	898	899	∅	∅	97	974	518	∅	∅
48	899	892	∅	∅	98	518	976	∅	∅
49	892	902	∅	901	99	976	968	∅	∅
50	901	894	∅	∅	100	968	974	∅	523

Tableau 6.8 EPS_0 avec la structure de voisinage $p_{(j+i+1 \bmod p)}$

iter.	p_0	p_1	p_2	p_3	iter.	p_0	p_1	p_2	p_3
1	1	2	3	4	51	894	546	886	901
2	5	6	7	8	52	899	548	546	904
3	6	7	8	12	53	892	546	906	907
4	13	6	7	8	54	885	908	909	901
5	6	7	8	13	55	911	912	767	913
6	808	6	7	8	56	914	546	916	917
7	810	611	26	27	57	918	909	908	914
8	811	810	813	31	58	914	908	909	904
9	814	6	7	8	59	921	916	906	917
10	817	814	47	39	60	922	168	916	913
11	818	619	44	58	61	914	924	925	904
12	819	818	820	27	62	926	927	546	901
13	818	614	38	45	63	929	930	906	53
14	822	611	26	48	64	53	128	931	421
15	823	619	44	822	65	932	924	546	53
16	824	617	47	27	66	934	927	908	466
17	825	6	7	8	67	929	930	909	461
18	826	825	440	591	68	921	935	906	53
19	825	440	591	829	69	914	937	916	55
20	825	442	440	591	70	938	168	908	466
21	442	445	591	829	71	940	128	909	938
22	833	834	445	836	72	941	942	168	461
23	837	445	839	840	73	943	168	546	941
24	841	837	591	829	74	945	943	558	466
25	845	846	847	841	75	943	558	947	469
26	841	445	849	850	76	948	128	558	53
27	851	842	852	845	77	950	930	563	481
28	853	851	855	853	78	481	950	900	563
29	856	857	858	837	79	953	924	563	107
30	859	839	849	860	80	950	128	563	957
31	861	854	852	845	81	958	168	900	107
32	862	861	591	864	82	959	166	558	960
33	865	842	829	866	83	166	962	963	964
34	867	445	869	829	84	951	518	956	107
35	851	871	872	873	85	518	966	558	584
36	873	543	871	872	86	958	518	900	574
37	865	545	872	873	87	951	968	956	969
38	867	571	545	879	88	950	544	563	970
39	851	545	880	881	89	544	968	546	960
40	882	558	883	873	90	544	544	87	546
41	885	883	880	887	91	544	544	168	574
42	888	545	889	890	92	968	968	975	107
43	885	559	550	879	93	518	976	168	563
44	892	558	893	550	94	544	518	924	97
45	894	550	895	890	95	979	974	942	563
46	885	484	550	879	96	968	968	168	900
47	898	559	886	887	97	974	976	927	954
48	899	558	776	873	98	518	518	924	563
49	892	484	900	901	99	976	544	935	558
50	901	545	903	696	100	968	523	930	900

Tableau 6.9 Évolution de $T3$ avec la structure de voisinage $p_{(j+i+1 \bmod p)}$

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	697	701	∅	∅
2	5	9	∅	7	52	678	681	∅	∅
3	7	590	7	8	53	681	685	∅	∅
4	8	12	8	∅	54	685	686	∅	∅
5	12	455	12	∅	55	686	689	∅	∅
6	455	599	∅	12	56	25	714	∅	713
7	12	604	12	∅	57	713	685	∅	∅
8	604	608	∅	∅	58	685	686	∅	∅
9	12	610	∅	∅	59	686	681	∅	436
10	610	612	∅	∅	60	678	721	∅	720
11	612	615	∅	∅	61	720	724	720	∅
12	12	65	∅	∅	62	724	727	724	∅
13	65	610	∅	∅	63	727	523	727	∅
14	610	622	∅	∅	64	523	524	∅	∅
15	622	612	∅	∅	65	727	732	∅	730
16	612	65	∅	∅	66	730	735	∅	∅
17	65	610	∅	∅	67	735	739	735	∅
18	610	622	∅	∅	68	739	741	∅	735
19	622	612	∅	53	69	735	743	∅	696
20	12	8	∅	627	70	735	746	∅	∅
21	627	632	627	∅	71	746	730	∅	∅
22	632	635	∅	∅	72	730	751	730	∅
23	635	639	635	∅	73	751	740	∅	∅
24	639	642	∅	∅	74	740	754	∅	∅
25	642	646	642	∅	75	754	735	∅	∅
26	646	650	∅	642	76	735	751	∅	∅
27	642	651	∅	∅	77	751	740	∅	∅
28	651	655	∅	∅	78	740	754	∅	∅
29	655	658	∅	∅	79	754	735	∅	∅
30	642	659	∅	∅	80	735	760	∅	∅
31	659	651	∅	∅	81	760	762	∅	674
32	651	662	∅	∅	82	730	740	∅	∅
33	662	664	662	∅	83	740	751	∅	764
34	664	667	∅	∅	84	764	771	∅	∅
35	667	642	∅	∅	85	730	751	∅	∅
36	642	672	∅	∅	86	751	740	∅	∅
37	672	664	∅	∅	87	740	754	∅	∅
38	664	678	∅	∅	88	754	696	∅	∅
39	678	681	678	∅	89	696	712	696	∅
40	681	685	∅	∅	90	712	702	∅	∅
41	685	686	∅	∅	91	702	716	∅	∅
42	686	689	∅	∅	92	716	718	∅	781
43	689	681	∅	∅	93	781	712	∅	∅
44	681	685	∅	∅	94	712	730	∅	∅
45	685	686	∅	∅	95	730	702	∅	∅
46	686	689	∅	∅	96	702	712	∅	∅
47	689	687	∅	72	97	712	789	∅	∅
48	72	693	∅	∅	98	789	793	∅	∅
49	678	694	∅	∅	99	696	702	∅	∅
50	694	697	∅	∅	100	702	708	∅	∅

Tableau 6.10 EPS_0 pour $T3$ avec 5 procédures séquentielles

itér.	p_0	p_1	p_2	p_3	p_4	itér.	p_0	p_1	p_2	$p_3 v$	p_4
1	1	2	3	4	587	51	697	696	667	699	700
2	5	6	7	8	588	52	678	702	642	696	703
3	7	8	588	12	6	53	681	704	702	702	706
4	8	588	592	6	7	54	685	706	664	708	709
5	12	8	588	592	6	55	686	702	672	696	706
6	455	238	12	8	598	56	689	704	642	712	713
7	12	8	601	602	603	57	713	708	678	702	715
8	604	45	12	8	6	58	685	439	685	708	678
9	12	609	610	39	611	59	686	436	681	718	439
10	610	8	612	613	614	60	678	719	689	439	720
11	612	39	12	8	617	61	720	722	719	702	723
12	12	45	618	27	619	62	724	725	720	722	726
13	65	620	610	39	598	63	727	720	681	696	720
14	610	58	621	45	611	64	523	729	689	674	727
15	622	39	612	620	623	65	727	719	686	730	674
16	612	45	618	58	6	66	730	725	733	684	734
17	65	620	610	39	10	67	735	720	678	730	738
18	610	8	621	45	624	68	739	729	681	740	735
19	622	53	12	8	625	69	735	719	742	696	739
20	12	421	626	625	627	70	735	744	678	702	746
21	627	53	629	626	625	71	746	722	685	708	748
22	632	625	626	625	624	72	730	749	681	696	750
23	635	636	637	636	625	73	751	730	689	753	739
24	639	53	641	640	635	74	740	722	694	702	746
25	642	57	626	625	645	75	754	755	685	712	748
26	646	60	642	57	625	76	735	729	686	718	730
27	642	66	651	636	652	77	751	719	689	716	674
28	651	651	653	654	636	78	740	725	678	702	735
29	655	480	655	656	657	79	754	720	757	712	739
30	642	60	658	648	655	80	735	757	758	696	759
31	659	66	660	636	648	81	760	719	757	674	735
32	651	660	661	654	636	82	730	722	763	764	730
33	662	478	663	625	657	83	740	764	766	767	768
34	664	499	662	666	625	84	764	767	757	764	770
35	667	57	669	602	666	85	730	764	763	772	746
36	642	60	663	666	671	86	751	772	733	773	730
37	672	82	661	673	666	87	740	773	774	764	739
38	664	57	676	666	602	88	754	764	775	776	746
39	678	66	679	680	666	89	696	776	763	777	735
40	681	60	682	683	678	90	712	545	696	778	760
41	685	478	642	666	685	91	702	558	760	779	545
42	686	66	687	680	666	92	716	550	757	776	781
43	689	60	642	691	683	93	781	563	782	777	551
44	681	478	651	683	684	94	712	559	761	778	781
45	685	66	655	692	680	95	730	558	783	764	784
46	686	57	642	680	691	96	702	783	785	767	786
47	689	72	659	691	684	97	712	563	783	788	781
48	72	75	651	683	680	98	789	559	790	767	792
49	678	72	662	666	691	99	696	794	795	796	784
50	694	695	664	696	666	100	702	545	783	799	800

Tableau 6.11 Évolution de $T3$ avec 5 procédures séquentielles

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	1094	1097	1094	∅
2	5	9	∅	6	52	1097	1099	∅	∅
3	6	13	6	∅	53	1099	1070	∅	∅
4	13	17	13	7	54	1070	1082	∅	∅
5	7	14	7	∅	55	1082	1099	∅	∅
6	14	24	14	∅	56	1099	1070	∅	∅
7	24	28	24	∅	57	1070	1102	∅	∅
8	28	32	∅	∅	58	1102	1082	∅	∅
9	32	36	32	∅	59	1082	1108	∅	∅
10	36	40	36	∅	60	1108	1111	∅	1110
11	40	42	∅	∅	61	1110	1115	∅	∅
12	42	993	∅	∅	62	1094	1087	∅	∅
13	36	46	∅	∅	63	1087	1122	∅	∅
14	46	49	∅	∅	64	1122	1126	∅	∅
15	49	52	49	∅	65	1094	1070	∅	∅
16	52	54	52	1002	66	1070	1082	∅	∅
17	1002	1005	∅	∅	67	1082	1099	∅	1134
18	1005	1008	1005	∅	68	1134	1070	∅	∅
19	1008	1011	∅	∅	69	1070	1102	∅	∅
20	1011	1014	1011	∅	70	1102	1082	∅	∅
21	1014	1017	∅	∅	71	1082	1099	∅	∅
22	1017	1014	1017	∅	72	1099	1087	∅	∅
23	1014	1011	∅	∅	73	1094	1151	∅	1149
24	1011	1017	∅	∅	74	1149	1154	1149	∅
25	1017	1029	∅	∅	75	1149	1158	∅	∅
26	1029	1014	∅	∅	76	1158	1162	∅	∅
27	1014	1033	∅	∅	77	1149	1156	∅	∅
28	1033	1036	1033	∅	78	1156	1166	∅	∅
29	1036	1039	∅	∅	79	1166	1167	∅	∅
30	1039	1043	1039	∅	80	1149	1165	∅	∅
31	1043	1047	∅	∅	81	1165	1156	∅	∅
32	1047	1050	∅	∅	82	1156	1152	∅	∅
33	1050	1033	∅	∅	83	1152	1163	∅	1172
34	1033	1043	∅	∅	84	1172	1165	∅	∅
35	1043	1050	∅	∅	85	1165	1156	∅	∅
36	1050	1033	∅	1058	86	1156	1144	∅	∅
37	1058	1043	∅	∅	87	1144	1163	∅	∅
38	1043	1063	∅	∅	88	1149	1187	∅	∅
39	1063	1065	∅	∅	89	1187	1190	1187	∅
40	1065	1069	∅	∅	90	1190	1193	∅	∅
41	1039	1050	∅	1070	91	1193	1196	∅	∅
42	1070	1074	∅	∅	92	1196	1199	1196	∅
43	1074	1077	1074	∅	93	1199	1202	1199	∅
44	1077	1079	∅	∅	94	1202	1205	∅	∅
45	1079	1082	∅	∅	95	1205	1208	∅	∅
46	1082	1079	1082	∅	96	1199	1211	∅	∅
47	1079	1087	∅	∅	97	1211	1214	1211	∅
48	1087	1090	∅	∅	98	1214	1217	∅	∅
49	1082	1092	∅	∅	99	1217	1221	∅	∅
50	1092	1094	∅	∅	100	1211	1219	∅	∅

Tableau 6.12 EPS_0 avec stratégie $MPDS$ pour $T3$

itér.	p_0	p_1	p_2	p_3	itér.	p_0	p_1	p_2	p_3
1	1	2	3	4	51	1094	1070	1096	1058
2	5	6	7	8	52	1097	1094	1098	1096
3	6	7	8	12	53	1099	1099	1100	1094
4	13	6	7	8	54	1070	1100	1101	1024
5	7	8	13	6	55	1082	1102	1103	1101
6	14	21	22	7	56	1099	1070	1104	1054
7	24	14	21	237	57	1070	1082	1105	1049
8	28	29	24	8	58	1102	1099	1106	1024
9	32	987	988	39	59	1082	1070	1107	1058
10	36	32	990	613	60	1108	1082	1109	1110
11	40	991	36	8	61	1110	1112	1113	1114
12	42	993	994	27	62	1094	1116	1117	1118
13	36	995	996	45	63	1087	1119	1120	1121
14	46	997	998	48	64	1122	1123	1124	1110
15	49	999	1000	58	65	1094	1127	1128	1129
16	52	49	1002	27	66	1070	1130	1101	1132
17	1002	1003	52	49	67	1082	1094	1134	1135
18	1005	1006	1007	48	68	1134	1070	1136	1129
19	1008	1005	1010	8	69	1070	1138	1139	1140
20	1011	1012	1013	451	70	1102	1141	1134	1132
21	1014	1015	451	1011	71	1082	1143	1144	1145
22	1017	1018	831	1019	72	1099	1138	1147	1135
23	1014	49	1017	1022	73	1094	1148	1149	1150
24	1011	1023	1022	1025	74	1149	1150	1152	1153
25	1017	1026	1027	1022	75	1149	1155	1156	1149
26	1029	49	1031	1024	76	1158	1159	1160	1161
27	1014	1031	1024	1011	77	1149	1070	1163	1164
28	1033	1034	1031	1024	78	1156	1141	1165	1153
29	1036	1037	1033	1034	79	1166	1155	1156	1129
30	1039	1040	1041	1024	80	1149	1138	1144	1161
31	1043	1039	1040	1046	81	1165	1168	1163	1164
32	1047	1043	1048	1039	82	1156	1169	1168	1171
33	1050	1033	1051	1024	83	1152	1172	1173	1174
34	1033	1052	1053	1054	84	1172	1175	1176	1177
35	1043	1055	1056	1046	85	1165	1178	1179	1150
36	1050	1033	1057	1058	86	1156	1181	1182	1129
37	1058	1059	1060	1061	87	1144	1183	1184	1132
38	1043	1047	1062	1049	88	1149	1185	1186	1150
39	1063	1043	1024	1024	89	1187	1188	1189	1164
40	1065	1066	1067	1068	90	1190	1172	1192	1187
41	1039	1070	1071	1061	91	1193	1194	1195	1129
42	1070	1072	1024	1049	92	1196	1197	1198	1164
43	1074	1075	1076	1024	93	1199	1172	1201	1196
44	1077	1078	1074	1068	94	1202	1199	1204	1129
45	1079	1080	1046	1081	95	1205	1206	1207	1161
46	1082	1083	1058	1084	96	1199	1209	1176	1150
47	1079	1085	1082	1086	97	1211	1194	1212	1213
48	1087	1070	1054	1058	98	1214	1213	1211	1216
49	1082	1091	1046	1024	99	1217	1218	1176	1220
50	1092	1093	1024	1046	100	1211	1222	1223	1224

Tableau 6.13 Évolution de $T3$ avec la stratégie $MPDS$

6.3.4 Dynamique d'interaction récursive

Hypothèse 6.2 *Il existe un lien d'interaction entre les stratégies d'exploration des procédures par recherches multiples coopérantes basé sur la dynamique d'interaction récursive.*

Le prochain test vise à confirmer l'hypothèse 6.2. Pour cette expérience nous empêcherons les stratégies d'exploration des procédures séquentielles de $T3$ de s'actualiser à partir d'information en provenance de p_0, en empêchant p_0 d'envoyer des messages aux autres procédures séquentielles de $T3$ (p_0 envoie toujours le même message $\emptyset$). Dans ce test, p_0 peut lire les messages en provenance des autres procédures séquentielles et donc utiliser l'historique de recherche $\bar{h}_{ik}$. Nous montrerons que le parcours de p_0 est modifié par le fait que cette procédure séquentielle n'envoie plus de messages aux autres procédures, ce qui nous amènera à conclure que la différence entre les deux parcours de p_0 provient de l'élimination de la dynamique d'interaction récursive provoquée par le changement à p_0.

Le tableau 6.5 résume les résultats qui ont été obtenus. En comparant le tableau 6.5 avec les tableaux 6.1 et 6.2, on constate une fois de plus que la trajectoire de p_0 diffère entre les trois tableaux. Par exemple, 40 des solutions visitées dans la colonne x du tableau 6.5 sont différentes de celles du tableau 6.1 pour la même colonne x.

L'impact de l'absence d'envoi de messages par p_0 sur son propre parcours du graphe de transitions se fait sentir pour la première fois à l'itération 6 suite au message "8" reçu par p_0 à l'itération 5, qui diffère du message $\emptyset$ de la procédure $T3$ originale (tableau 6.1). Par la suite le parcours de p_0 diffère de celui du tableau 6.1 (sauf pour les itérations 19 et 20) soit parce que les solutions explorées ne sont pas les mêmes (dans 40% des cas) ou encore les solutions explorées diffèrent par l'itération où leur exploration s'effectue par rapport au tableau 6.1.

Comme c'était le cas pour les tableaux 6.3et 6.4, les tableaux 6.6 et 6.7 comparent le parcours de $T3$ sans modification à p_0 avec le cas où p_0 n'envoie que

itér.	x	x'	m_s	m_r	itér.	x	x'	m_s	m_r
1	1	5	∅	∅	51	1340	1343	1340	∅
2	5	9	∅	6	52	1343	1307	∅	∅
3	9	202	9	∅	53	1307	1347	∅	∅
4	202	203	202	7	54	1340	1350	∅	1349
5	203	205	203	∅	55	1350	1353	∅	∅
6	205	209	205	∅	56	1353	1254	∅	∅
7	209	210	209	∅	57	1340	1302	∅	∅
8	210	214	210	∅	58	1302	1359	∅	∅
9	214	218	214	∅	59	1359	1362	∅	∅
10	218	221	∅	∅	60	1362	1365	∅	∅
11	221	224	221	1238	61	1365	1302	∅	∅
12	224	226	224	∅	62	1302	1359	∅	1369
13	226	230	226	∅	63	1359	1350	∅	∅
14	230	231	∅	∅	64	1350	1365	∅	∅
15	231	1249	∅	∅	65	1340	1380	∅	∅
16	1249	1252	1249	∅	66	1380	1307	1380	∅
17	1252	1255	1252	∅	67	1307	1385	∅	∅
18	1255	1258	∅	∅	68	1385	1387	∅	∅
19	1258	1261	∅	∅	69	1387	1390	1387	∅
20	1252	1264	∅	∅	70	1390	1393	1390	∅
21	1264	1267	∅	64	71	1393	1380	∅	∅
22	1267	1271	∅	∅	72	1380	1399	∅	∅
23	1252	1275	∅	∅	73	1390	1403	∅	∅
24	1275	1278	1275	∅	74	1403	1407	∅	1404
25	1278	1281	∅	∅	75	1407	1411	1407	1410
26	1281	1283	∅	∅	76	1411	1415	∅	∅
27	1283	1286	∅	∅	77	1415	1407	∅	∅
28	1286	1289	1286	∅	78	1407	1411	∅	1421
29	1289	1291	∅	1290	79	1411	1424	∅	∅
30	1291	1294	∅	∅	80	1424	1415	∅	∅
31	1294	1275	∅	∅	81	1415	1428	∅	∅
32	1275	1298	∅	∅	82	1428	1411	∅	∅
33	1298	1289	∅	∅	83	1411	1390	∅	∅
34	1289	1291	∅	∅	84	1390	1424	∅	∅
35	1291	1298	∅	∅	85	1424	1411	∅	∅
36	1298	1302	∅	∅	86	1407	309	∅	∅
37	1302	1304	∅	∅	87	309	307	309	∅
38	1304	1307	∅	∅	88	307	1441	307	∅
39	1286	1310	∅	∅	89	1441	1442	∅	∅
40	1310	1313	∅	∅	90	1442	1444	∅	1443
41	1313	1316	∅	∅	91	307	1446	∅	∅
42	1286	1289	∅	∅	92	1446	1450	∅	∅
43	1289	1298	∅	∅	93	1450	307	∅	∅
44	1298	1294	∅	∅	94	307	1453	∅	∅
45	1294	1291	∅	∅	95	1453	1455	∅	∅
46	1291	1289	∅	∅	96	1455	1458	∅	∅
47	1289	1275	∅	∅	97	1458	1446	∅	∅
48	1275	1298	∅	∅	98	1446	1453	∅	∅
49	1298	1289	∅	∅	99	1453	1455	∅	∅
50	1286	1340	∅	∅	100	1455	1458	∅	∅

Tableau 6.14 EPS_0 avec stratégie $MPDS$ et sans lecture de m_r

iter.	p_0	p_1	p_2	p_3	iter.	p_0	p_1	p_2	p_3
1	1	2	3	4	51	1340	1341	1342	1319
2	5	6	7	8	52	1343	1300	1345	1333
3	9	7	8	12	53	1307	392	1305	1306
4	202	9	7	8	54	1340	1303	1348	1349
5	203	8	202	9	55	1350	1290	1351	1352
6	205	206	207	203	56	1353	1301	1354	532
7	209	205	206	1231	57	1340	1303	1355	1356
8	210	211	209	8	58	1302	1357	1342	1349
9	214	215	1234	210	59	1359	1308	1360	1361
10	218	214	1236	1237	60	1362	1300	1363	1364
11	221	1238	1239	8	61	1365	1366	1367	1349
12	224	1241	1242	221	62	1302	1369	1370	1366
13	226	224	1241	27	63	1359	1372	1373	1374
14	230	1245	226	39	64	1350	1375	1305	1373
15	231	226	1248	45	65	1340	1377	1378	1379
16	1249	1250	1251	223	66	1380	1381	1382	1383
17	1252	1253	1249	1250	67	1307	1380	1360	1374
18	1255	1256	1257	1252	68	1385	1380	1378	1376
19	1258	1259	1260	8	69	1387	1369	1389	1349
20	1252	1262	1263	12	70	1390	1387	1360	1392
21	1264	1265	12	64	71	1393	1392	1390	1395
22	1267	1268	1269	1270	72	1380	1369	1397	1398
23	1252	1250	1273	1274	73	1390	1400	1401	1402
24	1275	1276	1277	756	74	1403	1404	1405	1392
25	1278	1275	1276	709	75	1407	1408	1409	1410
26	1281	1281	226	1270	76	1411	1407	1413	1414
27	1283	1284	242	1285	77	1415	1414	1417	1418
28	1286	1275	247	1288	78	1407	1419	1420	1421
29	1289	1290	226	1270	79	1411	1422	1423	1418
30	1291	1292	242	1293	80	1424	1412	1425	1414
31	1294	1295	239	1296	81	1415	1426	1427	1421
32	1275	1297	247	1285	82	1428	1419	1429	1352
33	1298	1299	233	1293	83	1411	1430	1373	1418
34	1289	1300	242	1296	84	1390	1432	1433	1434
35	1291	392	239	1300	85	1424	1435	1436	1414
36	1298	1301	247	1293	86	1407	1437	1373	1352
37	1302	1303	226	1270	87	309	1439	1378	1418
38	1304	392	1305	1306	88	307	309	1433	1434
39	1286	1305	1306	599	89	1441	1404	307	1421
40	1310	1290	1311	1312	90	1442	1416	1378	1443
41	1313	1303	1314	1315	91	307	1445	1305	1414
42	1286	392	629	1306	92	1446	1404	1448	1449
43	1289	1318	831	1319	93	1450	1408	1449	1451
44	1298	1320	1321	1322	94	307	1416	1373	1452
45	1294	1323	1324	1306	95	1453	1412	1241	1454
46	1291	1326	1327	1328	96	1455	1241	1456	1449
47	1289	1329	1330	1319	97	1458	1419	1459	1456
48	1275	1331	1332	1333	98	1446	1416	1461	1462
49	1298	1300	1335	1336	99	1453	1412	1463	1449
50	1286	1337	1305	1339	100	1455	1404	1465	1466

Tableau 6.15 Évolution de $T3$ avec $MPDS$ et p_0 sans lecture de m_r

le message $\emptyset$ aux autres procédures séquentielles. Contrairement aux tableaux 6.3 et 6.4, cette fois le parcours de p_0 est le dernier à être affecté par la modification qu'a subie cette procédure. C'est p_1 qui est le premier touché à l'itération 4 suite à la non-réception du message "6" en provenance de p_0, dans ce cas p_1 a choisit la solution $x' = 14$ dans le voisinage de la solution courante "7" à l'itération 3. Les procédures séquentielles p_2 et p_3 sont touchées à l'itération 5. p_2 ne reçoit pas à l'itération 4 le message "13" et choisit alors la solution "19" dans le voisinage de la solution "7" de l'itération 4. p_3 choisit la solution "14" suite à la réception de ce message "14" à l'itération 4 en provenance de p_1 qui a envoyé ce message à la place du message "6" par suite de la modification de son parcours suivant la non-réception d'un message de p_0. Toutes ces modifications du parcours de p_1, p_2 et p_3 finissent par affecter p_0 à l'itération 6 suite à la réception du message "8" en provenance de p_1.

Dans ce test, le parcours du graphe de transitions effectué par les procédures séquentielles autres que p_0 est d'abord affecté par le fait que les stratégies d'exploration de ces procédures s'actualisent à partir de leur propre historique de recherche à la place de messages en provenance de p_0. Une fois que le parcours de ces procédures est modifié, la séquence de messages qu'elles transmettent n'est plus la même, y compris les messages envoyés à p_0. Conséquemment, l'information externe $\bar{h}_{0k}$ à partir de laquelle s_0 s'actualise n'est plus la même et change donc le parcours de p_0. Mais tout ce qui a changé au niveau de $T3$ c'est l'envoi de messages en provenance de p_0. La différence dans la colonne x de p_0 entre le présent test et la procédure $T3$ originale provient donc de l'absence de l'effet de la dynamique d'interaction récursive sur p_0. Donc, les messages qu'une procédure p_i envoie aux autres procédures ont un impact sur les messages que p_i reçoit des autres procédures séquentielles et par conséquent, sur le parcours du graphe de transitions accompli par la procédure p_i.

Cette dynamique d'interaction récursive est similaire au comportement de réactions en chaîne à l'intérieur d'une procédure séquentielle que nous avons décrit

à la section 6.3. À la différence cependant, que la dynamique de réactions en chaîne s'exécute entre le parcours d'une procédure p_i et l'ensemble global des connaissances acquises c'est-à-dire que par le biais de l'échange d'information, le parcours de la procédure p_i réagi aux connaissances acquises de toutes les procédures séquentielles. Cette expérimentation valide l'hypothèse selon laquelle il existe une dynamique d'interaction récursive entre les stratégies d'exploration et la possibilité pour une stratégie d'exploration s_i en interaction avec les autres stratégies d'exploration de mettre en branle un cycle complexe de modifications du parcours de p_i de l'espace de solutions.

6.4 Les effets du couplage et du nombre de procédures séquentielles

Les dynamiques d'interaction que nous venons de décrire montrent que l'échange d'information, même lorsqu'il ne provoque aucune uniformisation du contenu des historiques de recherche, engendre une interdépendance complexe du comportement d'exploration des procédures séquentielles entre elles. Nous montrerons à partir de ces deux patrons comment des facteurs comme le nombre de procédures séquentielles et la structure de voisinage, affectent le comportement global des recherche multiples coopérantes.

6.4.1 Impact de la structure de voisinage entre les procédures séquentielles

Le prochain test illustre les effets de la structure de voisinage sur le parcours du graphe de transitions accomplit par les procédures séquentielles. Nous changerons la structure de voisinage de la procédure $T3$ de $p_{(k+i \bmod p)}$ à $p_{(k+i+1 \bmod p)}$. La trajectoire de la procédure séquentielle p_0 suite à cette modification de la structure de voisinage de $T3$ apparaît dans le tableau 6.8. En comparant la colonne x de

ce tableau avec celle du tableau 6.1, on constate que seulement 6 solutions ont été visitées par les deux procédures p_0. Le tableau 6.9 montre le parcours de la procédure parallèle $T3$, on y observe le même type de changement de parcours pour toutes les procédures séquentielles. On voit donc qu'une procédure par recherches multiples coopérantes est très sensible à la définition de la structure de voisinage. Un changement dans la structure de voisinage modifie les séquences de messages que reçoivent les procédures séquentielles. Ces changements dans les séquences de messages sont amplifiés par les dynamiques d'interaction en chaîne et d'interaction récursive, ce qui se traduit en dernière analyse par une modification profonde de l'exploration de l'espace de solutions.

Nous avions émis l'hypothèse au chapitre 5, selon laquelle la différence des structures de voisinage (plus que la synchronisation) est responsable des différences de performance entre les stratégies de parallélisation p-KS et collégiale asynchrone. Rappelons que la stratégie de parallélisation p-KS est basée sur un échange synchrone de messages alors qu'une procédure parallèle collégiale asynchrone échange l'information entre les procédures séquentielles de manière asynchrone. Nous avions établi que le mode de synchronisation définit de manière indirecte des structures de voisinage avec des diamètres différents, un large diamètre pour les procédures synchrones et un faible diamètre pour les procédures asynchrones. En montrant que le comportement d'une procédure parallèle par recherches multiples coopérantes n'est pas le même si les structures de voisinage changent, l'expérimentation de la présente section renforce l'hypothèse du chapitre 5, selon laquelle les performances des procédures parallèles sont affectées par la structure de voisinage entre les procédures séquentielles.

6.4.2 Impact du nombre de procédures séquentielles

Les tests des sections 6.3.3, 6.3.4 et 6.4.1 vont nous permettre d'apprécier l'impact du nombre de procédures séquentielles sur le comportement dynamique de réaction

en chaîne entre les stratégies d'exploration.

Définition 6.9 *Soit $\mathcal{P}$ l'ensemble original de procédures séquentielles d'une procédure par recherches multiples coopérantes (dans le cas de T3, $\mathcal{P} = \{p_0, p_1, p_2, p_3, \}$). $\mathcal{P}^-$ l'ensemble de procédures séquentielles résultant d'une diminution du nombre de procédures dans $\mathcal{P}$ et $\mathcal{P}^+$ l'ensemble de procédures séquentielles résultant d'une augmentation du nombre de procédures dans $\mathcal{P}$. Soit $M^{\mathcal{P}}$ l'ensemble des messages échangés entre les procédures séquentielles de l'ensemble $\mathcal{P}$.*

Considérons le cas de la procédure parallèle $T3$ originale. Lorsque l'ensemble de procédures séquentielles passe de P à P^-, les procédures séquentielles de P^- ne recevront pas les messages $M^{\mathcal{P}\setminus\mathcal{P}^-}$ en provenance des procédures de $\mathcal{P}\setminus\mathcal{P}^-$. Or, nous avons vu à la section 6.3.3, que lorsque la colonne m_r d'une procédure séquentielle est modifiée (suite ici à la non réception des messages $M^{\mathcal{P}\setminus\mathcal{P}^-}$, le parcours de l'espace de solutions exécuté par la procédure change également. Un phénomène similaire se produit pour $\mathcal{P}^+$ où cette fois les procédures de $\mathcal{P}$ reçoivent de nouveaux messages en provenance des nouvelles procédures séquentielles ajoutées à $T3$.

Soulignons qu'une variation dans le nombre de procédures séquentielles engendre nécessairement un changement dans la structure de voisinage, ce qui induit les comportements que nous avons observés à la section 6.4.1. Dans tous les cas, les procédures séquentielles de $\mathcal{P}$ ou $\mathcal{P}^-$ verront leurs colonnes m_s et m_r être fortement perturbées par le changement du nombre de procédures. Ultimement, les comportements dynamiques de réaction en chaîne observés dans les sections 6.3.3 et 6.3.4 vont agir sur la procédure parallèle et transformer profondément l'exploration de l'espace de solutions effectuée par cette dernière.

Le tableau 6.10 montre comment p_0 est affectée par l'ajout d'une seule nouvelle procédure séquentielle par recherches multiples coopérantes $T3$ (p_0 peut lire et envoyer des messages). Le tableau 6.11 montre le parcours de $T3$ avec les 5 procédures séquentielles; on constate que seul p_1, jusqu'à l'itération 34, effectue une exploration dans la région explorée par $T3$ avec 4 procédures.

6.4.3 Impact des paramètres de la stratégie d'exploration

Dans cette section nous mesurons l'impact sur $T3$ de changements aux paramètres de recherche des procédures séquentielles. Nous avons testé des modifications simples aux paramètres de recherche de p_0 tel des changements dans la taille des listes tabou à court terme. Ces changements n'ont eu que très peu d'impact sur le parcours de p_0 et conséquemment sur celui de $T3$. Notons que nous n'avons pas fait de tests au niveau de la diversification, la procédure $T3$ fait une diversification forcée à chaque fois qu'un message non vide arrive dans la colonne m_r des messages reçus. On peut cependant déduire qu'un changement de diversification affecterait fortement le comportement de $T3$ puisque ce type de changement implique une lecture différente de la séquence de messages. Or, si les messages ne sont pas lus de la même manière, ceci équivaut à changer la colonne m_r, et nous savons qu'un tel type de changement affecte $T3$.

Par contre, des modifications à la stratégie de différentiation de l'ensemble des procédures séquentielles ont un impact important. Le tableau 6.12 montre le parcours de p_0 suite à un changement de stratégie de différentiation qui passe de $MPSS$ à $MPDS$ pour les quatre procédures séquentielles impliquées dans la procédure parallèle $T3$. Comme on peut le voir, il en résulte un parcours différent par p_0 en comparaison avec l'ensemble de solutions visitées lorsque la stratégie $MPSS$ était appliquée. Le tableau 6.13 montre le parcours du graphe de transitions de $T3$ avec $MPDS$ comme stratégie de différentiation. Le domaine d'exploration de $T3$ avec $MPSS$ s'étendait de 1 à 199 (voir les tableaux 6.3 et 6.4), tandis qu'avec $MPDS$ il va de 1002 à 1224 sauf pour un recouvrement partiel des solutions explorées avec la stratégie $MPSS$ dans les phases initiales d'exploration.

Nous avons testé si les observations qui ont été faites dans les dernières sections restent valides pour d'autres stratégies de différentiation de la recherche tabou. Les tableaux 6.14 et 6.15 montrent l'impact sur p_0 et sur la procédure parallèle $T3$ avec $MPDS$ lorsque p_0 ne lit pas les messages en provenance des autres

procédures. On constate que des bouleversements similaires à ceux observés pour le traitement avec $MPSS$ résultent d'un changement dans la séquence de messages reçus par une procédure. D'autres tests nous confirment que les mêmes comportements observés pour la procédure parallèle $T3$ $MPSS$ se manifestent lorsqu'on utilise d'autres paramètres de recherche pour les procédures séquentielles.

6.5 Conclusion

Les tests effectués dans ce chapitre montrent très clairement que la stratégie d'exploration est loin d'être le seul facteur déterminant le parcours de l'espace de solutions des recherches multiples coopérantes. Le partage des connaissances entre les procédures séquentielles rend le comportement d'exploration de chaque procédure séquentielle dépendant de facteurs comme le comportement des autres procédures, le nombre de procédures et la structure de voisinage.

Ces facteurs interfèrent avec la méthode de recherche pour définir, avec les stratégies d'exploration, la recherche exécutée par chaque procédure séquentielle. Le questionnement qui suit consiste à se demander dans quelle direction au sens de la logique d'optimisation d'une méthode de recherche, ces facteurs influencent-ils le progrès de l'exploration de l'espace de solutions? Par exemple, de quelle façon une interaction en chaîne change-t-elle l'état de la recherche, dans quelle direction l'exploration d'une procédure séquentielle se trouve-t-elle propulsée après avoir été impliquée dans une dynamique de réactions en chaîne? Quel est l'effet de ces différents facteurs sur les performances d'une procédure parallèle par recherches multiples coopérantes? Le prochain chapitre apporte un premier niveau de réponses à ces questions.

Chapitre 7

Analyse de l'interaction entre les stratégies d'exploration

7.1 Introduction

Nous savons que les performances d'une procédure parallèle par recherches multiples dépendent du nombre de procédures séquentielles et des stratégies d'exploration. Par exemple, nous savons que si l'on augmente le nombre de procédures séquentielles, les performances de la procédure parallèle seront aussi bonnes ou meilleures que sans l'augmentation. Nous avons vu au chapitre 4, que ce n'est pas le cas pour les stratégies de parallélisation avec recherches multiples coopérantes. Plusieurs résultats affichent de moins bonnes performances suite à une augmentation du nombre de procédures séquentielles. Au chapitres 5 et 6, nous avons montré que l'échange d'information interfère de manière importante avec les stratégies d'exploration de la méthode tabou, indiquant que les performances des recherches multiples coopérantes ne sont pas liées uniquement à la méthode de recherche.

Nous avons vu qu'il existe des modèles qui tentent d'expliquer l'impact de l'échange d'information sur le comportement d'exploration des procédures séquentielles des recherches multiples coopérantes. Les modèles proposés par Huberman [56], Clearwater, Huberman & Hogg [25], Clearwater, Hogg & Huberman [24] reposent sur l'ensemble suivant d'hypothèses dont certaines sont inhérentes à l'utilisation d'un modèle probabiliste:

1. La probabilité de succès de chaque procédure séquentielle suit une distribution géométrique (voir Battiti [10] et Taillard [93]);

2. Le contenu des messages est indépendant du comportement des procédures séquentielles;

3. Les messages ne sont pas corrélés entre eux (tous les messages portent sur une sous-région différente de l'espace de solutions);

4. Une solution n'est explorée que par une seule procédure séquentielle (diversité parfaite des stratégies d'exploration);

5. Les procédures séquentielles sont indépendantes les unes des autres (au sens probabiliste).

Il est clair selon le contenu du chapitre précédent, que les hypothèses d'indépendance des procédures séquentielles les unes par rapport aux autres (hypothèse 5) et d'indépendance des messages par rapport au comportement des procédures séquentielles (hypothèse 2) ne sont jamais satisfaites. Le fait que ces deux hypothèses ne soient pas satisfaites rend ces modèles peu fiables dans leur approximation du comportement d'une procédure avec recherches multiples coopérantes. Cette lacune est connue des auteurs de ces modèles. Par exemple dans Clearwater, Hogg & Huberman [24], les messages sont donnés avant que l'exécution de la procédure parallèle ne commence. Les messages sont envoyés aux procédures séquentielles à partir d'un processus externe au traitement parallèle pour que puissent être satisfaites les hypothèses 2 et 5.

Il existe à notre avis, une autre hypothèse sous-jacente à tous les modèles actuels du traitement parallèle avec recherches multiples coopérantes. Cette autre hypothèse consiste à penser que, globalement, soit que l'échange d'information suit la même logique d'optimisation que les méthodes de recherche ou qu'il n'a aucun impact sur cette logique. Relativement à l'interrogation du chapitre précédent: "par rapport à la logique d'optimisation d'une méthode de recherche, dans quelle direction l'exploration d'une procédure séquentielle se trouve-t-elle propulsée après avoir été impliquée dans une dynamique de réactions en chaîne", cette hypothèse

offre deux catégories de réponses. La première catégorie suppose, a priori, que les réactions en chaîne vont parfois aider et parfois nuire à l'exploration d'une procédure séquentielle, mais globalement l'impact des réactions en chaîne est nul sur l'exploration des procédures séquentielles. Les réponses de la deuxième catégorie découlent implicitement du raisonnement suivant: puisque l'information échangée entre les procédures séquentielles porte sur des données de l'espace de solutions (par exemple il n'y a pas de solutions non réalisables échangées), on ne voit pas comment les méthodes de recherches pourraient être systématiquement détournées de la logique d'optimisation de la méthode de recherche. Par exemple, on n'imagine pas qu'une méthode de recherche locale puisse cesser de s'orienter vers les optima locaux sous l'effet d'échanges d'information avec d'autres recherches locales. Donc, on fait l'hypothèse que les méthodes de recherche et l'échange d'information vont dans la même direction, celle de la logique d'optimisation!

Un résultat assez important du développement théorique de ce chapitre sera de montrer que ces réponses sont à tout le moins partiellement fausses. Nous montrerons que l'échange de connaissances acquises entre les procédures séquentielles ne sert pas uniquement à la logique d'optimisation de la méthode de recherche, mais que cet échange sert également à coordonner les recherches entre elles selon une logique qui n'a parfois rien à voir avec la logique d'optimisation. Ce faisant, nous mettrons à jour certaines règles qui gouvernent les procédures parallèles avec recherches multiples coopérantes.

Dans ce chapitre, nous concevrons plusieurs procédures parallèles avec recherches multiples coopérantes. Pour ces procédures, l'information échangée entre les procédures séquentielles sera générée à partir de lois de la théorie qualitative des systèmes dynamiques [3, 5, 35, 69, 76, 83, 86] et non pas à partir de connaissances acquises sur l'espace de solutions. Nous remplacerons donc l'échange d'information basé sur une logique d'optimisation, par un échange d'information basé sur les lois qui modèlisent l'évolution dans le temps d'un système dynamique.

C'est donc dire que le progrès d'une procédure parallèle avec recherches multiples coopérantes sera analysé comme étant celui de l'évolution d'un système dynamique et que l'information échangée sera significative selon cette interprétation de la recherche parallèle. Nous montrerons ensuite que sous certains aspects, ces nouvelles procédures parallèles se comportent de la même manière que les procédures parallèles s'échangeant des connaissances sur l'espace de solutions. À partir de ces résultats, il nous sera possible d'affirmer que les procédures parallèles avec recherches multiples coopérantes se comportent comme des systèmes dynamiques complexes, où le parcours de l'espace de solutions est déterminé partiellement par les lois d'interaction entre les variables d'un système dynamique. Le contenu des messages est interprété globalement à la fois selon un modèle dynamique par les facteurs qui interfèrent avec la méthode de recherche et selon une logique d'optimisation par la méthode de recherche. L'objectif des nouvelles procédures introduites dans ce chapitre, en particulier du modèle dynamique de l'échange d'information, sera d'interpréter correctement la dérive de l'exploration de l'espace de solutions provoquée par l'interférence des recherches multiples coopérantes sur le déroulement normal de la méthode tabou.

À l'aide de ce modèle, nous étudierons deux comportements particuliers des procédures avec recherches multiples coopérantes: la détérioration des performances que l'on observe parfois suite à une augmentation du nombre de processeurs et l'hypothèse selon laquelle l'interaction entre les stratégies d'exploration engendre un comportement d'auto-organisation de l'exploration de l'espace de solutions.

La détérioration des performances lorsque le nombre de processeurs augmente est un comportement assez inusité au niveau du traitement parallèle et constitue un aspect plutôt ennuyeux des recherches multiples coopérantes. L'auto-organisation est un phénomène qui apparaît sous la forme du développement d'une structure (attracteur, synergétique, catastrophe) qui n'est pas dirigé ou organisé par un autre système et se manifeste par une réduction de l'entropie du système dynamique [86].

Des phénomènes d'auto-organisation ont été décrits dans des domaines aussi variés que la physique (formation de patrons, dynamique non linéaire, mécanique statistique, physique des plasmas, etc), en mathématique, en biologie, en génie, en théorie des systèmes complexes et même en sciences sociales. S'il y a auto-organisation au niveau des recherches multiples coopérantes, les principes de ce type de comportement pourraient nous aider à mieux comprendre l'interaction entre les stratégies d'exploration et possiblement nous permettre de mieux contrôler l'évolution dynamique de ce type de procédures parallèles.

Ce chapitre est organisé de la manière suivante. Dans la prochaine section, nous introduirons certaines notions sur la théorie des systèmes dynamiques et présenterons en détail le modèle de système dynamique qui sera utilisé pour remplacer l'échange de connaissances acquises entre les procédures séquentielles. À la section suivante, nous montrerons à l'aide de ce modèle, pourquoi la meilleure solution d'une procédure avec recherches multiples coopérantes peut être moins bonne lorsque le nombre de processeurs augmente. Enfin, nous montrerons que l'hypothèse d'auto-organisation de la recherche se vérifie au niveau de la procédure de traitement parallèle collégiale asynchrone du chapitre 4.

7.2 Théorie des systèmes dynamiques

L'évolution d'un système dynamique est définie par une ensemble de *variables d'état* et par une *loi dynamique* qui gouverne le changement d'état dans le temps des variables du système. Nous nous intéresserons à des modèles où le temps, les variables d'état et les paramètres du système correspondent à des valeurs discrètes. Dans ce cadre, un modèle général de système dynamique est donné par:

$$E(k+1) = \phi[E(k), P, k]$$

où:

$$
\begin{aligned}
E &= \{x_1, x_2, \ldots, x_p\} \\
\phi &= \{f_1, f_2, \ldots, f_p\} \\
P &= \{P_1, P_2, \ldots, P_m\}
\end{aligned}
$$

où E est le vecteur des variables d'état du système, ϕ est le vecteur de fonctions de la loi dynamique et P un ensemble de paramètres [3, 5, 35, 69, 76, 83, 86]. Dans le contexte de la procédure parallèle $T3$ du précédent chapitre, E correspond à l'état de la procédure parallèle $T3$ à l'itération k, f_i décrit l'interaction existant entre la stratégie d'exploration s_i et les autres stratégies d'exploration du système ($T3$) et P correspond au couplage entre les procédures séquentielles. Ce système décrit le comportement dynamique de E à partir de son *état initial* $E_0 = \{x_{01}, x_{02}, \ldots, x_{0p}\}$. L'ensemble de toutes les valeurs possibles que peuvent prendre les variables d'état constitue l'*espace d'état* (state space) du système.

Un système dynamique est dit *autonome* si son évolution dépend de l'environnement externe uniquement pour l'état initial et que pour le reste, l'évolution dépend de l'interaction entre les variables du système sans apport externe. Dans le cas contraire le système est dit *non autonome*. Pour les recherches multiples coopérantes, l'interaction entre les stratégies d'exploration n'est pas le seul facteur qui détermine le parcours de l'espace de solutions, l'actualisation à partir d'information locale a aussi un impact. Le changement d'états d'une procédure avec recherches multiples coopérantes correspond donc à un système dynamique non autonome.

Battiti [11] a conçu un modèle de l'évolution d'une procédure séquentielle de la recherche tabou à partir d'un système dynamique autonome. Les variables d'état correspondent aux variables de décisions de la fonction objective, l'espace d'états correspond à l'espace de solutions réalisables du problème et la loi dynamique ϕ est représentée par la stratégie d'exploration de la procédure séquentielle. La dynamique de la recherche tabou séquentielle est alors parfaitement décrite par l'application de sa stratégie d'exploration et son historique de recherche. Une procédure séquentielle de la méthode tabou est dans ce cas un système dynamique autonome.

À partir d'un état initial E_0, la séquence d'états du vecteur E générée par l'action de ϕ correspond à la *trajectoire* du système dynamique. Le comportement à long terme de la trajectoire du système peut diverger à l'infini ou converger vers un attracteur. Un *attracteur* correspond à un ensemble d'états du système qui est invariant en termes de ϕ, c'est-à-dire que si la trajectoire du système emprunte un état appartenant à un attracteur, l'action du système dynamique va rester dans cet ensemble indéfiniment. Si le système est perturbé à une distance suffisamment petite d'un attracteur, l'action de la loi dynamique ϕ va ramener le système dans l'attracteur. L'ensemble des états initiaux convergeant vers un attracteur est appelé *bassin d'attraction* de l'attracteur. Si le système dynamique est initialisé dans un état qui appartient au bassin d'attraction d'un attracteur, le système évoluera vers cet attracteur. Les états par lesquels évolue une trajectoire qui n'appartiennent pas à l'attracteur lui même sont appelés *transients*. Un attracteur peut consister d'un seul état, on réfère à ce type d'attracteurs comme étant un *point fixe* ou *état stationnaire*, ou d'une succession périodique d'états (*cycle périodique*), ou encore il peut avoir une structure plus complexe (*attracteur chaotique*).

En général, l'espace des états d'un système dynamique possède plusieurs attracteurs, chacun entouré de son bassin d'attraction. Les attracteurs sont considérés comme une manifestation de phénomènes d'auto-organisation au niveau des systèmes dynamiques. Lorsqu'un système évolue dans le temps vers un attracteur, son entropie diminue ce qui est considéré comme une indication d'un accroissement de l'organisation du système. Ici la notion d'entropie $\mathcal{E}$ est définie de manière habituelle, c'est-à-dire que $\mathcal{E}$ est le logarithme du nombre en moyenne d'états possibles d'un système

$$\mathcal{E} = \sum_i p_i \log_2 p_i$$

où p_i est la probabilité de l'état i.

7.2.1 Théorie des automates cellulaires

Nous utiliserons l'automate cellulaire comme modèle mathématique pour simuler le comportement dynamique des recherches multiples coopérantes. L'automate cellulaire est un outil souvent utilisé d'étude et de simulation des mécanismes de base des systèmes dynamiques [17, 65, 101] . Un automate cellulaire analyse un système dynamique dans la perspective de l'interaction entre ses variables, c'est-à-dire comme s'il s'agissait d'un système autonome. D'après Wolfram [105, 106, 107], bien que de construction simple, les automates cellulaires sont capables de simuler des comportements dynamiques d'une grande complexité.

Définition 7.1 *Formellement, un automate cellulaire est une matrice où chaque entrée correspond à une variable du système dynamique. Chaque variable peut être considérée comme étant un automate à états fini, défini par le triplet* (K, Σ, ϕ) *où*

- K *est l'ensemble des états de l'automate fini;*
- Σ *est un alphabet;*
- ϕ *est la fonction de transition* $K^N \rightarrow K$ *de l'automate fini.*

Dans sa forme canonique, un automate cellulaire correspond à une machine parallèle abstraite synchrone et déterministe (voir la section 2.2.1), c'est-à-dire que (1) les changements d'état des variables du système dynamique s'exécutent de manière synchrone et que (2) la fonction ϕ est déterministe.

Définition 7.2 *Le* **voisinage** V *d'une variable* x_i *est un ensemble de* N *variables d'une sous-matrice de l'automate cellulaire ayant pour centre la variable* x_i.

À chaque itération, l'état de chaque variable x_i est fonction de l'état des variables de son voisinage. Par convention, une variable est considérée être son propre voisin.

Définition 7.3 *L'état du voisinage d'une variable x_i est le produit vectoriel de l'état des variables dans le voisinage de x_i.*

L'alphabet Σ correspond à l'ensemble des états possibles du voisinage $\mathcal{V}$. La taille de Σ est donnée par $|\Sigma| = |K|^N$.

Définition 7.4 *Une fonction de transition ϕ est définie en associant un seul état de K à chaque état possible du voisinage.*

Puisqu'il y a $|K|$ choix d'états possibles à associer pour chacun des K^N états du voisinage, on peut définir $|K|^{|K|^N}$ fonctions de transition ϕ différentes. On utilisera la notation $\mathcal{D}_N^K$ pour référer à l'ensemble de toutes les fonctions de transition possibles étant donné le voisinage de $|\mathcal{V}|$ voisins et $|K|$ états pour chaque variable. La fonction de transition ϕ représente l'interaction entre une variable x_i les autres variables du système dynamique. Dans le cas d'un automate cellulaire, cette relation est la même pour toutes les variables x_i.

Définition 7.5 *L'état d'un automate cellulaire à l'itération k est fonction de son état à l'itération $k-1$ et de l'application simultanée de la règle de transition ϕ à chaque variable de la matrice.*

L'automate cellulaire élémentaire

Nous nous intéresserons plus particulièrement aux *automates cellulaires élémentaires* [31, 105, 106]. Il s'agit d'un automate cellulaire limité à une seule ligne de la matrice de l'automate cellulaire canonique. Pour l'automate cellulaire élémentaire, $K = 2$, c'est-à-dire que chaque variable ne peut prendre que les valeurs 0 ou 1. La loi dynamique de l'automate cellulaire élémentaire est donnée par la fonction:

$$x_i^k = \phi[x_{i-r}^{k-1}, x_{i-r+1}^{k-1}, \cdots, x_i^{k-1}, \cdots, x_{i+r}^{k-1}]$$

où x_i^k dénote l'état de la variable x_i à l'itération k. Le paramètre r définit la distance des entrées du vecteur de l'automate cellulaire élémentaire pouvant faire partie du

voisinage d'une variable x_i. Lorsque $r = 1$, seules les variables adjacentes à x_i, c'est-à-dire x_{i-1}, x_i, x_{i+1} peuvent former le voisinage de x_i, conséquemment $N = 3$.

Définition 7.6 *Le statut d'une variable x_i à l'itération k de l'automate cellulaire élémentaire dépend du statut de $2r + 1$ variables du voisinage à l'itération $k - 1$.*

Définition 7.7 *Le nombre de variables pouvant être affectées par le statut d'une variable donnée x_i croît au plus par r entrées du vecteur de l'automate cellulaire élémentaire dans chaque direction à chaque itération de l'automate cellulaire. Après k itérations, une région d'au plus $1 + 2rk$ entrées peut avoir été affectée par l'état initial de x_i.*

Définition 7.8 *L'***exposant de Lyapunov** λ *mesure la vitesse de propagation de l'information reliée aux conditions initiales de l'automate cellulaire. L'automate cellulaire élémentaire ne possède que deux exposants de Lyapunov mesurant la propagation de l'information à droite et à gauche d'une entrée donnée du vecteur. La valeur de $\lambda \leq r$ (voir Wolfram [107]).*

L'exposant de Lyapunov λ est un concept de la théorie des systèmes dynamiques qui mesure le taux de divergence entre deux points rapprochés dans un système dynamique. De manière très générale. l'exposant de Lyapunov (maximal) représente le temps en moyenne du taux de croissante logarithmique de la distance entre deux points rapprochés. Un exposant de Lyapunov positif indique une dépendance sensible sur les conditions initiales puisque la distance croît de manière exponentielle en fonction du temps.

Étant donné cette configuration $|K| = 2$ et $r = 1$, la cardinalité de l'alphabet de l'automate cellulaire élémentaire est $|\Sigma| = |K|^{2r+1} = 2^3 = 8$ et la cardinalité de l'ensemble des règles de transitions $|\phi| = |K|^{|K|^{2r+1}} = 2^8 = 256$. Une règle de transition est définie en spécifiant l'état de la variable x_i^k pour chacun des 8 éléments $x_{i-1}^{k-1}, x_i^{k-1}, x_{i+1}^{k-1}$ de l'alphabet Σ. Le tableau 7.1 représente l'énoncé d'une règle de transition de l'automate cellulaire élémentaire.

111	—	0
110	—	1
101	—	0
100	—	0
011	—	1
010	—	1
001	—	0
000	—	0

Tableau 7.1 Règle de transition 76 de l'automate cellulaire élémentaire

Les valeurs binaires se trouvant à droite de chaque flèche correspondent à la séquence de 8 bits 01001100. La conversion de cette séquence dans sa valeur équivalente en base 10 donne le numéro de la règle (01001100 = 76 pour la règle dans le tableau 7.1), nous utiliserons ce numéro pour identifier les règles de transition de l'automate cellulaire élémentaire. Wolfram a étudié en détail le comportement qualitatif de la dynamique de ces règles. Par exemple, il a analysé leur structure espace-temps, c'est-à-dire l'évolution de 0 et de 1 d'un vecteur initial à travers plusieurs itérations. Il a regroupé ces règles dans les quatre classes suivantes:

- Classe 1: Inclut les règles qui après un certain nombre d'itérations de l'automate cellulaire élémentaire atteignent un point fixe et homogène (toutes les entrées du vecteur ont la même valeur 0 ou 1) qui est indépendant des conditions initiales de l'automate cellulaire. C'est le cas des règles 0, 8, 32, 40, 128 et 136.

- Classe 2: Inclut les règles qui après un transient initial relativement court entrent dans un cycle qui montre des configurations de 0 et de 1 constantes ou périodiques dans des régions séparées du plan espace-temps, les périodes sont $\leq 2^p$. C'est le cas par exemple pour les règles 4, 12, 37, 56, 73, 74 et plusieurs autres. Ces cycles ne sont cependant pas indépendants des conditions initiales mais les structures des configurations le sont relativement. Pour cette classe et la classe 1, l'exposant de Lyapunov $\lambda = 0$, ce qui implique que l'information sur les conditions initiales reste localisée et que le statut d'une entrée particulière

à tout moment dans le temps dépend des conditions initiales dans un intervalle fixe.

- Classe 3: Inclut des règles qui à partir d'un état initial aléatoire donnent naissance à des structures chaotiques, c'est-à-dire qui n'ont aucune périodicité $\leq 2^p$. C'est le cas par exemple pour les règles 18, 22, 30, 45, 146 et d'autres encore. Les structures de cette classe sont sensibles aux changements de conditions initiales sur une région toujours plus grande en fonction du nombre d'itérations. En effet, la classe 3 possède des exposants de Lyapunov λ positifs ce qui implique qu'une légère différence dans les conditions initiales va se propager sur tout le vecteur de l'automate cellulaire élémentaire et que le statut d'une entrée particulière, après un nombre suffisant d'itérations, dépend de manière critique sur les valeurs initiales.

- Classe 4: Inclut des règles très complexes et irrégulières avec des structures qui se propagent dans le temps et l'espace. C'est le cas des règles 60 et 110. La classe 4 a des exposants de Lyapunov λ positifs qui cependant tendent vers zéro de manière asymptotique. Les structures espace-temps sont affectées de manière significative par les conditions initiales, les changements de conditions initiales provoquent des effets irréguliers sur celles-ci.

7.3 Simulation des recherches multiples coopérantes

Nous allons maintenant définir une procédure parallèle avec recherches multiples coopérantes où l'interaction entre les procédures séquentielles sera formellement définie. Nous identifierons par SIM cette procédure parallèle. Le contenu exact de chaque message envoyé par une procédure séquentielle p_i de SIM sera déterminé à la fois par l'état de la recherche de p_i, la structure de voisinage entre les procédures séquentielles et la loi dynamique qui simule l'impact des facteurs d'interférence sur la recherche tabou. L'interaction entre les procédures séquentielles

de cette procédure de simulation sera plus générale que les procédures des chapitres précédents. Dans le modèle d'interaction entre les procédures séquentielles de la procédure SIM, l'échange de connaissances acquises sur l'espace de solutions sera considéré comme un cas particulier de l'échange d'information entre les procédures séquentielles interprétées comme étant des variables d'un système dynamique complexe.

Nous définirons de manière rigoureuse la procédure SIM puisqu'en plus d'être une procédure parallèle, SIM servira d'outil mathématique pour obtenir et démontrer certains résultats sur les causes de la dégradation des performances des procédures avec recherches multiples coopérantes. La procédure SIM servira également pour simuler certains comportements spécifiques des recherches multiples coopérantes. Lorsqu'utilisée comme outil de simulation, la procédure SIM servira à reproduire l'interférence des recherches multiples coopérantes sur le déroulement de la méthode tabou.

7.3.1 Définition de la procédure parallèle SIM

Chaque procédure séquentielle de SIM va consister d'une recherche tabou telle que définie à la section 4.5. les règles de transition de la méthode tabou sont les mêmes que celles de la section 4.5, MPDS sera la stratégie de différentiation de cette procédure parallèle. Nous supposerons que l'exploration de l'espace de solutions est effectuée par p procédures séquentielles en t itérations. Puisque les messages échangés entre les procédures séquentielles ne portent pas nécessairement sur l'espace de solutions, les procédures séquentielles pourront faire de l'exploration en dehors de l'espace de solutions, conséquemment l'espace de configurations de $SIM, \mathcal{C} \subseteq B^n$ inclus des solutions non réalisables.

Le couplage entre les procédures séquentielles est défini de la manière suivante: M est l'ensemble des messages pouvant être échangés entre les procédures séquentielles, $M \subseteq B^n$, $|M| = 2^n - 1$, $m_k \in M$. L'échange de messages se fait

de manière synchrone. Le réseau d'échange d'information entre les procédures séquentielles sera constitué à partir d'une architecture de type blackboard telle que décrite au chapitre 4. Nous identifierons par δ le nombre d'itérations entre deux points d'interaction, nous ferons l'hypothèse que $\delta = 25$, c'est-à-dire qu'il y a échange d'information à toutes les 25 itérations des procédures séquentielles. Lorsqu'arrivée à un point d'interaction, chaque procédure séquentielle p_j envoie à la mémoire centrale le contexte C_j des variables de décision de sa meilleure solution à moyen terme (voir définition 4.6). Les contextes C_j sont des vecteurs de variables binaires de taille n. Sur réception, la mémoire centrale fait la concaténation des p vecteurs C en un seul vecteur E_0 de taille $p \times n$. La concaténation des contextes C_j est telle que l'entrée $C_j(k)$ correspond à l'entrée $E_0((j \times n) + k)$, c'est-à-dire que les contextes C_j sont ordonnés dans le vecteur E_0 en fonction de l'indice j de la procédure séquentielle p_j.

La mémoire centrale est un automate cellulaire élémentaire. Le vecteur E_0 représente l'état initial de cet automate cellulaire. Lorsque E_0 a atteint la taille $p \times n$, l'exécution de l'automate cellulaire élémentaire se déclenche, appliquant une règle de transition ϕ de l'automate cellulaire sur E_0 pour $\tau = \lceil \frac{2n-1}{2r} \rceil$ itérations. La valeur de τ est choisie en fonction de l'exposant de Lyapunov λ le plus élevé parmi les règles de transition ϕ testées par SIM. τ correspond au nombre d'itérations nécessaire pour que le statut de $C_j(k)$ puisse avoir un impact sur la région de E_0 se situant dans le voisinage de la procédure séquentielle p_j. La structure de voisinage entre les procédures séquentielles est donc implicitement définie par la valeur de τ et l'ordonnancement des contextes C_j dans E_0. En fixant la valeur de τ à $\lceil \frac{2n-1}{2r} \rceil$, les conditions initiales du contexte C_j vont influencer uniquement l'état des contextes C_{j-1} et C_{j+1}. Le voisinage d'une procédure séquentielle p_j correspond donc aux procédures séquentielles p_{j-1} et p_{j+1}.

Lorsque les τ itérations ont été exécutées, la mémoire centrale retourne à la procédure séquentielle p_j un contexte C_j correspondant aux entrées

$[(j \times n)+0, (j \times n)+1, \ldots, (j \times n)+n]$ du vecteur $E_{(\lceil \frac{2n-1}{2r} \rceil - 1)}$, c'est-à-dire les entrées correspondant au contexte C_j du vecteur E_0. Puisque l'intervalle δ entre chaque point d'interaction est de 25 itérations et que $t = 300$, il y aura $\frac{t}{\delta} = 12$ exécutions de l'automate cellulaire élémentaire pour chaque exécution d'une procédure parallèle SIM. À chaque simulation, la même règle de transition ϕ sera appliquée à chaque point d'interaction. Pour certains tests, nous utiliserons des valeurs de τ définissant d'autres structures de voisinage, par exemple $\tau = \lceil \frac{(p \times n)-1}{2r} \rceil$ pour obtenir une propagation des conditions initiales qui porte sur tout le vecteur E_0.

Nous identifierons par SIM^{2^i} une procédure parallèle SIM où le nombre de procédures séquentielles est égal à 2^i et par SIM_ϕ une procédure parallèle SIM dont la mémoire centrale applique la règle de transition ϕ de l'automate cellulaire élémentaire.

Définition 7.9 *Les conditions limites de la structure de voisinage* $\{p_{j-1}, p_j, p_{j+1}\}$ *des procédures parallèles* SIM^{2^i} *sont résolues de la manière suivante: la structure de voisinage de* p_0 *est donnée par* $\{p_{2^i-1}, p_0, p_1\}$ *tandis que celle de* p_{2^i-1} *est donnée par* $\{p_{2^i-2}, p_{2^i-1}, p_0\}$.

Définition 7.10 *Toute procédure parallèle* $SIM_\phi^{2^{i+1}}$ *est construite récursivement à partir de la procédure parallèle* $SIM_\phi^{2^i}$ *en brisant le lien de la structure de voisinage entre les procédures séquentielles* p_0 *et* p_{2^i-1} *de* $SIM_\phi^{2^i}$ *et en introduisant* 2^i *nouvelles procédures séquentielles entre* p_0 *et* p_{2^i-1}.

Étant donnée la construction d'une procédure parallèle $SIM_\phi^{2^{i+1}}$, les stratégies d'exploration seront ordonnées de la manière suivante pour $SIM_\phi^{2^{i+1}}$: les procédures séquentielles 0 à $2^i - 1$ utilisent dans le même ordre les stratégies d'exploration 0 jusqu'à $2^i - 1$ de la procédure parallèle $SIM_\phi^{2^i}$ suivies des 2^i nouvelles stratégies d'exploration. Cette construction des procédures parallèles SIM vise à ne pas redéfinir un ensemble complet de 2^{i+1} nouvelles stratégies d'exploration à chaque fois que le nombre de procédures séquentielles change, par exemple lors du passage

d'une procédure parallèle avec 2^i procédures séquentielles à celui d'une procédure parallèle avec 2^{i+1} procédures séquentielles.

Lemme 7.1 *La structure de voisinage entre les procédures séquentielles p_1 à p_{2^i-2} est identique pour les procédures parallèles $SIM_\phi^{2^i}$ et $SIM_\phi^{2^{i+1}}$.*

Preuve: Puisque l'introduction des 2^i nouvelles procédures séquentielles s'effectue entre les procédures p_0 et p_{2^i-1} de la structure de voisinage de $SIM_\phi^{2^i}$, seules les procédures séquentielles p_0 et p_{2^i-1} changent l'un de leurs deux voisins lors du passage d'une procédure parallèle SIM^{2^i} à une procédure parallèle $SIM^{2^{i+1}}$.

Définition 7.11 *La structure de voisinage de la procédure séquentielle p_0 change de $\{p_{2^i-1}, p_0, p_1\}$ dans SIM^{2^i} à $\{p_{2^{i+1}-1}, p_0, p_1\}$ dans $SIM^{2^{i+1}}$ et celle de p_{2^i-1} change de $\{p_{2^i-2}, p_{2^i-1}, p_0\}$ dans SIM^{2^i} à $\{p_{2^i-2}, p_{2^i-1}, p_{2^i}\}$ dans $SIM^{2^{i+1}}$.*

Pour chaque procédure parallèle SIM_ϕ, nous illustrerons à l'aide d'un tableau le comportement dynamique typique de la règle de transition ϕ utilisée au niveau de la mémoire centrale. Ce tableau est une matrice $t^- \times 60$ qui illustre le statut des 60 premières entrées du vecteur E pour un nombre significatif t^- d'itérations de la règle ϕ. Le symbole '*' signifie que l'entrée correspondante du vecteur E est égale à 1 tandis qu'un blanc indique un 0. La première itération de chaque tableau correspond au vecteur initial E_0. L'objectif de cette matrice est de faire ressortir le comportement de chaque règle, indépendamment des conditions initiales de E_0. Nous ne ferons aucune simulation avec une règle en provenance de la classe 1 de Wolfram puisque l'évolution des règles de cette classe conduit rapidement vers un état uniforme de 1 ou 0 pour toutes les entrées de E.

La procédure SIM comme système dynamique

La valeur de τ et la règle de transition ϕ contrôlent l'interaction entre les procédures séquentielles du simulateur SIM. Dans la perspective des stratégies de parallélisation, on peut considérer que SIM est une stratégie de parallélisation basée sur une

décomposition fonctionnelle du traitement spéculatif. Le paramètre du couplage relatif à l'échange d'information appartient à la catégorie de création de nouvelle information (voir la section 5.3.3 du chapitre 5). La nouvelle information est générée à partir des solutions envoyées par les procédures séquentielles et de la fonction d'interaction définie par la règle de transition de l'automate cellulaire entre les procédures séquentielles d'un même voisinage.

Du point de vue de la mémoire centrale, SIM est un système dynamique non autonome où chaque procédure séquentielle est une variable du système. Le modèle du système dynamique de SIM est défini par $|K| = 2^n$, $r = 1$, $\mathcal{V} = \{p_{i-1}, p_i, p_{i+1}\}$. La loi dynamique est exprimée par $p_i^k = \phi[p_{i-i}^{k-1}, p_i^{k-1}, p_{i+1}^{k-1}]$. Ce système dynamique est converti en un automate cellulaire élémentaire de taille $p \times n$ (cette conversion préserve les propriétés du système dynamique) où les contextes de solutions envoyés par les procédures séquentielles à chaque point d'interaction forment l'état initial E_0 du système dynamique défini par la règle de transition ϕ. Étant donné une valeur du paramètre r, de τ et de l'exposant de Lyapunov λ de chaque règle ϕ, la trajectoire de E générée par l'action de ϕ va entrer dans un bassin d'attraction et ainsi converger vers un attracteur, ou elle va diverger continuellement dans des états transients. À la fin des τ itérations de l'automate cellulaire, si E se trouve dans un des états d'un attracteur, la mémoire centrale retourne aux procédures séquentielles des contextes appartenant à un état du système dynamique ϕ ayant un moindre degré d'entropie. Si les stratégies d'exploration s'actualisent à partir de ces contextes pour la phase de diversification, l'exploration de l'espace de solutions va redémarrer dans une région possédant une énergie minimum du point de vue du système dynamique ϕ. Nous étudierons donc le comportement d'interaction entre les stratégies d'exploration en fonction du type d'attracteur, de la vitesse de convergence vers les attracteurs et des similitudes existant avec les recherches multiples coopérantes.

La réduction du niveau d'entropie est une manifestation de l'auto-organisation des systèmes dynamiques. Cependant un niveau d'énergie minimal de ϕ n'est

d'aucune façon relié à un attracteur de l'espace de solutions, c'est-à-dire une région possédant des solutions intéressantes du point de vue du problème d'optimisation traité. Ce qui est probable cependant, c'est que l'interaction des procédures séquentielles dans les recherches multiples coopérantes constitue un système dynamique ayant ses propres attracteurs, capable donc de diriger l'évolution de la recherche des procédures tabous vers les régions définies par ces attracteurs. L'interaction entre les stratégies d'exploration définirait un autre paysage (landscape) qui influence l'évolution des procédures séquentielles tabous, en opposition avec l'hypothèse probabiliste selon laquelle ce paysage serait uniforme et sans impact sur la recherche tabou des recherches multiples coopérantes.

7.4 Étude du comportement en fonction du nombre de processeurs

Dans cette section nous montrerons pourquoi au niveau des recherches multiples coopérantes, une augmentation du nombre de procédures séquentielles peut se traduire par une diminution des performances de la procédure parallèle. Nous obtiendrons ce résultat à partir de l'analyse des dégradations des performances d'une procédure SIM et en utilisant les propriétés de cette procédure pour expliquer l'origine de ces dégradations. Mais tout d'abord, nous avons besoin d'obtenir un résultat préliminaire sur les recherches multiples. Dans la prochaine section, nous utiliserons donc SIM pour simuler le comportement des recherches multiples.

7.4.1 Simulation des procédures parallèles par recherches multiples

Les recherches multiples se caractérisent par l'absence d'échange explicite entre les procédures séquentielles de connaissances portant sur l'espace de solutions. Puisqu'il n'y a aucun échange d'information, il n'y a donc aucun comportement de coordination entre les procédures séquentielles basé sur les connaissances acquises. Nous

```
*       * *     *   * *   *   **   **  *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
*       * *     *   * *   *   *    *   *   * * * * *
```

Tableau 7.2 Comportement dynamique de la règle 12

pouvons reproduire cette loi simple d'absence d'interaction des recherches multiples à l'aide d'une procédure SIM. En terme de la loi dynamique des automates cellulaires élémentaires, l'absence d'interaction signifie que nous devons trouver une règle dont l'exposant de Lyapunov est nul, de tel sorte que les conditions initiales du vecteur E_0 se propagent à une très faible distance de chaque côté d'une entrée de ce vecteur. Les règles des classes 1 et 2 répondent à ce critère. Cependant, nous exclurons les règles de la classe 1 parce que l'évolution vers un état homogène de ces règles équivaut non seulement à l'absence de communication, mais entraîne également la destruction de toute information au niveau de chaque procédure séquentielle, ce qui n'est pas l'effet recherché pour cette simulation. Au niveau des règles de la classe 2, certaines se caractérisent par un transient initial d'une très courte durée (une seule itération) suivi d'un cycle où la même configuration se répète constamment. C'est ce genre de règles que nous pouvons utiliser pour la simulation du comportement des recherches multiples. Elles n'impliquent pratiquement aucun transport d'information entre les entrées du vecteur E_0, en particulier entre les entrées n'appartenant pas à la même procédure séquentielle. C'est le cas pour la règle 12 qui est illustrée par la figure 7.2.

Nous avons construit une procédure parallèle SIM à partir de la règle 12 de la classe 2 de Wolfram, simulation que nous identifierons par SIM_{12}. Comme le montre cette figure, la règle 12 se caractérise par un transient initial d'une très

courte durée (une seule itération). Par la suite, la règle 12 entre dans un cycle où la même configuration se répète constamment. Ce comportement dynamique de la règle 12 est typique des règles appartenant à la classe 2, et traduit la faible dépendance du statut des entrées du vecteur E sur les conditions initiales. Le statut d'une entrée k de E_j est déterminé par celui des entrées $\{k-\alpha, \ldots, k, \ldots, k+\alpha\}$ du vecteur E_0 où α a une valeur très faible en relation avec la phase d'états transients du système dynamique. Il y a donc peu de propagation de l'information à l'intérieur du vecteur E et conséquemment entre les procédures séquentielles appartenant à un même voisinage. La propagation de l'information étant très faible, le contexte C_j d'une procédure séquentielle p_j n'a presque pas d'impact sur le parcours du graphe de transitions des procédures séquentielles dans le voisinage de p_j. C'est ce qui explique le comportement au niveau des performances de SIM_{12} que l'on voit au tableau 7.3.

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{12}					
P1	0	21	0	21	0	10
P2	0.16 (1.66)	61	0.16 (1.66)	61	0.14 (0.71)	17
P3	0	156	0	156	0	156
P4	0.27 (0.79)	238	0.27 (0.79)	238	0 (0.13)	208
P5	0.38 (0.99)	15	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.92)	54	0.05 (1.76)	233	0 (3.18)	179
P7	0.07	25	0.07	25	0	162
P8	0.85	217	0	39	0	33
P9	0	168	0	168	0	112
P10	1.16 (2.76)	254	0.77	155	0.77	155
P11	0.61	72	0.61	72	0.61	72
P12	0.11	227	0 (0.30)	95	0	35
MOY	0.3050		0.1625		0.1283	

Tableau 7.3 Simulation des procédures parallèles par recherches multiples

On observe d'abord que les performances s'améliorent ou restent les mêmes en fonction du nombre de procédures séquentielles au niveau de la moyenne de

l'ensemble des problèmes et pour chaque problème individuel. Relativement aux problèmes 1, 2, 3, 4, 7, 9 et 11, on constate que les solutions sont identiques et ont été trouvées à la même itération pour $SIM_{12}^{2^2}$ et $SIM_{12}^{2^3}$. On peut faire la même observation pour les simulations $SIM_{12}^{2^3}$ et $SIM_{12}^{2^4}$ et les problèmes 3, 5, 10 et 11. Enfin, on voit que pour les problèmes 3 et 11, la solution est identique pour les trois catégories de simulation. L'absence de détérioration des performances est un comportement bien connu d'une procédure parallèle par recherches multiples.

Définition 7.12 *La meilleure solution trouvée par une procédure parallèle basée sur une décomposition fonctionnelle du traitement spéculatif correspond à la meilleure solution de la procédure séquentielle p_j lorsque cette solution est meilleure que toutes les solutions trouvées par les autres procédures séquentielles.*

Proposition 7.1 *Soit S^{i+1} la meilleure solution trouvée par la procédure parallèle $SIM_{12}^{2^{i+1}}$ et S^i la meilleure solution trouvée par la procédure parallèle $SIM_{12}^{2^i}$. Si $S^{i+1} = S^i$, alors la meilleure solution trouvée par la procédure parallèle $SIM_{12}^{2^{i+1}}$ et $SIM_{12}^{2^i}$ provient de la même procédure séquentielle p_j.*

Preuve: Cette proposition est vraie par construction de $\mathrm{SIM}^{2^{i+1}}$ à partir des procédures séquentielles de SIM^{2^i}.

La proposition 7.1 s'applique si le parcours de la procédure p_j n'est pas perturbé par le passage de $SIM2_{12}^i$ à $SIM2_{12}^{i+1}$ lors des itérations dans les états transients. La simulation SIM_{12} reproduit le comportement des procédures parallèles par recherches multiples si la construction de ces dernières procède selon la définition 7.10. Enfin, comme le montre la figure 7.2, les transitions de la mémoire centrale à chaque point d'interaction de la simulation SIM_{12} perturbent légèrement le vecteur de contexte envoyé à la mémoire centrale, ce qui fait que le contexte envoyé et celui reçu de la mémoire centrale ne sont pas les mêmes.

Donc, prises individuellement, les procédures séquentielles n'exécutent pas exactement le même parcours du graphe de transitions lors d'une simulation SIM_{12}

que celui exécuté dans une procédure parallèle par recherches multiples avec la même stratégie d'exploration. Cependant, un examen détaillé des simulations montre que du point de vue de l'impact sur les performances que peuvent avoir des changements au nombre de procédures séquentielles ou à la structure de voisinage, le comportement des simulations SIM_{12} est identique à celui des procédures parallèles par recherches multiples. Par conséquent, la proposition 7.1 s'applique, et c'est ce que l'on peut observé dans le tableau 7.3 où:

- La procédure séquentielle p_3 a trouvé la meilleure solution du problème P2 pour SIM^{2^2} et SIM^{2^3}
- La procédure séquentielle p_1 a trouvé la meilleure solution du problème P3, la procédure séquentielle p_3 a trouvé la meilleure solution du problème P11, pour les simulations SIM^{2^2}, SIM^{2^3} et SIM^{2^4};
- La procédure séquentielle p_4 a trouvé la meilleure solution des problèmes P5 et P10 pour les simulations SIM^{2^3} et SIM^{2^4};
- etc.

Nous avons exécuté cette simulation avec la règle 12 pour obtenir le résultat de la proposition 7.1 qui sera utile ultérieurement. Mais cette simulation montre également de quelle façon nous pouvons utiliser des résultats de la théorie sur les automates cellulaires pour simuler de manière précise certains comportements des procédures parallèles avec recherches multiples coopérantes et ainsi obtenir des résultats formels sur ce type de procédures parallèles.

7.4.2 Simulation d'une diminution de la qualité des solutions

Dans la présente section nous utiliserons une simulation avec SIM pour expliquer les occurences d'une augmentation du gap entre la solution optimale et la meilleure solution des recherches multiples coopérantes suite à une augmentation du nombre

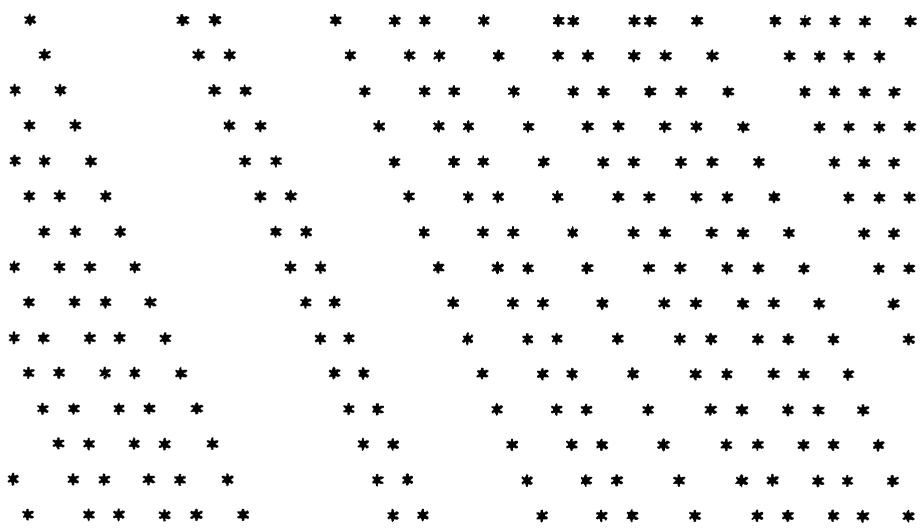


Tableau 7.4 Comportement dynamique de la règle 56

de procédures séquentielles. Pour ce faire nous avons besoin de règles au niveau de l'automate cellulaire qui simulent l'échange de connaissances sur l'espace de solutions entre les procédures séquentielles. Encore une fois il nous faut des règles avec un exposant de Lyapunov nul, dont la dynamique ne détruit pas l'information des messages en provenance des procédures séquentielles. Deuxièment, la dynamique de ces règles doit propager des configurations dans l'espace pour que l'échange de connaissances entre les procédures séquentielles soit possible. C'est le cas de certaines règles de la classe 2. La phase transient de ces règles est extrêmement courte, par la suite la dynamique de la règle évolue de manière cyclique dans un attracteur où le passage d'un état à l'état suivant correspond à un décalage des configurations dans le vecteur E (donc propagation des configurations dans l'espace). On retrouve ce type de comportement par exemple pour les règles 56 et 74 dont le comportement dynamique est illustré par les figures 7.4 et 7.5. Nous identifierons ces simulations respectivement par SIM_{56} et SIM_{74}.

Comme pour la règle 12, le transient de ces deux règles est très bref, suivie

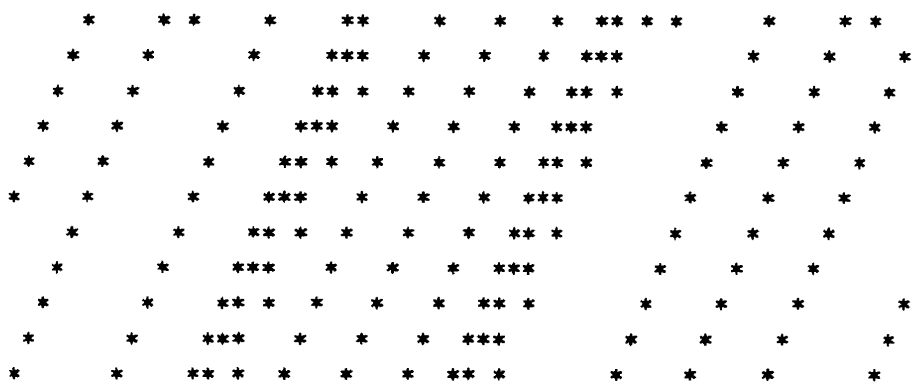


Tableau 7.5 Comportement dynamique de la règle 74

pour la règle 74 d'une période de deux itérations où les configurations se répètent ($E_k = E_{k-2}$) et d'une période d'une seule itération pour la règle 56. Les règle 56 et 74 possèdent cette particularité par rapport à la règle 12 qu'une fois entrée dans l'attracteur du système dynamique, bien que la configuration du vecteur E reste constante, la trajectoire de E se caractérise par un décalage à droite (règle 56) ou à gauche (règle 74) de l'information contenue à l'itération précédente (voir les figures 7.4 et 7.5). Au niveau des procédures parallèles SIM_{56} et SIM_{74}, ce décalage correspond à transférer le contexte de la meilleure solution à moyen terme d'une procédure p_j vers une des deux procédures dans le voisinage de p_j (lorsque $\tau = \lceil \frac{2n-1}{2r} \rceil$).

La démonstration montrant comment les recherches multiples coopérantes interfèrent avec la méthode pour déteriorer les performances de la procédure parallèle se fera à partir de la simulation SIM_{56}. Les mêmes résultats peuvent être obtenus avec la simulation SIM_{74} si on tient compte du sens du décalage.

Lemme 7.2 *Le nombre d'états de l'attracteur de SIM_{56} est borné supérieurement par $(p \times n) + \alpha$ où α est le nombre d'itérations dans la phase d'états transients.*

Preuve: Puisque après la phase d'états transients la configuration de E change uniquement par un processus de décalage des entrées, après $p \times n$ itérations, ce

processus de décalage va entrer dans une nouvelle période reproduisant uniquement les états de la période précédente.

Après l'exécution de $p \times n$ itérations dans la mémoire centrale, la trajectoire de E va revenir à la configuration initiale E_0. À la limite, une simulation SIM avec la règles 56 où le nombre d'itérations exécutés en mémoire centrale serait égal à $p \times n$ est équivalent à simuler une procédure parallèle par recherches multiples puisque chaque procédure séquentielle p_j reçoit le même contexte de la mémoire centrale que celui soumis à l'arrivée au point d'interaction. Dans la présente simulation, puisque nous limitons le nombre d'itérations à $\lceil \frac{2n-1}{2r} \rceil$, les procédures séquentielles vont recevoir des contextes différents de ceux correspondant à leur propre exploration de l'espace de solutions.

Lemme 7.3 *Le parcours du graphe de transitions exécuté par la procédure séquentielle p_j de la simulation SIM_{56} est déterministe et défini par la stratégie d'exploration s_j de la procédure p_j et par la séquence de $\frac{t}{\delta}$ contextes soumise à p_j par la mémoire centrale lors de l'exécution d'une simulation.*

Preuve: Le parcours de l'espace de solutions de la méthode tabou séquentielle est déterministe et complètement défini par la stratégie d'exploration. Dans le cadre de la procédure SIM_{56}, les points d'interaction entre les procédures séquentielles sont synchrones, conséquemment le parcours de chaque procédure p_j est déterministe. Pour chaque procédure séquentielle p_j, la seule modification effectuée à la stratégie d'exploration consiste à remplacer l'appel de diversification par un redémarrage de l'exploration de l'espace de solutions dans une région définie par le contexte reçu de la mémoire centrale. Conséquemment, le parcours de l'espace de solutions exécuté par p_j dépend uniquement de la stratégie d'exploration s_j et des contextes reçus de la mémoire centrale, plus particulièrement de la séquence des $\frac{t}{\delta}$ contextes reçus lors d'une exécution complète de SIM_{56}. Aucun autre facteur n'influence le comportement d'une procédure séquentielle p_j.

Lemme 7.4 *Soit $\Gamma_y = \{C_0, C_1, \ldots, C_{p-1}\}$ l'ensemble des contextes retournés par la mémoire centrale au point d'interaction y. L'ensemble Γ_y est invariant pour*

$$\tau = [\lceil \frac{2n-1}{2r} \rceil, 2(\lceil \frac{2n-1}{2r} \rceil), \ldots, p(\lceil \frac{2n-1}{2r} \rceil)]$$

pour toute simulation avec la procédure parallèle SIM_{56}.

Preuve: Puisque le nombre d'états de l'attracteur de la règle est borné supérieurement par $p \times n + \alpha$, la règle de transition de l'automate cellulaire élémentaire cycle dans le même attracteur quelque soit la valeur de τ, il n'y a donc pas de nouveaux états du système dynamique lorsque τ change. Si τ est un multiplicateur de $\lceil \frac{2n-1}{2r} \rceil$, chaque procédure séquentielle va recevoir un contexte correspondant à la meilleure solution à moyen terme d'une procédure p_k.

Cependant, lorsque τ change, l'ordre dans lequel chaque procédure séquentielle va recevoir les contextes en provenance de la mémoire centrale va différer. Par exemple, si $\tau = \lceil \frac{2n-1}{2r} \rceil$, chaque stratégie s_j va interagir avec le contexte de la procédure séquentielle p_{j-1}, si $\tau = 2(\lceil \frac{2n-1}{2r} \rceil)$, chaque stratégie s_j va interagir avec le contexte de la procédure séquentielle p_{j-2}, etc. Conséquemment, il y aura une exploration différente de l'espace de solutions par SIM_{56} en fonction de la valeur de τ.

Théorème 7.1 *Soit $[C_{j01}, \ldots, C_{j\frac{i}{\delta}}]$ la séquence de contextes reçue par une procédure séquentielle p_j en provenance de la mémoire centrale. Si la procédure séquentielle p_j est telle que $j < 2^i$, alors les contextes $C_{j1}, \ldots, C_{jj}$ reçus par p_j pour les procédures $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$ sont identiques et dans le même ordre.*

Preuve: Par construction (définition 7.10), la structure de voisinage des procédures séquentielles p_1 à p_{2^i-2} est la même pour les procédures parallèles $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$ et les stratégies d'exploration s_0 à s_{2^i-1} sont les mêmes pour SIM^{2^i} et $SIM^{2^{i+1}}$. Conséquemment, les contextes envoyés à la mémoire centrale par une procédure p_j pour les points d'interaction $1, \ldots, j+1$ seront inchangés. Lorsque $\tau = \lceil \frac{2n-1}{2r} \rceil$, p_j

reçoit de la mémoire centrale le contexte de la meilleure solution à moyen terme de p_{j-1}. Puisque p_{j-1} envoie à la mémoire centrale la même information pour $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$ aux points d'interaction 1 à $j+1$, les contextes C_{j1} à C_{jj} retournés à p_j par la mémoire centrale seront identiques pour les deux procédures parallèles $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$.

Corollaire 7.1 *Soit p_j une procédure séquentielle telle que $j < 2^i$. Le nombre d'itérations où le parcours du graphe de transitions accompli par $p_j \in SIM_{56}^{2^i}$ et celui accompli par $p_j \in SIM_{56}^{2^{i+1}}$ sont identiques, est borné inférieurement par $(j+1) \times \delta$.*

Preuve: Puisque la stratégie d'exploration est la même pour $p_j \in SIM_{56}^{2^i}$ et $p_j \in SIM_{56}^{2^{i+1}}$, le parcours de l'espace de solutions pour les procédures séquentielles p_0 à p_{2^i-1} est identique pour les itérations 1 à δ de $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$. Les contextes reçus en provenance de la mémoire centrale sont les mêmes pour les points d'interaction 1 à j, ce qui implique que le comportement de la procédure séquentielle change au plus tôt après la réception du premier contexte différent entre $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$, c'est-à-dire au point d'interaction $j+1$ ($p_0 = j+1 = 0+1 = 1, p_1 = j+1 = 1+1 = 2$, etc). Il faut alors au moins 4 points d'interaction par exemple avant que la procédure séquentielle p_3 commence à recevoir des contextes différents au niveau de l'exécution de la procédure parallèle $SIM_{56}^{2^2}$ et celle de $SIM_{56}^{2^3}$. Il faut au moins 8 points d'interaction avant que la procédure séquentielle p_7 commence à recevoir des contextes différents entre $SIM_{56}^{2^3}$ et $SIM_{56}^{2^4}$.

Corollaire 7.2 *Soit p_j une procédure séquentielle telle que $j \geq 2^i$. Le nombre d'itérations où le parcours du graphe de transitions accompli par $p_j \in SIM_{56}^{2^{i+1}}$ est indépendant du parcours des procédures séquentielles p_k où $k < 2^i$ est borné inférieurement par $(j - 2^i + 1) \times \delta$.*

Preuve: Le parcours d'une procédure séquentielle p_j où $j \geq 2^i$ est d'abord déterminé par la stratégie d'exploration s_j avant le premier point d'interaction, ensuite par la séquence de contextes reçue à chaque point d'interaction. Cette séquence suit

le même ordre que celui créé par la structure de voisinage entre les procédures séquentielles de $SIM_{56}^{2^{i+1}}$. La procédure séquentielle p_j va donc d'abord recevoir des contextes de solutions en provenance des procédures séquentielles $p_k \in \{SIM_{56}^{2^{i+1}} \setminus SIM_{56}^{2^i}\}$ avant de recevoir ceux en provenance des procédures séquentielles p_k pour $k < 2^i$.

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{56}					
P1	0	21	0	21	0	10
P2	0.05 (1.77)	208	0.22 (1.72)	193	0.08 (1.55)	153
P3	0	182	0	284	0	220
P4	0.21 (2.55)	138	0.24 (1.61)	51	0.08 (0.74)	42
P5	0.04 (3.01)	141	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.28)	194	0.01 (0.68)	267	0 (0.08)	236
P7	0.07	25	0	247	0	99
P8	1.32	64	0	112	0	33
P9	0	75	0	64	0	69
P10	1.53 (2.30)	133	1.09	175	1.16	93
P11	0.61 (3.04)	160	0	153	0.61	218
P12	0.11	92	0 (1.03)	138	0	35
MOY	0.3325		0.1317		0.1625	
	SIM_{74}					
P1	0	21	0	21	0	10
P2	0.30 (2.00)	266	0.26 (1.90)	269	0.14 (0.71)	17
P3	0.01 (0.53)	89	0.01 (0.12)	54	0	162
P4	0.51 (1.47)	37	0.12 (1.61)	18	0.08 (0.74)	47
P5	0 (0.45)	185	0.02 (0.27)	9	0 (1.25)	244
P6	0.05 (1.92)	54	0.05 (0.62)	187	0.05 (0.34)	45
P7	0.07	25	0.07	25	0.07	25
P8	0.85	275	0	41	0	41
P9	0	73	0	73	0	143
P10	1.68 (4.41)	227	1.16	97	1.16	97
P11	0.61	139	0.61	129	0.61	139
P12	0	143	0	197	0	35
MOY	0.3400		0.1917		0.1758	

Tableau 7.6 Simulations avec des règles de la classe 2

Le tableau 7.6 montre les résultats obtenus pour les simulations SIM_{56} et

SIM_{74}. Deux cas se présentent lorsque les résultats sont identiques entre les procédures SIM^{2^i} et $SIM^{2^{i+1}}$ dans ce tableau. La meilleure solution de certains problèmes (P1 et P5 pour la règle 56, et P1 et P7 pour la règle 74) est obtenue avant toute interaction entre les procédures séquentielles , alors la proposition 7.1 relativement aux procédures séquentielles indépendantes s'applique. Par contre, tous les autres résultats identiques dérivent du corrollaire 7.1. Les résultats identiques de P8, P9 et P10 entre les procédures $SIM_{74}^{2^i}$ et $SIM_{74}^{2^{i+1}}$ sont attribuables au fait que la meilleure solution de la procédure parallèle $SIM_{74}^{2^i}$ a été obtenue par une procédure séquentielle pour une itération inférieure à $(j+1) \times \delta$ de la méthode tabou, donc à une itération où le parcours du graphe de transitions est identique pour les deux procédures parallèles.

Théorème 7.2 *Soit S^i la meilleure solution du problème P obtenue à l'itération y de $p_j \in SIM^{2^i}$ tel que $y < ((j+1) \times \delta)$ et S^{i+1} la meilleure solution du problème P obtenue par une procédure parallèle $SIM_{56}^{2^{i+1}}$. Si $S^i \neq S^{i+1}$ alors $S^{i+1} < S^i$.*

Preuve: Ce théorème découle de la définition 7.12 et du corrollaire 7.1. Puisque le parcours accompli par une procédure séquentielle p_j est identique pour $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$ pour toutes les itérations inférieures à $(j+1) \times \delta$, toute meilleure solution trouvée par $p_j \in SIM_{56}^{2^i}$ à l'itération $y < (j+1) \times \delta$ sera également trouvée par $SIM_{56}^{2^{i+1}}$. Supposons que $S^i \neq S^{i+1}$ et que $S^{i+1} < S^i$. Si $S^i \neq S^{i+1}$, c'est parce que la procédure parallèle $SIM_{56}^{2^{i+1}}$ a trouvé une solution meilleure que celle trouvée par la procédure séquentielle p_j de $SIM_{56}^{2^i}$. Dans ce cas, les circonstances de l'obtention de la solution S^{i+1} appartiennent à l'une des 3 catégories suivantes:

1. S^{i+1} a été obtenue par une procédure séquentielle $p_j \in SIM_{56}^{2^{i+1}}$ où $j \geq 2^i$, suite à l'utilisation de la nouvelle stratégie d'exploration s_j et sans interaction avec un contexte en provenance d'une procédure p_k, $k \neq j$;

2. S^{i+1} a été obtenue par une procédure séquentielle $p_j \in SIM_{56}^{2^{i+1}} \setminus SIM_{56}^{2^i}$ suite à l'interaction de s_j avec des contextes en provenance de procédures séquentielles

p_k, $k \neq j$;

3. S^{i+1} provient de $p_j \in SIM_{56}^{2^i}$ suite à une interaction de la stratégie d'exploration s_j avec au moins un contexte en provenance de procédures séquentielles $p_k \in SIM_{56}^{2^{i+1}} \setminus SIM_{56}^{2^i}$.

Mais dans les trois cas, pour que $S^{i+1} > S^i$, il faudrait que le parcours du graphe de transitions de la procédure p_j ait été détruit avant que la solution S^i ne soit trouvée. Or le parcours de $p_j \in SIM_{56}^{2^i}$ et $p_j \in SIM_{56}^{2^{i+1}}$ est identique à l'itération y de p_j. Donc si $S^i \neq S^{i+1}$, on a nécessairement $S^{i+1} < S^i$.

Le théorème 7.2 explique un certain nombre de résultats du tableau 7.6. La meilleure solution du problème P8 de $SIM_{56}^{2^2}$ a été trouvée par p_3 à l'itération 64 et celle de P8 pour $SIM_{56}^{2^3}$ a été trouvée par p_2 à l'itération 112, soit après avoir interagit avec les contextes des procédures séquentielles p_7 et p_6 aux points d'interaction à l'itération 75 et 100. Les problèmes P9 et P12 de $SIM_{56}^{2^2}$ et P4 de $SIM_{56}^{2^3}$ se trouvent dans la même situation.

Théorème 7.3 *Soit S^i la meilleure solution du problème P obtenue à l'itération y de $p_j \in SIM_{56}^{2^i}$ et S^{i+1} la meilleure solution du problème P obtenue par une procédure parallèle $SIM_{56}^{2^{i+1}}$. $S^{i+1} > S^i$ ssi $y > ((j+1) \times \delta)$ et un changement dans la séquence de contextes envoyés par la mémoire centrale à la procédure p_j suite au passage de $SIM_{56}^{2^i}$ à $SIM_{56}^{2^{i+1}}$ a provoqué une modification du parcours du graphe de transitions de la procédure séquentielle p_j.*

Preuve: Pour que $S^{i+1} > S^i$, il faut que la solution S^i ne puisse pas être trouvée par la procédure parallèle $SIM_{56}^{2^{i+1}}$. Pour que cela soit possible, il faut que de manière minimale le parcours de p_j soit modifié. Par le corrollaire 7.1, on sait que si $y < ((j+1) \times \delta)$ alors les parcours de $p_j \in SIM_{56}^{2^i}$ et de $p_j \in SIM_{56}^{2^{i+1}}$ sont identiques et donc $S^{i+1} \leq S^i$. Nous savons par le lemme 7.2 que le parcours des procédures séquentielles est strictement déterminé par la stratégie d'exploration et la séquence de contextes en provenance de la mémoire centrale. Par construction

de $SIM_{56}^{2^{i+1}}$ on sait que la stratégie d'exploration s_j est présente dans $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$. Pour que le parcours de p_j soit modifié, il faut donc que la séquence de contextes en provenance de la mémoire centrale soit différente. Or si τ est constant pour $SIM_{56}^{2^i}$ et $SIM_{56}^{2^{i+1}}$, le seul facteur pouvant provoquer un changement dans la séquence de contextes reçus par une procédure p_j est la modification du nombre de procédures séquentielles suite au passage de $SIM_{56}^{2^i}$ à $SIM_{56}^{2^{i+1}}$.

Le théorème 7.3 montre que l'ajout des procédures séquentielles $p_{2^i}, p_{2^i+1}, \ldots, p_{2^{i+1}-1}$ dans $SIM_{56}^{2^{i+1}}$ modifie les parcours des procédures séquentielles de $p_j \in SIM_{56}^{2^i}$ pour les itérations au-delà de $(j+1) \times \delta$. Conséquemment, les solutions trouvées dans les parcours éliminés disparaissent également, y compris la meilleure solution de $p_j \in SIM_{56}^{2^i}$ si elle se trouvait sur l'un de ces parcours éliminés. Le théorème 7.3 explique donc pourquoi l'accroissement du nombre de procédures séquentielles peut parfois résulter en une diminution des performances d'une procédure parallèle basée sur une décomposition fonctionnelle du traitement spéculatif.

Le théorème 7.3 explique le comportement des procédures parallèles $SIM_{56}^{2^2}$ et $SIM_{56}^{2^3}$ pour les problèmes P2, P4, $SIM_{74}^{2^2}$ et $SIM_{74}^{2^3}$ pour le problème P5, et $SIM_{56}^{2^3}$ et $SIM_{56}^{2^4}$ pour les problèmes P10 et P11. Pour les problème P2, P4 et P5, les meilleures solutions ont été trouvées respectivement par les procédures séquentielles p_0, p_3 et p_2 à des itérations bien au-delà de $(j+1) \times \delta$. Dans tous les cas le parcours du graphe de transitions conduisant à la solution S^i a été détruit lors du passage à la procédure parallèle SIM^{2^3}. On note le même phénomène pour les problèmes P10 et P11 au niveau de la procédure parallèle SIM^{2^3}.

Proposition 7.2 *Soit T_i et T_j deux procédures parallèles avec recherches multiples coopérantes de la méthode tabou. La démonstration des causes de détérioration des performances de la procédure SIM_{56} se généralise à toute procédure parallèle T_i et T_j à la seule condition que les stratégies d'exploration d'une des deux procédures parallèles soit un sous-ensemble des stratégies d'exploration de l'autre procédure*

parallèle.

Preuve: L'augmentation du nombre de procédures séquentielles dans $SIM_{56}^{2^i}$ change la structure de voisinage des procédures séquentielles p_0 et p_{2^i-1}. Le changement dans la structure de voisinage de p_0 en particulier, fait que la stratégie d'exploration s_0 de la procédure p_0 ne s'actualise plus à partir de la même information dans $SIM_{56}^{2^{i+1}}$ dès le premier point d'interaction. Il en résulte que les connaissances acquises par p_0 changent dans $SIM_{56}^{2^{i+1}}$, ce qui, à travers le comportement dynamique de réactions en chaîne décrit au chapitre précédent, finit par affecter l'ensemble global des connaissances acquises. Puisque l'ensemble global des connaissances acquises auxquelles ont accès les stratégies d'exploration des procédures séquentielles p_0 à p_{2^i} de $SIM_{56}^{2^{i+1}}$ n'est plus le même, le parcours de ces procédures séquentielles s'en trouve modifier. Or, cette séquence d'évènements, 1- changement de la structure de voisinage, 2- changement de l'actualisation de certaines stratégies d'exploration, 3- changement de l'ensemble global des connaissances acquises et finalement 4- changement dans le parcours de l'espace de solutions (détruisant le chemin de la meilleure solution) est la même pour SIM_{56} comme pour toutes les autres procédures parallèles avec recherches multiples coopérantes qui respectent la condition de la proposition 7.2. Conséquemment, les conclusions portant sur la procédure SIM_{56} se généralisent aux autres procédures parallèles avec recherches multiples coopérantes quelque soit le mode de synchronisation, le nombre de procédures séquentielles ou encore la structure de voisinage.

Discussion

À la section 6.3.3 nous avons rapporté les observations de Verhoeven selon lesquelles les performances d'une méthode de recherche séquentielle dépendent, entre autre, de la façon dont le graphe de transitions est parcouru qui à son tour dépend de la stratégie d'exploration. Une des principale conclusion de l'analyse théorique de cette section est que le parcours du graphe de transitions n'est pas seulement

tributaire d'une seule stratégie d'exploration, mais plutôt d'un ensemble de stratégies d'exploration et de la manière dont ces stratégies d'exploration interagissent entre elles. D'après la définition 7.12, la meilleure solution d'une procédure parallèle basée sur une décomposition fonctionnelle du traitement spéculatif correspond à la meilleure solution de la procédure séquentielle p_j, si aucune autre procédure séquentielle n'a une meilleure solution que p_j. Mais la meilleure solution est tributaire du parcours du graphe de transitions qui a permis de trouver cette solution. Or, le parcours du graphe de transitions d'une procédure séquentielle est la résultante d'une interaction complexe entre les stratégies d'exploration. Il ne faut donc pas que la définition 7.12 nous porte à croire que la meilleure solution a été trouvée par p_j puisqu'en fait, la meilleure solution de toute procédure séquentielle des recherches multiples coopérantes dépend des solutions trouvées par plusieurs autres procédures.

Dès que l'interaction complexe entre les stratégies d'exploration est modifiée, par l'ajout par exemple d'un nouvel ensemble de procédures séquentielles, il est possible que le parcours du graphe de transitions ayant permit de trouver la meilleure solution ne soit pas exécuté. Si la nouvelle interaction entre les stratégies d'exploration résultant de l'ajout du nouvel ensemble de procédures séquentielles ne réussit pas à produire un parcours qui permette de trouver une solution aussi bonne que la précédente, alors il y aura déterioration de la qualité de la meilleure solution.

Un changement dans la dynamique d'interaction entre les stratégies d'exploration d'une procédure avec recherches multiples coopérantes s'apparente à celui d'un changement dans la stratégie d'exploration d'une procédure de recherche tabou séquentielle. Dans ce dernier cas, on ne peut pas prévoir à l'avance si les changements dans la stratégie d'exploration vont résulter en une amélioration ou une déterioration des performances de la recherche tabou. Nous allons maintenant voir que l'interaction entre les stratégies d'exploration semble reposer sur un certain niveau d'organisation, ce qui laisse un peu d'espoir de pouvoir contrôler

ce phénomène de détérioration de la qualité de la meilleure solution.

7.5 Auto-organisation des recherches multiples coopérantes

Nous introduirons maintenant une simulation dont l'objectif est de montrer que l'échange de messages des procédures par recherches multiples coopérantes n'obéit pas à la logique d'optimisation de la méthode de recherche, mais suit plutôt des lois utilisées pour décrire l'évolution dans le temps de systèmes dynamiques. Nous montrerons qu'à long terme, des dynamiques comme l'interaction en chaîne et l'interaction récursive échappent à la logique d'optimisation et se comportent comme un système dynamique. Conséquemment, les facteurs d'interférence sur la méthode de recherche provoquent une dérive réelle de l'exploration de l'espace de solutions, dérive que nous modéliserons. Nous nous intéresserons plus particulièrement à un modèle qualitatif des systèmes dynamiques, celui du comportement d'auto-organisation des systèmes dynamiques. Comme mentionné dans l'introduction de ce chapitre, le comportement d'auto-organisation est assez bien connu dans certains milieux scientifiques et du génie. Il y a également plusieurs chercheurs qui pensent que cette dynamique s'applique aux systèmes de calcul comme celui des recherches multiples coopérantes [40].

Supposons que les procédures séquentielles d'une procédure parallèle par recherches multiples s'échangent des messages dont le contenu correspond à de bonnes solutions trouvées par ces procédures séquentielles. Selon la thèse actuelle, sous-jacente aux modèles du calcul exécuté par les recherches multiples coopérantes, l'interaction entre les procédures séquentielles accentue ou n'affecte pas la découverte de bonnes solutions. Selon cette interprétation, on pourrait verbaliser de la manière suivante pour les procédures séquentielles un échange de messages: "suite à nos échanges d'information, je t'envoie maintenant une solution encore meilleure que

celle que je t'aurais envoyée s'il n'y avait pas eu ces échanges de messages", ou encore "cette fois je t'envoie une meilleure solution mais la dernière fois ma solution était moins bonne que celle que je t'aurais envoyée sans échange de message, mais ne t'en fais pas en moyenne mes solutions ne seront ni pires ni meilleures suite à ces interactions". La thèse que nous allons défendre maintenant se traduirait par le type suivant de messages: "suite à nos interactions passées, je t'envoie maintenant une solution qui pourra t'aider à définir ton chemin d'exploration en fonction du mien et ainsi améliorer l'organisation de nos deux recherches". La thèse sous-jacente à ce dernier message est fort différente de la première, selon cette dernière thèse les chemins d'exploration des procédures séquentielles progressent en fonction des chemins des autres procédures séquentielles et non pas en fonction de la structure de l'espace de solutions telle que définie selon la logique d'un problème d'optimisation.

Pour soutenir notre thèse, nous définirons une procédure parallèle avec recherches multiples coopérantes où le contenu des messages échangés vise à créer de l'organisation dans la procédure parallèle et non pas à échanger des solutions sur l'espace de solutions. Pour ce faire, nous allons recourir à des règles de l'automate cellulaire élémentaire qui peuvent transformer les messages provenant de plusieurs procédures séquentielles. Ces messages correspondent à des solutions du problème d'optimisation, ils seront transformés dans la mémoire centrale en de nouveaux messages traduisant une coordination complexe des procédures séquentielles entre elles. Il nous faut donc des règles dont l'évolution est sensible aux conditions initiales du vecteur E_0, c'est-à-dire des règles ayant un exposant de Lyapunov positif. Les règles des classes 3 et 4 se qualifient selon ce critère. Puisque que nous cherchons à tester l'hypothèse d'un comportement d'auto-organisation des recherche multiples coopérantes, nous avons besoin de règles dont le comportement d'auto-organisation est bien connu, c'est le cas, d'après Wolfram, pour les règles de la classe 3. Nous avons choisi les règles 18 et 22 de la classe 3, la figure 7.7 illustre le comportement dynamique de la règle 18. Nous identifierons par SIM_{18} et SIM_{22} les deux

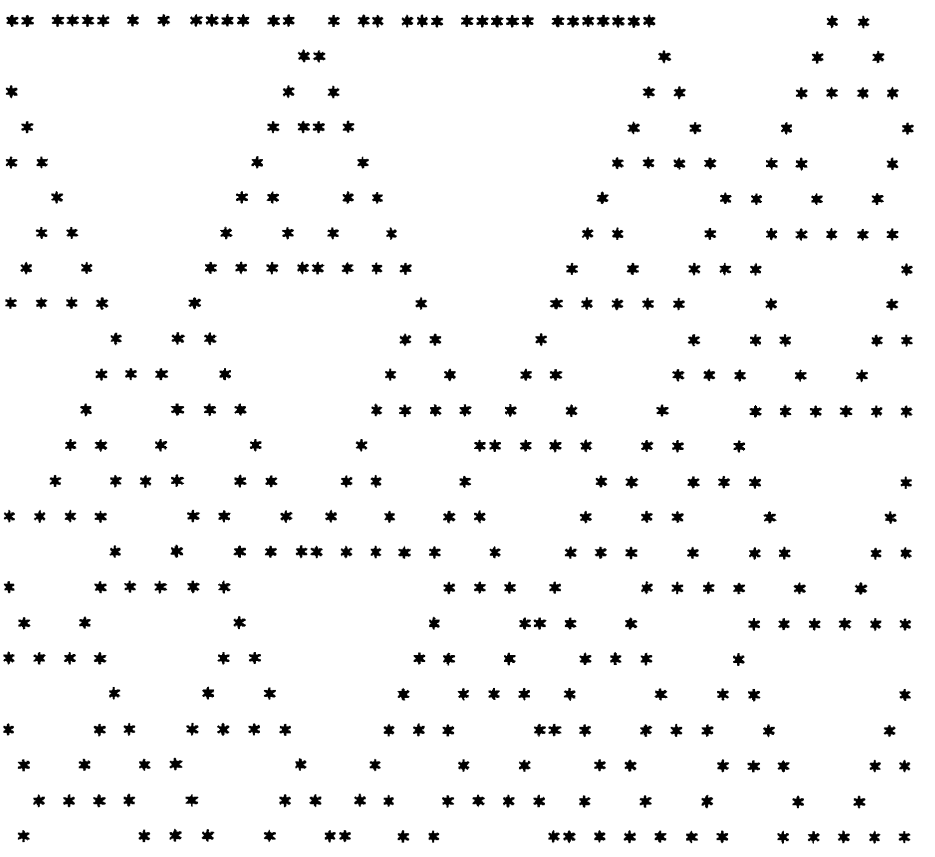


Tableau 7.7 Règle 18 au point d'interaction 1 de SIM_{18}

procédures parallèles où l'interaction entre les procédures séquentielles est basée sur ces règles.

7.5.1 Évolution dynamique des règles 18 et 22

Comme le montre la figure 7.7, la dynamique des règles appartenant à la classe 3 est beaucoup plus complexe que celle des règles que nous avons utilisées jusqu'à maintenant, il peut même sembler que les contextes retournés par la mémoire centrale sont générés de manière aléatoire. Nous allons maintenant voir que ce n'est pas le cas.

On remarque d'abord, que l'évolution de la règle 18 ne se fixe pas dans une

configuration stable ou périodique. On remarque également certaines structures espace-temps ayant la forme de triangles. Ces structures révèlent le fait que le statut de chaque entrée de E dépend du statut d'un nombre toujours plus grand d'entrées du vecteur E_0 en fonction du nombre d'itérations de la règle ϕ. Les définitions suivantes décrivent de manière précise la relation existant entre le contexte que la mémoire centrale retourne aux procédures séquentielles et ceux qu'elle reçoit en provenance de ces procédures, en particulier ceux dans le voisinage de chaque procédure séquentielle p_j.

Définition 7.13 *Soit* $C_j = \{c_0, c_1, \ldots, c_n\}$ *le contexte retourné par la mémoire centrale à la procédure séquentielle* p_j *et* τ *le nombre d'itérations exécutées par la règle* ϕ *de la mémoire centrale. De manière générale, chaque entrée* $c_k \in C_j$ *dépend des entrées*

$$\{((j \times n) - \tau) + k, \ldots, (j \times n) + k, \ldots, ((j \times n) + \tau) + k\}$$

du vecteur d'état inital E_0.

Définition 7.14 *Si* $\tau = \lceil \frac{2n-1}{2r} \rceil$, *la valeur de* $c_0 \in C_j$ *est déterminée par les entrées*

$$\{(j \times n) - \tau, \ldots, j \times n, \ldots, (j \times n) + \tau\}$$

du vecteur E_0, *c'est-à-dire le contenu du contexte envoyé à la mémoire centrale par les procédures séquentielles* p_{j-1} *et* p_j.

Définition 7.15 *Si* $\tau = \lceil \frac{2n-1}{2r} \rceil$, *la valeur de* $c_n \in C_j$ *est déterminée par les entrées*

$$\{((j+1) \times n) - \tau, \ldots, (j+1) \times n, \ldots, ((j+1) \times n) + \tau\}$$

du vecteur E_0, *c'est-à-dire le contenu du contexte envoyé à la mémoire centrale par les procédures* p_j *et* p_{j+1}.

En d'autres termes, chaque entrée c_i du contexte reçu par une procédure séquentielle est défini par la loi dynamique de la classe 3 en se basant sur le

statut de 44 entrées de chaque côté de c_i dans E_0. Une fois cette transformation effectuée, le contenu du message reçu par une procédure est sensé traduire un état d'organisation globale de la recherche régissant l'interaction entre les procédures séquentielles. À chaque fois qu'une procédure utilise un contexte provenant d'un message, elle redémarre sa recherche non pas une région intéressante selon la logique d'optimisation, mais dans une région intéressante selon un plan d'organisation d'ensemble de la recherche, plan qui ne vise pas nécessairement à trouver de bonnes solutions au problème d'optimisation. Dans le cas des règles 18 et 22, ce plan vise à réduire l'entropie de l'exploration de l'espace de solutions en créant une pression sur les procédures séquentielles pour qu'elles explorent l'espace de solutions selon des structures de contextes similaires à celles de la figure 7.7

Loi d'entropie des règles 18 et 22

La dépendance d'une entrée donnée du vecteur E sur un nombre toujours croissant d'entrées de E_0 en fonction du nombre d'itérations de ϕ traduit l'existence d'un processus complexe de structuration des contextes générés par la mémoire centrale. Nous savons que la loi dynamique de l'automate cellulaire élémentaire est donnée par la fonction:

$$x_i^k = \phi[x_{i-r}^{k-1}, x_{i-r+1}^{k-1}, \ldots, x_i^{k-1}, \ldots, x_{i+r}^{k-1}].$$

De cette règle ϕ de transition locale on peut déduire une règle de transition globale entre les ensembles de configuration:

$$\Phi : \Sigma \rightarrow \Sigma$$

où Σ est l'ensemble des configurations possibles de l'automate cellulaire élémentaire (dans le cas de l'automate cellulaire élémentaire de la mémoire centrale $|\Sigma| = 2^{p \times n}$). L'application $\Phi : \Sigma \rightarrow \Sigma$ implique que la règle de transition ϕ est appliquée à chaque configuration se trouvant dans Σ. Soit $\Omega^\tau = \Phi^\tau \Sigma$ l'ensemble des configurations

générées après τ itérations de Φ. On sait qu'en général [106]:

$$\Omega^{\tau+1} = \Phi\Omega^{\tau} \subseteq \Omega^{\tau}$$

c'est-à-dire l'application de la règle Φ à toutes les configurations de l'ensemble Ω^{τ} à l'itération τ résulte en un ensemble $\Omega^{\tau+1}$ à l'iteration $\tau + 1$ dont le nombre de configurations est inférieur ou égal à celui de l'ensemble Ω^{τ}. L'ensemble des configurations générées par certaines règles de transition de la classe 3 se contracte ou reste inchangé, c'est le cas pour les règles 18 et 22 que nous utiliserons pour les simulations. Cette évolution à la baisse du nombre de configurations dans les ensembles Ω signifie que l'entropie associée aux ensembles Ω^{τ} décroît ou reste la même avec l'augmentation du nombre d'itérations. La diminution de l'entropie d'un système dynamique est une manifestation de l'auto-organisation du système.

7.5.2 Comportement d'auto-organisation des procédures SIM_{18} et SIM_{22}

La diminution du nombre de configurations au niveau des ensembles Ω est observable par le fait que certains blocs de longueur l deviennent exclus des configurations générées avec le nombre croissant d'itérations de ϕ.

Définition 7.16 *Un bloc de longueur l est exclu après τ itérations de la règle ϕ s'il n'y a pas un bloc dans le vecteur initial E_0 de longueur $(l + 2r) \times \tau$ qui évolue vers le bloc de longueur l.*

Définition 7.17 *Un bloc de longueur l est nouvellement exclu à l'itération τ ssi aucun bloc de longueur $l + 2r$ existant à l'itération $\tau - 1$ n'évolue vers ce bloc, mais au moins un bloc de longueur $l + 2r$ nouvellement exclu à l'itération $\tau - 1$ aurait évolué vers ce bloc (voir Wolfram [106]).*

Proposition 7.3 *L'évolution dans le temps de la règle 18 correspond à celle d'une dynamique d'auto-organisation.*

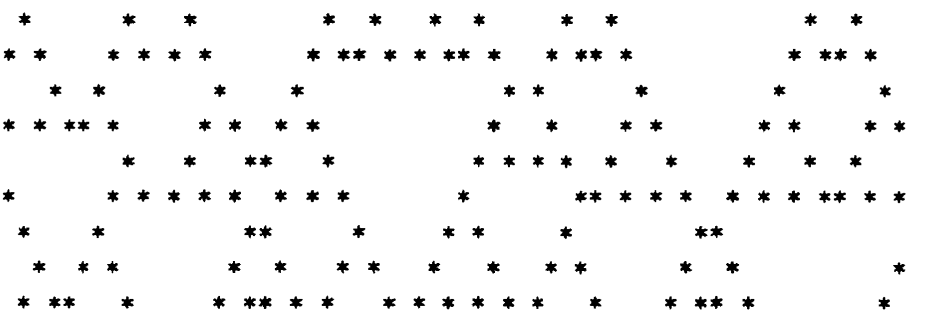


Tableau 7.8 Règle 18 au point d'iteraction 2 de SIM_{18}

Preuve: Selon les définitions 7.16 et 7.17, la diminution de l'entropie traduisant une dynamique d'auto-organisation se manifeste par l'exclusion de blocs de configurations pendant l'évolution dans le temps de la loi dynamique. Comme le montre la figure 7.7, le bloc [111] de longueur 3 est exclu dès le départ, c'est-à-dire pour le vecteur E_1. De même le bloc [1101011] de longueur 7, les blocs [110001011] et [1101001] de longueur 8, etc. La dynamique de la règle 18 manifeste donc un comportement d'auto-organisation.

Nous pourrions obtenir le même résultat pour la règle 22, nous ferons l'hypothèse que la dynamique de la règle 22 manifeste également un comportement d'auto-organisation.

Le comportement d'auto-organisation des règles de transitions de la mémoire centrale a un impact direct sur les procédures séquentielles de la simulation SIM_{18} comme on peut le constater en comparant les figures 7.7 et 7.8. Ces deux figures montrent la dynamique de la règle 18 aux points d'interaction 1 et 2 d'une procédure parallèle SIM_{18}. On peut observer la grande différence du vecteur E_0 entre les points d'interaction 1 et 2. On voit par exemple que les blocs [1111], [11111], [1111111] n'apparaisent plus dans l'état initial E_0 lors du point d'interaction 2. Cette disparition de certains blocs montre que l'évolution de la règle appliquée au niveau de la mémoire centrale a un succès évident à imposer une structure sur la recherche effectuée par les procédures séquentielles.

Hypothèse 7.1 *La dynamique d'auto-organisation des règles 18 et 22 interfère avec la méthode tabou des procédures séquentielles des recherches multiples coopérantes SIM_{18} et SIM_{22} pour organiser l'exploration à long terme de l'espace de solutions selon un plan qui ne correspond pas nécessairement à la logique d'optimisation de la méthode tabou.*

Nous allons maintenant montrer que ce comportement d'auto-organisation des procédures SIM_{18} et SIM_{22} se retrouve également au niveau des procédures avec recherches multiples coopérantes basées sur un échange de connaissances acquises entre les procédures séquentielles. Pour ce faire, il nous d'abord définir une mesure du niveau d'organisation des procédures avec recherches multiples coopérantes qui puisse nous permettre de comparer les procédures parallèles entre elles.

7.5.3 Mesure du niveau d'organisation de la recherche parallèle

Nous utiliserons la *distance globale de Hamming* comme base de comparaison des comportements des différentes procédures parallèles.

Définition 7.18 *La distance de Hamming $H(v_i, v_j)$ entre deux vecteurs binaires v_i et v_j de même taille correspond au nombre d'entrées qui n'ont pas le même statut dans les deux vecteurs.*

Par exemple la distance de Hamming $H(v_1, v_2)$ entre $v_1 = [0, 1, 1, 1, 0]$ et $v_2 = [1, 0, 1, 1, 0]$, est de 2. La distance globale de Hamming d'une procédure de traitement parallèle correspond à trouver la somme des distances de Hamming entre tous les contextes de solutions trouvés par la procédure parallèle.

Définition 7.19 *Soit C_{kj} le contexte de la solution trouvée à l'itération k par la procédure séquentielle p_j. La distance de Hamming $H^-(p_j(k), p_j)$ entre le contexte de la solution à l'itération k de la procédure séquentielle p_j et celui de toutes les autres solutions visitées par p_j est donnée par $H^-(p_j(k), p_j) = \sum_{l=1}^{t} H(C_{kj}, C_{lj})$.*

Définition 7.20 *La distance de Hamming $H^*(p_j(k), p)$ entre le contexte de la solution trouvée à l'itération k de la procédure séquentielle p_j et le contexte de toutes les solutions trouvées par une procédure parallèle est donnée par:*

$$H^*(p_j(k), p) = \sum_{l=0}^{l=p-1} H^-(p_j(k), p_l)$$

Définition 7.21 *La distance globale de Hamming d'une procédure parallèle est donnée par $H^G = \sum_{l=1}^{t \times p} H^*(l, p)$, c'est-à-dire la somme des distances de Hamming entre chaque solution d'une procédure parallèle et toutes les autres solutions de cette même procédure parallèle.*

La distance globale de Hamming est une mesure insensible à l'espace et au temps, elle nous permet de mesurer le niveau d'organisation globale de l'exploration de l'espace de solutions tout en faisant abstraction de l'itération et de la procédure séquentielle où les solutions sont visitées. Par contre, la distance globale de Hamming est sensible à tout phénomène de structuration de cette exploration, phénomène qui se manifeste ici par l'exclusion de blocs dans les configurations, une manifestation de l'auto-organisation selon Wolfram. Lorsqu'un plus grand nombre de blocs sont exclus, les configurations de contextes tendent à être plus similaires et donc la distance de Hamming entre ces configurations est moindre.

Définition 7.22 *Nous dirons que le niveau d'organisation de deux procédures parallèles tend à être le même si la distance globale de Hamming de ces deux procédures tend vers la même valeur.*

Les quatre composantes de la distance globale de Hamming

La distance de Hamming entre les contextes de deux solutions différentes indique le nombre de variables de décisions qui n'ont pas la même valeur dans les deux contextes. Soit C_i et C_j les contextes des solutions i et j. Comme nous le savons, au

niveau d'une procédure parallèle tabou avec recherches multiples coopérantes, il y a deux raisons possibles pour la différence entre C_i et C_j: soit que C_j ait été obtenu à partir de C_i suite à un nombre quelconque d'applications d'une ou plusieurs règles de la méthode tabou au contexte C_i; soit que C_i et C_j n'appartiennent pas à la même procédure séquentielle, dans ce cas leur différence s'explique par la stratégie de différentiation de la procédure parallèle.

Au niveau de la méthode tabou définie à la section 4.5, il y a trois règles de transition différentes: add/drop, swap et diversification. La transition la plus fréquente est le add/drop, cette transition ne modifie cependant qu'une seule entrée du vecteur de contexte à chaque application. Chaque transition swap provoque un changement de statut de deux entrées du vecteur de contexte mais son application est moins fréquente que le add/drop. La fréquence d'exécution des transitions add/drop et swap dépend principalement de la stratégie d'exploration de chaque procédure séquentielle, mais est très peu affectée par la définition du couplage entre les procédures séquentielles. Conséquemment, ces deux règles de transition ne font pas varier la valeur de la distance globale de Hamming entre les procédures parallèles. Notons enfin une particularité des transitions add/drop et swap: celles-ci ne modifient pas de manière uniforme tout le vecteur de contexte, les modifications ont tendance à se concentrer sur un sous-ensemble des entrées, en fonction de la structure de l'espace de solutions et de la stratégie d'exploration.

La diversification est la transition dont la fréquence d'exécution est la plus faible mais dont l'impact sur la distance globale de Hamming est considérable. En effet, tel que défini à la section 4.5, la diversification change le statut d'un grand nombre d'entrées du vecteur de contexte. De plus, les entrées du vecteur de contexte modifiées par la diversification sont celles qui ont été peu affectées par les transitions add/drop et swap, ce qui accroît encore l'importance de la diversification. La fréquence d'exécution de la diversification est fortement affectée par la définition du couplage entre les procédures séquentielles et par la stratégie de parallélisation. Par

exemple, pour les procédures par recherches multiples, la fréquence d'exécution de la diversification est fonction uniquement de la stratégie d'exploration, tandis que pour la procédure SIM_{12}, aucune transition de diversification n'est exécutée.

Outre les transitions des procédures séquentielles, les stratégies de différentiation (MPDS, MPSS, SPDS) constituent une source importante de différence entre les contextes de solutions. Les stratégies de différentiation ont un impact considérable sur la valeur de la distance globale de Hamming.

Le ratio de la distance globale de Hamming attribuable aux stratégies de différentiation versus celle attribuable aux transitions des procédures séquentielles sera une des mesures utiles pour faire l'analyse des résultats qui suivent.

Définition 7.23 *La distance globale de Hamming $H^{G(p_j)}$ d'une procédure séquentielle p_j correspond à la somme des distances de Hamming H^- de chaque contexte des t solutions de p_j: $H^{G(p_j)} = \sum_{k=1}^{t} H^-(p_j(k), p_j)$. La part de la distance globale de Hamming attribuable aux changements de contexte provoqués par les transition add/drop, swap et diversification est donnée par $H^{seq} = \sum_{j=0}^{j=p-1} H^{p_j}$, c'est-à-dire la somme des distances globales de Hamming de chaque procédure séquentielle.*

Puisque l'impact du add/drop et du swap sur la distance globale de Hamming ne varie pas en fonction du comportement de la mémoire centrale, la distance globale de Hamming attribuable aux transitions des procédures séquentielles traduit en fait le rôle de la diversification sur les variations de la distance globale de Hamming en fonction des différentes procédures parallèles.

7.5.4 Interprétation des résultats expérimentaux

Rappelons que nous cherchons à valider une thèse selon laquelle l'échange de connaissances acquises entre les procédures séquentielles des recherches multiples coopérantes, auto-organise l'exploration de l'espace de solutions selon un plan qui n'est pas nécessairement relié à une logique d'optimisation. Nous validerons cette thèse à partir d'un ensemble de tests visant à comparer la distance globale de Hamming

entre plusieurs procédures parallèles. Pour chaque test, nous analyserons le rôle des quatre composantes dans la constitution de la distance globale de Hamming. Nous comparerons les tests entre eux et montrerons les liens existants entre les composantes de la distance globale de Hamming et l'échange d'information entre les procédures séquentielles. Le principal enchaînement de notre argumentation est le suivant:

1. Nous allons analyser la distance globale de Hamming des recherches multiples;
2. Montrer que le niveau d'organisation des recherches multiples coopérantes est supérieur à celui des recherches multiples, dû à l'échange de connaissances acquises entre les procédures séquentielles;
3. Montrer que le niveau d'organisation des procédures SIM_{18} et SIM_{22} est le même que celui des recherches multiples coopérantes;
4. Enfin, nous conclurons de la similitude entre les niveaux d'organisation des recherches multiples coopérantes et des procédures SIM_{18} et SIM_{22}, que l'échange de connaissances acquises entre les procédures séquentielles vise plutôt à réduire l'entropie de l'exploration de l'espace de solutions et non pas à fournir des indications sur des régions intéressantes de l'espace de solutions du point de vue de la logique d'optimisation.

Afin de mieux faire ressortir les résultats obtenus avec les simulations basées sur des dynamiques d'auto-organisation, nous avons réalisé des simulations où les contextes retournés aux procédures séquentielles sont générés de manière aléatoire par la mémoire centrale. Ces procédures parallèles se différencient des simulations SIM_{ϕ} uniquement au niveau de la mémoire centrale. Au lieu d'un automate cellulaire élémentaire, la mémoire centrale retourne des contextes aux procédures séquentielles où le statut des entrées est généré par un choix aléatoire dans l'ensemble $\{0, 1\}$. Nous identifierons ces procédures parallèles de génération aléatoire de contextes par $PROB_{alea}$ où *alea* est le germe aléatoire utilisé. Nous avons également

réalisé des simulations basées sur des règles de transition de la classe 4 (SIM_{60} et SIM_{110}) dont le comportement d'auto-organisation n'est pas bien défini. Notre évaluation porte donc sur 4 types de procédures parallèles où les comportements d'interaction entre les procédures séquentielles diffèrent de manière typique:

1. Les procédures parallèles par recherches multiples et SIM_{12} où les procédures séquentielles n'interagissent pas entre elles;

2. Les procédures parallèles collégiales asynchrones avec lesquelles nous comparerons le comportement des procédures SIM pour vérifier la thèse d'un comportement d'auto-organisation au niveau des recherches multiples coopérantes. Les procédures séquentielles interagissent entre elles par l'échange asynchrone de connaissances acquises, c'est-à-dire l'échange de contexte de solutions;

3. Les procédures parallèles SIM_{18} et SIM_{22} contruites à partir de règle de la classe 3 ayant une évolution qui manifeste un comportement connu d'auto-organisation. Les procédures séquentielles s'échangent de l'information consistant de configurations de l'espace d'états du système dynamique ayant un faible niveau d'entropie;

4. Les procédures parallèles $PROB_1$, $PROB_2$, $PROB_3$, SIM_{60} et SIM_{110} où, par construction, la taille Ω reste constante ou diminue à une échelle qui n'est pas perceptible pour les tests que nous ferons (règles 60 et 110). Les procédures séquentielles s'échangent de l'information consistant de configurations de l'espace d'états de B^n générées aléatoirement, ce qui suppose une absence d'organisation résultant de l'interaction entre les procédures séquentielles.

L'analyse des résultats nous permettra de montrer que les procédures parallèles des groupes 2 et 3 ont un comportement similaire du point de vue de l'organisation de la recherche parallèle.

Contenu des tableaux de résultats

Le tableau 7.9 montre le ratio H^{seq}/H^G pour les procédures parallèles avec recherches multiples (p-RI), collégiales asynchrones (Async), SIM_{18}, SIM_{22}, SIM_{12}, SIM_{74}, $PROB_1$ (Aléa) et SIM_{110}. Comme le montrent les lignes de la moyenne des résultats (Moy), le ratio de la distance globale de Hamming attribuable aux règles de transition pour toutes les procédures parallèles testées se situe à environ 23%, 11% et 5% respectivement pour les tests avec 4, 8 et 16 procédures séquentielles. La différence, c'est-à-dire 77%, 89% et 95% est le ratio de la distance globale de Hamming attribuable aux stratégies de différentiation des contextes entre procédures séquentielles. Les tableaux 7.10 à 7.17 donnent la distance globale de Hamming pour plusieurs procédures parallèles. Nous avons réalisé trois simulations avec des germes différents pour la génération aléatoire de contextes (voir les tableaux 7.15 et 7.16). Pour les simulations avec des règles de l'automate cellulaire élémentaire, nous avons mis entre paranthèses la valeur de τ utilisée lorsque celle-ci diffère de $\tau = \lceil \frac{2n-1}{2r} \rceil$. Deux tests ont été effectués avec la procédure parallèle collégiale asynchrone puisque le calcul réalisé par ces procédures parallèles est non déterministe. Nous avons créé un type spécial de procédure parallèle par recherches multiples sur lequel nous reviendrons.

Distances globales de Hamming de p-RI, SIM_{12} et SIM_{74}

Les procédures parallèles avec recherches multiples sont celles dont la distance globale de Hamming est la plus élevée (voir p-RI du tableau 7.10). On peut contraster cette valeur avec celle de la procédure parallèle SIM_{12} (tableau 7.14) que nous avons utilisée pour simuler le comportement des recherches multiples. La distance globale de Hamming de SIM_{12} est la moins élevée de tous les tests. Dans le cas de SIM_{12}, le contexte C_i envoyé à la mémoire centrale par une procédure séquentielle p_i et le contexte reçu de la mémoire centrale par la même procédure p_i sont presque identiques. Alors dans les faits, la mémoire centrale de SIM_{12},

Prob.	p-RI	Async.	18	22	12	74	Aléa	110
	4 procédures séquentielles							
P1	23.32	24.20	23.89	24.08	22.41	24.27	24.21	24.23
P2	23.28	23.08	23.54	23.49	19.63	23.99	23.08	22.82
P3	24.17	23.83	23.17	23.98	21.89	23.94	23.84	23.84
P4	22.97	23.08	22.85	22.73	17.25	23.53	23.08	22.79
P5	23.38	22.91	23.10	23.29	16.53	24.10	22.91	22.86
P6	23.36	23.29	22.19	23.08	18.68	24.20	23.29	22.51
P7	23.64	22.73	22.66	23.32	21.82	23.73	23.20	23.47
P8	23.03	23.81	23.30	23.54	22.59	24.40	23.95	23.84
P9	24.40	23.34	24.10	23.96	21.36	24.12	23.39	24.17
P10	23.39	23.30	23.54	23.43	18.05	24.39	23.34	23.82
P11	23.51	22.83	23.30	23.39	19.40	24.37	22.90	23.84
P12	23.69	23.60	23.52	23.97	18.45	24.37	23.61	23.72
Moy	23.51	23.33	23.26	23.52	19.84	24.12	23.40	23.49
	8 procédures séquentielles							
P1	11.69	12.07	11.92	12.01	11.54	12.14	11.95	12.02
P2	11.57	11.44	11.59	11.52	10.06	11.77	11.40	11.54
P3	12.03	11.84	11.75	11.82	11.14	11.89	11.82	11.84
P4	11.49	11.41	11.17	11.07	9.42	11.66	11.46	11.31
P5	11.33	11.28	10.89	11.21	8.63	11.72	11.60	11.45
P6	11.57	11.45	10.99	11.55	9.48	11.80	11.20	11.10
P7	11.72	11.35	12.01	11.70	11.26	11.94	12.03	11.89
P8	11.67	11.48	11.71	11.87	11.01	12.03	11.88	11.86
P9	12.10	11.84	11.87	11.83	10.70	11.67	11.94	11.98
P10	11.70	11.55	11.54	11.73	9.75	12.10	11.69	11.75
P11	11.80	11.51	11.68	11.63	10.41	11.98	11.79	11.72
P12	11.80	11.73	11.49	11.84	9.67	12.05	11.84	11.62
Moy	11.71	11.58	11.55	11.65	10.26	11.90	11.72	11.67
	16 procédures séquentielles							
P1	5.87	6.01	5.93	5.97	5.88	6.02	5.95	6.00
P2	5.69	5.62	5.66	5.70	4.92	5.73	5.69	5.65
P3	5.96	5.86	5.89	5.93	5.52	5.90	5.90	5.94
P4	5.69	5.62	5.56	5.61	4.73	5.70	5.64	5.69
P5	5.68	5.61	5.53	5.67	4.14	5.78	5.65	5.69
P6	5.65	5.64	5.57	5.63	4.97	5.72	5.61	5.58
P7	5.87	5.73	5.86	5.92	5.67	5.95	5.97	5.92
P8	5.85	5.76	5.89	5.89	5.38	6.02	5.91	5.92
P9	5.94	5.82	5.99	5.89	5.45	5.96	5.92	5.95
P10	5.83	5.76	5.82	5.83	4.79	5.88	5.83	5.91
P11	5.90	5.10	5.83	5.81	5.36	5.97	5.88	5.89
P12	5.85	5.77	5.83	5.85	5.22	5.89	5.86	5.88
Moy	5.82	5.69	5.78	5.81	5.17	5.88	5.82	5.83

Tableau 7.9 Ratio de H^{seq}/H^{G}

Prob	p = 4	p = 8	p = 16
	p-RI		
P1	10472042	37283682	148920018
P2	12165930	47464746	191306992
P3	10128906	38167472	155312802
P4	12231950	47973642	196044322
P5	11609382	46324714	191107700
P6	12103916	47776650	193173546
P7	8177905	29565440	120542518
P8	9198614	34359038	141001659
P9	8724796	33010204	139580696
P10	10810398	39418660	157462065
P11	8826493	35366122	145271065
P12	9507103	37560875	155515648
MOY	10329786	39522603	161269919
	p-RI (importation de contexte)		
P1	9233848	34615058	140370236
P2	11675060	44909177	187506291
P3	9227463	36322228	148778977
P4	11709672	46424320	189529774
P5	10519212	43268550	182504322
P6	11689572	46782462	192016940
P7	6600074	24307446	95879696
P8	9432602	33726186	137992943
P9	8345516	30611357	124849407
P10	8470682	34501464	146561831
P11	8105654	31756964	134503750
P12	9972663	38451352	154745218
MOY	9581834	37139713	152936615

Tableau 7.10 H^G pour la procédure parallèle p-RI

Prob	p = 4	p = 8	p = 16
	1er test		
P1	6544523	27127187	110753063
P2	10196267	40753078	163554218
P3	8169040	31303376	125940704
P4	10511216	41710930	165787158
P5	9779934	39751248	162373970
P6	9953606	40607090	168933312
P7	7060246	27291980	113449220
P8	7135749	29715919	121150512
P9	6218284	31303376	109565144
P10	8414525	34295182	140055168
P11	6842760	28897783	139132459
P12	7700852	31052398	126423019
MOY	8210583	33650795	137259828
	2ième test		
P1	6544527	21945391	102517544
P2	10196267	39547777	159362107
P3	8169050	30501213	123721635
P4	10511216	40570064	165394696
P5	9779934	38056365	159498470
P6	9953606	40607090	168933312
P7	7060252	26846496	112953816
P8	7056272	29362374	121150596
P9	6210148	24708342	107739166
P10	8358353	33262806	140055276
P11	6842762	39751248	117837308
P12	7700862	40607090	125258742
MOY	8198604	33813854	133701889

Tableau 7.11 H^G de la procédure collégiale asynchrone

Prob	p = 4	p = 8	p = 16
	SIM_{18}		
P1	6441348	24140619	104493843
P2	9895244	39932837	166303252
P3	7431169	31460412	128374782
P4	11278506	46260170	183742609
P5	9628766	42341262	170591888
P6	10560432	41580526	168300132
P7	4291314	16391230	73846493
P8	5885788	24129015	104235880
P9	5854062	22821213	96767210
P10	6900029	28228062	120289894
P11	6630424	26078993	112898006
P12	7248392	30649343	125331585
MOY	7670456	31167806	129597964
	SIM_{18} $(\tau = \lceil \frac{(p \times n)-1}{2r} \rceil)$		
P1	6609788	24946979	109263588
P2	10192127	39235016	166142346
P3	7916611	29980014	124085332
P4	11527090	46164485	185070362
P5	10175084	41399678	174277795
P6	9466556	39984628	172392455
P7	5005224	16972135	72651522
P8	6248354	24665978	104571612
P9	5810994	21918144	91758452
P10	6906415	27355827	118635264
P11	6378560	26038858	111984641
P12	7463202	30134989	126997456
MOY	7808333	30733060	129819235

Tableau 7.12 H^G avec la règle 18 de la classe 3

Prob	p = 4	p = 8	p = 16
	SIM_{22}		
P1	6732640	26319684	113240100
P2	10236394	41740142	168844422
P3	7691982	33270520	134041967
P4	11494450	45816860	191065379
P5	11068821	43257914	181106310
P6	11299182	43382848	182610343
P7	5490248	18408216	80754274
P8	6518142	25140624	113217162
P9	7084810	23971320	107074222
P10	7936606	30121012	123153253
P11	7178887	27226834	117279048
P12	8212247	31881154	133597682
MOY	8412034	32544760	137165346
	SIM_{22} ($\tau = \lceil \frac{(p \times n)-1}{2r} \rceil$)		
P1	6954252	27397027	113000926
P2	10335520	41858549	171329643
P3	7918998	31737904	130530017
P4	11571288	45798316	187887747
P5	10653089	42871923	179508204
P6	10891674	42668180	178392134
P7	5395664	19738863	79496191
P8	6482178	25424154	116240344
P9	6767934	24491570	105446844
P10	7144280	30621171	128303340
P11	7544006	29209268	116232662
P12	7787081	33131958	135251544
MOY	8287163	32912406	136801633

Tableau 7.13 H^G avec la règle 22 de la classe 3

Prob	p = 4	p = 8	p = 16
	SIM_{12}		
P1	4398172	17793228	72743208
P2	7154056	30064208	120956920
P3	6209248	25050773	104136450
P4	8641152	31612130	128352932
P5	6344251	30369833	138984060
P6	7804042	33592608	136885969
P7	3307584	12799844	54863070
P8	3962889	17424814	78589764
P9	4448600	18311246	81437266
P10	5870548	24838178	107360308
P11	5737208	21232303	89240238
P12	6354552	25287575	100376740
MOY	5852691	24031395	101160577
	SIM_{74}		
P1	3709116	16308016	69665364
P2	7168906	28508558	124746130
P3	6102050	26203344	106503016
P4	10770506	42623045	163326046
P5	7276482	34031686	146618282
P6	7443654	37646191	157824558
P7	3936088	13487449	58983077
P8	4406296	17997554	80369530
P9	4360054	18855118	81441970
P10	6519124	24904252	104438998
P11	5515452	20510144	88433134
P12	5858848	24902008	103850602
MOY	6088881	25498113	107183392

Tableau 7.14 H^G avec les règles 12 et 74 de la classe 2

Prob	p = 4	p = 8	p = 16
	$PROB_1$		
P1	7457976	29120668	126046706
P2	11548108	43658906	181473838
P3	8509727	33215412	139155130
P4	11746172	46961650	189290436
P5	11076438	44605104	184009239
P6	11637816	46365316	189216020
P7	5798190	23569269	96098550
P8	7347155	29260554	125217538
P9	7625296	28595215	117778878
P10	8295301	34225418	143126157
P11	7901108	31109265	133004154
P12	8628270	35623672	145774148
MOY	8964296	35525870	147515899
	$PROB_2$		
P1	7455273	30498746	129078620
P2	11296050	45564247	83934990
P3	8516196	33073504	138875916
P4	11914342	47502562	185795252
P5	11042530	44737283	183634672
P6	11979758	47420193	190550882
P7	6385441	22543684	98633066
P8	7505530	28997866	122311466
P9	7119014	27827298	118548981
P10	8528966	34582936	144433091
P11	8378411	30852736	129880978
P12	8625454	36768144	146291482
MOY	9062247	35864099	139330783

Tableau 7.15 H^G avec génération aléatoire de contextes

Prob	p = 4	p = 8	p = 16
	$PROB_3$		
P1	7764736	30430548	128847786
P2	11091428	44752890	181178876
P3	8536836	33960225	139041290
P4	12013646	48199876	191454356
P5	1074385	44393112	183624001
P6	11966555	44019518	190019432
P7	6181150	22593943	95984780
P8	7899398	30960964	128661622
P9	7404188	28786852	122289544
P10	8941716	34499963	139125806
P11	8156842	30743648	126419172
P12	8639096	34728800	146908648
MOY	8305831	35672528	147796276

Tableau 7.16 H^G avec génération aléatoire de contextes

comme c'est le cas pour les procédures par recherches multiples, n'influence pas le comportement des procédures séquentielles. La principale différence entre p-RI et SIM_{12} se trouve donc au niveau de la diversification. La procédure p-RI implante la diversification de la méthode tabou définie à la section 4.5 tandis que pour SIM_{12} chaque appel à une transition de diversification par une procédure séquentielle p_i est remplacé par le redémarrage de l'exploration de l'espace de solutions dans la région de la meilleure solution à moyen terme de p_i. Conséquemment, pour les procédures parallèles SIM_{12}, seules les transitions add/drop et swap causent une augmentation de la distance globale de Hamming au niveau des procédures séquentielles.

En comparant ces deux procédures parallèles, on comprend que la diversification affecte grandement la valeur de la distance globale de Hamming. Le ratio H^{seq}/H^G de p-RI est l'un des plus élevé de toutes les procédures parallèles, tandis que celui de SIM_{12} est le plus faible. Le ratio de SIM_{12} traduit l'absence de diversification au niveau des procédures séquentielles, et montre que les transitions ont

Prob	p = 4	p = 8	p = 16
	SIM_{110} ($\tau = \lceil \frac{(iter.\times n)-1}{2r} \rceil$)		
P1	7697706	33668472	129443729
P2	11359951	46074149	188354276
P3	8419335	35281558	138514486
P4	12639970	48755824	192665470
P5	11036800	43748854	187219570
P6	12275827	47052916	190465706
P7	7032798	25756636	105204412
P8	7889817	31937447	131451218
P9	7897842	30101480	124053256
P10	9370024	35865690	147097061
P11	8118562	31886782	133559296
P12	9310125	37545994	151533575
MOY	9420729	37306316	151630171
	SIM_{60} ($\tau = \lceil \frac{(iter.\times n)-1}{2r} \rceil$)		
P1	7595118	29196860	126306924
P2	10363930	44291840	180048657
P3	8303154	34206420	141065750
P4	11896716	48671376	192165606
P5	11212792	44755398	192156642
P6	11618812	44922812	189089524
P7	6564542	23450286	93836933
P8	7841074	29765348	121429056
P9	7053129	25969288	115627634
P10	8230566	33163626	134151270
P11	8065638	30102998	123715407
P12	8582535	32397486	140703454
MOY	8944000	35074478	145858071

Tableau 7.17 H^G avec des règles de la classe 4

Prob	p = 4	p = 8	p = 16
	SIM_{60}		
P1	7680106	30715889	129053214
P2	11439030	44714248	181130282
P3	8352885	33522172	141019566
P4	11978566	47727742	195227156
P5	10363727	45089694	187798424
P6	11707900	46484397	186986516
P7	6584010	24516242	94574602
P8	7493604	29377414	126201127
P9	6921858	26482472	115091122
P10	8393925	32342412	140743216
P11	7488174	27923576	120684353
P12	8331770	33175440	137076054
MOY	8894629	35172641	146298802
	SIM_{60} $(\tau = \lceil \frac{(p \times n)-1}{2r} \rceil)$		
P1	7627752	28614872	123219271
P2	11266848	44278580	180343900
P3	8793036	33719022	141187774
P4	11969298	48339549	185976154
P5	10829523	45610720	185292380
P6	11719292	46221398	189364052
P7	6437077	22658198	95835148
P8	7319524	28263478	121639802
P9	6903960	26636234	109957526
P10	8610398	33171384	140400384
P11	7521481	28780344	128298946
P12	7924264	32619658	138137524
MOY	8910204	34909453	144971071

Tableau 7.18 H^G avec la règle 60 de la classe 4

joué un rôle proportionnellement moins important pour diversifier l'exploration de l'espace de solutions que ce ne fut le cas pour p-RI.

Bien que proportionnellement moindre, la distance globale de Hamming de p-RI attribuable aux stratégies de différentiation est supérieure en valeur absolue à la plupart des autres procédures parallèles. L'explication provient du fait qu'il n'y a pas d'échange de contexte de solutions entre les procédures séquentielles, ce qui maintient à son niveau maximum la diversité des contextes entre les procédure séquentielles. En effet, chaque fois qu'un contexte C_i d'une procédure séquentielle p_i est échangé (copié) à une autre procédure séquentielle p_j où C_i devient C_j, on a $H(C_i, C_j) = 0$ puisque les deux contextes sont identiques. Plus cette situation se reproduit souvent entre les procédures séquentielles, plus la distance globale de Hamming attribuable aux stratégies de différentiation diminue (on peut faire un lien avec le phénomène d'uniformisation des connaissances acquises du chapitre 5). Nous avons fait des tests où la diversification de chaque procédure séquentielle p_j de p-RI s'effectue à partir d'un contexte de solution en provenance d'une procédure séquentielle du voisinage de p_j. Les données de la deuxième partie du tableau 7.10 représentent les résultats de ces tests. On constate une nette diminution de la distance globale de Hamming, cette diminution est attribuable au fait que l'échange de contextes de solutions entre les procédures séquentielles a réduit l'impact des stratégies de différentiation sur la valeur de la distance globale de Hamming.

Pour se convaincre que l'interaction entre les procédures séquentielles réduit l'impact des stratégies de différentiation sur la distance globale de Hamming, nous avons fait des tests avec la procédure parallèle SIM_{74} utilisée pour simuler le comportement de détérioration des performances. Pour la procédure SIM_{74} comme pour toutes les autres simulations, la diversification est remplacée par un redémarrage de la recherche dans une région définie par le contexte en provenance de la mémoire centrale. Dans le cas de SIM_{74}, pour chaque procédure séquentielle p_j, le contexte de la mémoire centrale correspond à celui de la meilleure solution à moyen

terme d'une procédure séquentielle autre que p_j. On constate que pour SIM_{74}, le ratio de la distance globale de Hamming attribuable aux stratégies de différentiation est le plus faible de toutes les procédures parallèles. Ce comportement est essentiellement attribuable à l'échange de contextes de solution entre les procédures séquentielles qui réduit la diversité des contextes inter-procédures. Par contre, la distance globale de Hamming de SIM_{74} est supérieure à celle de SIM_{12} parce que le redémarrage de la recherche à partir d'une solution en provenance d'une autre procédure séquentielle fait que le add/drop et le swap génèrent un plus grand nombre de contextes différents.

Distance globale de Hamming de la procédure collégiale asynchrone

La fréquence d'application des transitions de diversification des procédures avec recherches multiples coopérantes se situe entre celle des procédures avec recherches multiples et celle d'une procédure comme SIM_{74}. C'est le cas de la procédure parallèle collégiale asynchrone. Cette procédure exécute le même type de diversification que la procédure p-RI, cependant la fréquence est moindre puisque, comme pour les autres recherches multiples coopérantes, certaines diversifications sont remplaçées par l'importation du contexte d'une solution en provenance d'une autre procédure séquentielle.

La valeur moindre de la distance globale de Hamming des procédures parallèles collégiales asynchrones par rapport à celle de p-RI est attribuable à la réduction de la fréquence des diversifications et à l'échange de contextes entre procédures séquentielles. En valeur absolue, la distance globale de Hamming attribuable aux transitions est moindre parce que les contextes importés bouleversent moins le statut des variables de décisions que celui provoqué par une vraie diversification (rappelons qu'une diversification a tendance à changer le statut des variables dont le statut a peu changé sous l'effet des transitions add/drop et swap, ce n'est pas le cas pour un contexte importé). En valeur absolue, la distance de Hamming attribuable aux

stratégies de différentiation est réduite parce que le contexte de la même solution peut se retrouver dans plusieurs procédures séquentielles par suite de l'échange de solutions. Comme nous l'avons expliqué dans les précédentes sections, l'échange de contextes entre les procédures séquentielles fait que la distance de Hamming est nulle entre des contextes identiques provenant de procédures séquentielles différentes.

Distance globales de Hamming des simulations

Dans la présente section, le terme simulation fait référence aux procédures $SIM_{18}, SIM_{22}, PROB_1, PROB_2, PROB_3, SIM_{60}$ et SIM_{110}. Toutes ces procédures possèdent les deux caractéristiques suivantes: il n'y a pas d'échange de connaissances acquises entre les procédures séquentielles et aucune diversification n'est exécutée. À chaque fois qu'un appel de diversification est fait, la recherche est redémarrée dans une région de l'espace de solutions défini par un contexte obtenu de la mémoire centrale.

Lemme 7.5 *En terme du nombre d'entrées dont le statut change d'état, les contextes en provenance de la mémoire centrale sont au moins aussi perturbés que le contexte résultant d'une diversification.*

En effet, pour les simulations SIM_ϕ où ϕ est une règle de la classe 3 ou 4, l'exposant de Lyapunov pour ces règles $\lambda = r$. Puisque $\tau = \lceil \frac{2n-1}{2r} \rceil$, alors $(\tau \times r) = n$, c'est-à-dire la taille de chaque contexte. Conséquemment, chaque entrée du contexte C_j est perturbée par le statut des entrées de E_0 correspondant aux contextes dans le voisinage de C_j. En ce qui concerne les simulations $PROB_{alea}$, le germe aléatoire détermine le statut de chaque entrée du vecteur de contexte.

Le lemme 7.5 implique que la perturbation sur les contextes à partir de la mémoire centrale des simulations est équivalente à celle de la diversification des procédures par recherches multiples en terme d'impact sur la distance globale de Hamming. Ce qui distingue les simulations entre elles et avec la procédure p-RI, c'est évidemment la manière dont sont effectuées ces perturbations.

Proposition 7.4 *La valeur moindre de la distance globale de Hamming des procédures parallèles SIM_{18} et SIM_{22} par rapport à celle de la procédure p-RI est attribuable à la dynamique d'auto-organisation des règles 18 et 22.*

Preuve: Bien que les entrées du contexte C_j en provenance de la mémoire centrale soient déterminées par le statut des entrées des contextes C_{j-1}, C_j et C_{j+1} de E_0 (si $\tau = \lceil \frac{2n-1}{2r} \rceil$), certains blocs sont exclus quelque soit la configuration du vecteur E_0 et donc quelque soit le statut des entrées de C_{j-1}, C_j et C_{j+1} de E_0. De plus, les blocs nouvellement exclus ont tendance à être exclus de manière uniforme dans le temps et dans l'espace pour les règles 18 et 22. Conséquemment, si un bloc x est nouvellement exclu dans le contexte C_j, le bloc x risque d'être nouvellement exclu au niveau de tous les contextes retournés par la mémoire centrale. Les mêmes blocs auront donc tendance à être exclus sur tout le vecteur $E_{\lceil \frac{2n-1}{2r} \rceil}$, ce qui fait qu'ils seront exclus de tous les contextes envoyés par la mémoire centrale aux procédures séquentielles. Cette observation qui s'applique dans l'espace est vraie également dans le temps, c'est-à-dire qu'à chaque point d'interaction, les mêmes blocs auront tendance à être exclus. On voit donc comment le comportement d'auto-organisation des règles 18 et 22 peut réduire la distance globale de Hamming attribuable aux transitions et aux stratégies de différentiation. Les mêmes blocs étant exclus de toutes les procédures séquentielles, cela réduit la capacité des stratégies de différentiation à créer une diversité des contextes entre les procédures séquentielles. Par contre, les mêmes blocs étant exclus à chaque diversification, cela a pour effet de réduire l'impact de la diversification sur chaque procédure séquentielle.

Si l'on analyse le contenu du tableau 7.9, on constate que le ratio H^{seq}/H^G est similaire pour les simulations, la procédure p-RI et les procédures collégiales asynchrones. Cependant, l'analyse des tableaux 7.10 à 7.17, montre qu'en valeur absolue, la distance globale de Hamming ne suit pas les mêmes tendances que le ratio H^{seq}/H^G. La distance globale de Hamming des simulations SIM_{18} et SIM_{22} tend vers celle des procédures collégiales asynchrones tandis que pour les simulations

$PROB_x$, SIM_{110} et SIM_{60} elle tend vers celle de la procédure p-RI. Ces résultats expérimentaux nous amènent à conclure que les procédures SIM_{18} et SIM_{22} se comportent de la même manière que les procédures avec recherches multiples coopérantes et nous conduisent à faire la généralisation suivante:

Hypothèse 7.2 *Le niveau d'organisation des procédures parallèles SIM_{18} et SIM_{22} tend à être le même que celui des procédures parallèles collégiales asynchrones.*

Cette hypothèse entre en contradiction directe avec la logique d'optimisation de la manière suivante:

Théorème 7.4 *L'échange de connaissances acquises entre les procédures séquentielles des procédures parallèles collégiales asynchrones oriente l'exploration de l'espace de solutions dans le sens d'un accroissement de l'organisation qui se fait en dehors de la logique d'optimisation.*

Preuve: Les résultats de Clearwater, Hogg & Huberman [24], montrent que si les messages envoyés aux procédures séquentielles transmettent des connaissances acquises tout en provenant d'une source indépendante, les hypothèses d'indépendance des procédures séquentielles sont rencontrées et il n'y a pas d'accroissement de l'organisation de la recherche parallèle. À l'inverse, pour toutes les simulations réalisées dans la présente section, il n'y a pas d'échange de connaissances acquises entre les procédures séquentielles, les critères d'appel à une transition de diversification sont les mêmes que ceux des procédures parallèles avec recherches multiples et finalement le lemme 7.5 s'applique de la même manière à toutes les simulations. Si l'accroissement d'organisation dépendait du contenu de "connaissances acquises" des messages, les procédures SIM_{18} et SIM_{22} qui sont identiques de ce point de vue aux autres simulations, devraient avoir un niveau d'organisation similaire à celui d'une procédure p-RI. De plus, les tests de Clearwater, Hogg & Huberman devraient indiquer un accroissement de l'organisation de la recherche. Il faut donc en déduire que le niveau d'organisation des recherches multiples coopérantes est indépendant

du contenu de "connaissances acquises" des messages échangés entre les procédures séquentielles. Selon la proposition 7.3 et l'hypothèse 7.1, la recherche à long terme des procédures SIM_{18} et SIM_{22} est organisée par la dynamique d'auto-organisation des règles 18 et 22. Puisque la distance globale de Hamming de SIM_{18} et SIM_{22} tend à être la même que celle des procédures parallèles collégiales asynchrones, la dynamique d'auto-organisation, et non pas la logique d'optimisation, gouverne le comportement à long terme des recherches multiples coopérantes.

Discussion

L'objectif de la présente section était de montrer que l'interaction complexe entre les procédures séquentielles (incluant les comportements dynamiques de réactions en chaîne) pouvait être la source de certains principes d'organisation de l'exploration de l'espace de solutions qui échappent à la logique d'optimisation de la méthode de recherche. En montrant que la distance globale de Hamming des procédures collégiales asynchrones tend vers les mêmes valeurs que celles de procédures dont l'organisation globale n'est pas déterminée par une logique d'optimisation, comme SIM_{18} et SIM_{22}, nous avons mis à jour une logique d'exploration à long terme des recherches multiples coopérantes qui semble ne pas suivre celle de la méthode de recherche des procédures séquentielles. En apparence et selon la logique d'optimisation de la méthode tabou, les procédures séquentielles communiquent entre elles pour s'échanger de l'information sur les meilleures régions de l'espace de solutions. Or à long terme, la fréquence d'échange de l'information, l'impact de la structure de voisinage, le nombre de procédures séquentielles, les dynamiques de réactions en chaîne sont des facteurs dont le comportement ne dépend pas du contenu de "connaissances acquises" des messages et qui donc échappent à la logique d'optimisation. Ces facteurs d'interférence sur la méthode de recherche provoquent une dérive réelle de l'exploration de l'espace de solutions.

Les résultats de la présente section sont encore très préliminaires, nous pensons

qu'il faut plus de tests portant sur la distance globale de Hamming des procédures parallèlles avec recherche multiples coopérantes et plus de tests avec des règles de simulation de la classe 3 des automates cellulaires élémentaires. Un ensemble plus large de tests numériques nous permettra d'ajouter à la présente argumentation certaines observations empiriques que nous avons jugé préférable de ne pas inclure ici parce que les tests ne permettaient pas de conclure avec certitude. Avec un plus grand nombre d'observations numériques et la pleine utilisation des résultats théoriques portant sur les automates cellulaires élémentaires, il nous sera possible de développer une argumentation suffisamment rigoureuse pour prouver ou infirmer la thèse selon laquelle la logique d'optimisation ne contrôle pas nécessairement les procédures parallèles avec recherches multiples coopérantes.

Lorsque cette thèse sera pleinement établie, nous pourrons commencer à étudier la forme que prend la dérive de l'exploration de l'espace de solutions des recherches multiples coopérantes par rapport à une logique d'optimisation. Nous avons déjà fait un bon bout de chemin dans cette direction en vérifiant l'hypothèse générale d'un comportement d'auto-organisation des recherches multiples coopérantes, c'est-à-dire d'un comportement où l'objectif premier est de coordonner les recherches entre elles. Ce type de comportement a déjà été observé pour de nombreux systèmes complexes, ces observations et les modèles théoriques utilisés pour décrire ce phénomène d'auto-organisation pourront être utilisés pour analyser le comportement des recherches multiples coopérantes.

Dans cette section, les tests ont été réalisés sans essayer de faire un lien entre la dynamique d'interaction et la qualité des solutions obtenues par la procédure parallèle. Comme en témoignent les tableaux 7.19 à 7.22, les distances globales de Hamming que nous avons obtenues ne sont pas le reflet de la qualité des solutions trouvées. Nous pensons, qu'il est tout à fait réaliste d'envisager que la formalisation abondante de la mécanique statistique et des systèmes dynamiques, les relations que certains chercheurs établissent entre ces domaines de la physique

théorique et la théorie du calcul, pourront être utilisées pour relier le comportement des recherches multiples coopérantes à celui de la logique d'optimisation des méthodes de recherches.

7.6 Conclusion

Dans ce chapitre, nous nous sommes efforcés de montrer le rôle joué par l'interaction entre les stratégies d'exploration au niveau du parcours de l'espace de solutions de chaque procédure séquentielle et conséquemment sur les performances d'une procédure avec recherches multiples coopérantes. C'est ainsi que nous avons montré formellement que l'exploration de l'espace de solutions exécutée par chaque procédure séquentielle p_i est déterminée par l'interaction entre les stratégies d'exploration des recherches multiples et non pas par la seule stratégie d'exploration de p_i. Il en résulte que tout facteur affectant la dynamique d'interaction entre les stratégies d'exploration affecte également l'exploration de l'espace de solutions exécutée par chaque procédure séquentielle. Un changement dans la dynamique d'interaction a donc un impact direct sur les performances de la procédure parallèle en changeant le parcours des procédures séquentielles, y compris celui ayant permis d'obtenir la meilleure solution. C'est ce qui se passe par exemple lorsqu'il y a une augmentation du nombre de procédures séquentielles.

L'analyse théorique développée à la section 7.4 avait pour objectif de formaliser tous les événements au niveau de l'interaction entre les procédures séquentielles pouvant avoir un impact sur les performances de la procédure parallèle. À partir de cette analyse, il a été possible de cibler avec précision les événements de l'interaction entre les procédures séquentielles susceptibles d'entraîner une déterioration des performances. C'est ainsi qu'on a pu montrer que la dynamique d'interaction entre les procédures séquentielles ayant conduit à la meilleure solution peut être détruite par une augmentation du nombre de procédures séquentielles. L'objectif de la section 7.5 était plus fondamental, il visait à expliquer pourquoi une méthode de recherche

(parallèle) peut détruire le chemin d'exploration conduisant à sa meilleure solution! Pourquoi deux procédures parallèles de recherche ne peuvent-elles pas retrouver les mêmes régions intéressantes de l'espace de solutions alors que le même ensemble de stratégies d'exploration est présent dans les deux procédures parallèles!

L'argumentation de la section 7.5 vient donc compléter celle de la section 7.4 en montrant que l'interaction entre les procédures séquentielles des recherches multiples coopérantes est indépendante dans une certaine mesure de la logique d'optimisation de la méthode de recherche des procédures séquentielles. S'il y avait une logique d'optimisation au niveau de l'interaction entre les tratégies d'exploration, la dynamique d'interaction ayant conduit à la meilleure solution ne pourrait évidemment pas être détruite par l'ajout de nouvelles procédures séquentielles.

Nous pensons, que les résultats de la section 7.5 sont suffisamment convaincants pour montrer que l'échange d'information ne contribue pas automatiquement à améliorer l'exploration des procédures séquentielles. Ces résultats contredisent également l'hypothèse probabiliste selon laquelle l'échange d'information ne provoque aucune dérive de l'exploration de l'espace de solutions des procédures parallèles avec recherches multiples coopérantes. Nous pensons qu'en montrant où il ne faut pas chercher, en particulier en tenant compte des outils que nous avons utilisés pour obtenir nos résultats, l'élaboration de stratégies de parallélisation basées sur les recherches multiples coopérantes et l'analyse des performances de ces procédures parallèles pourra se faire de manière plus systématique.

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{18}					
P1	0	21	0	21	0	10
P2	0.53 (2.29)	252	0.21 (2.30)	261	0.14 (0.71)	17
P3	0 (1.55)	35	0 (0.46)	57	0	183
P4	0.51 (1.47)	37	0.05 (0.27)	174	0.05 (0.27)	174
P5	0.38 (0.99)	15	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.92)	54	0.05 (2.94)	57	0.34	44
P7	0.07	25	0.07	25	0.07	25
P8	0.79 (3.04)	71	0.79 (2.87)	292	0	33
P9	0 (3.07)	287	0 (3.03)	59	0 (3.02)	51
P10	1.36 (2.65)	208	0.37 (1.98)	196	1.07	244
P11	0	113	0 (3.14)	159	0.61	158
P12	1.09	267	0.11	16	0	35
MOY	0.3983		0.1392		0.1917	
	SIM_{22}					
P1	0	21	0	21	0	10
P2	0.22 (1.86)	243	0.14 (1.46)	266	0.14 (0.71)	17
P3	0 (1.88)	265	0.01 (0.59)	31	0	187
P4	0.48 (0.49)	249	0.12 (1.61)	18	0.08 (0.74)	42
P5	0 (0.46)	153	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.92)	54	0.01 (0.68)	181	0.01 (0.68)	180
P7	0.07	25	0.07	25	0.07	25
P8	0.85 (2.80)	154	0	73	0	33
P9	0	102	0 (3.56)	155	0	52
P10	2.47 (4.93)	166	0.52 (3.43)	18	0.17 (4.83)	277
P11	1.52	70	0.61	210	0	285
P12	0 (1.03)	173	0	56	0	35
MOY	0.4717		0.1250		0.0408	

Tableau 7.19 Simulations avec des règles de la classe 3

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{60}					
P1	0	21	0	21	0	10
P2	0.25 (2.46)	157	0.25 (2.46)	157	0.14 (0.71)	17
P3	0.01 (0.47)	162	0.01 (0.59)	31	0	189
P4	0.45 (1.54)	257	0.12 (1.61)	18	0.08 (0.71)	42
P5	0.04 (2.57)	196	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.92)	54	0.05 (041)	229	0 (4.22)	196
P7	0.07	25	0	185	0	141
P8	0	81	0	81	0	33
P9	0	82	0	82	0	82
P10	0.37 (2.53)	120	0.52 (3.43)	18	0 (3.15)	233
P11	0.61	109	1.63 (3.65)	33	0.61	168
P12	0 (0.30)	150	0.11	16	0	35
MOY	0.1542		0.2258		0.0708	
	SIM_{110}					
P1	0	21	0	21	0	10
P2	0.12 (0.74)	88	0.13 (0.74)	88	0.12 (0.74)	88
P3	0 (0.58)	185	0 (0.32)	239	0	300
P4	0.21 (1.24)	218	0.24 (1.61)	51	0	300
P5	0.04 (1.50)	196	0.02 (0.27)	9	0.02	300
P6	0.05 (1.04)	122	0.05 (0.82)	135	0.05	300
P7	0.07	25	0.07	25	0.07	25
P8	0.85 (1.69)	67	0	217	0	33
P9	0 (2.61)	127	0 (1.88)	232	0	276
P10	1.16 (1.93)	226	0 (3.08)	217	0.35	163
P11	2.27	146	0.61 (1.59)	114	0.61	300
P12	0.87 (2.15)	64	0 (2.28)	218	0	35
MOY	0.4700		0.0933		0.1017	

Tableau 7.20 Simulations où ($\tau = \lceil \frac{(iter. \times n)-1}{2\tau} \rceil$)

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	$PROB_1$					
P1	0	21	0	21	0	10
P2	0.33 (2.12)	292	0.14 (1.54)	265	0.14 (0.71)	17
P3	0.01 (0.39)	284	0 (0.61)	107	0 (0.52)	212
P4	0.51 (1.47)	37	0.14 (1.60)	286	0 (3.74)	133
P5	0.38 (0.99)	15	0 (0.27)	199	0.02 (0.27)	9
P6	0.01 (0.56)	103	0.01 (2.05)	111	0 (0.87)	209
P7	0.07	25	0.07	25	0	165
P8	0	216	0.85 (2.80)	173	0	33
P9	0 (3.06)	187	0	185	0 (3.02)	207
P10	0.17 (3.26)	229	0.52 (1.88)	259	0.68	108
P11	0	237	0.61 (1.59)	137	0.33 (1.94)	173
P12	0 (1.03)	283	0 (0.30)	67	0	35
MOY	0.1233		0.1950		0.0975	
	$PROB_2$					
P1	0	21	0	21	0	10
P2	0.33 (1.48)	259	0.37 (1.53)	63	0.14 (0.71)	17
P3	0 (0.46)	41	0 (0.46)	81	0	97
P4	0.12 (1.08)	241	0.12 (1.12)	179	0.08 (0.74)	42
P5	0.38 (0.99)	15	0 (1.01)	133	0 (1.07)	121
P6	0.05 (0.97)	99	0.05 (0.46)	103	0 (0.59)	162
P7	0.07	25	0.07	25	0.07	25
P8	1.31 (3.44)	33	0	71	0	33
P9	0 (3.07)	124	0	164	0	147
P10	0.38 (3.45)	169	0	228	0.38 (2.00)	164
P11	1.38 (2.11)	254	0.61	297	0.61(1.59)	67
P12	0 (0.30)	48	0 (0.30)	48	0	35
MOY	0.3350		0.1017		0.1067	

Tableau 7.21 Simulations basées sur la génération aléatoire de contextes

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	$PROB_3$					
P1	0	21	0	21	0	10
P2	0.14 (1.76)	107	0.21 (1.06)	132	0.14 (0.71)	17
P3	0.01 (0.59)	31	0.01 (0.53)	283	0 (0.32)	296
P4	0.51 (1.47)	37	0.12 (1.30)	184	0.08 (0.74)	42
P5	0.02 (1.29)	265	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.30)	237	0.02 (2.09)	101	0 (0.82)	293
P7	0.07	25	0	220	0.07	25
P8	0.79 (2.87)	145	0	246	0	33
P9	0.97	157	0 (3.07)	145	0 (3.02)	51
P10	2.02 (3.69)	280	0 (1.70)	171	0.35 (3.06)	290
P11	1.52	272	0.61	72	0 (3.25)	175
P12	0 (1.03)	204	0.12	16	0	35
MOY	0.5083		0.0925		0.0550	

Tableau 7.22 Simulation avec génération aléatoire de contextes

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{60}					
P1	0	21	0	21	0	10
P2	0.39 (1.44)	273	0.24 (2.63)	276	0.14 (0.71)	17
P3	0 (0.72)	286	0	260	0 (0.51)	120
P4	0.51 (1.47)	21	0.24 (1.61)	51	0.08 (0.74)	42
P5	0.38 (0.99)	15	0.02 (0.27)	9	0 (1.07)	121
P6	0 (0.78)	284	0.01 (0.31)	212	0.01(0.31)	212
P7	0.07	25	0.07	25	0.07	25
P8	0	231	0	261	0	33
P9	0.97 (4.11)	164	0	202	0	144
P10	0.49 (4.81)	258	0.52 (3.43)	18	0.52 (3.43)	18
P11	1.38 (3.37)	153	0.61	280	0	228
P12	0.67 (3.91)	133	0.11	16	0	35
MOY	0.4050		0.1517		0.0683	
	SIM_{110}					
P1	0	21	0	21	0	10
P2	0.55 (1.33)	248	0.24 (0.69)	198	0.14 (0.71)	17
vP3	0 (0.51)	276	0	285	0	98
P4	0.45 (1.57)	185	0.12 (1.61)	18	0.08 (0.74)	42
P5	0.40 (0.79)	74	0.02 (0.27)	9	0.02 (0.27)	9
P6	0.05 (1.14)	290	0.05 (1.40)	156	0 (0.11)	264
P7	0.07	25	0.07	25	0.07	25
P8	0 (2.08)	262	0	174	0	33
P9	0	89	0 (3.03)	206	0 (3.02)	51
P10	1.43 (1.47)	191	0	287	0	115
P11	0.61	243	0	86	0	102
P12	0.18	250	0 (1.32)	195	0	35
MOY	0.3117		0.0417		0.0258	

Tableau 7.23 Simulations avec des règles de la classe 4

Prob	p = 4		p = 8		p = 16	
	Gap	Iter	Gap	Iter	Gap	Iter
	SIM_{18}					
P1	0	21	0	21	0	10
P2	0.37 (2.29)	252	0.41 (1.32)	180	0.14 (0.71)	17
P3	0.01 (0.59)	31	0	256	0	139
P4	0.41 (0.70)	173	0.12 (1.61)	18	0.08 (0.71)	42
P5	0.38 (0.99)	15	0	223	0 (1.88)	297
P6	0 (1.26)	243	0.05 (2.94)	57	0 (0.08)	270
P7	0.07	25	0.07	25	0	177
P8	1.89	86	0 (2.08)	224	0	33
P9	0	132	0	155	0 (2.61)	237
P10	1.54	294	0.52 (3.43)	18	0.38 (2.00)	266
P11	1.63 (3.65)	33	0.61 (0.37)	297	0.61	162
P12	0.11 (3.31)	286	0.11	16	0	35
MOY	0.5342		0.1575		0.1008	
	SIM_{60}					
P1	0	21	0	21	0	10
P2	0.24 (1.72)	193	0.19 (1.79)	151	0.14 (0.71)	17
P3	0	186	0	172	0	156
P4	0.51 (1.47)	37	0.14 (0.99)	159	0.08 (0.74)	42
P5	0.38 (0.99)	15	0.02 (0.27)	9	0.02 (0.27)	9
P6	0 (0.50)	192	0 (0.58)	164	0 (0.43)	289
P7	0.07	25	0.07	25	0.07	25
P8	0 (2.08)	88	0 (4.14)	16	0	33
P9	0 (2.61)	142	0	275	0	93
P10	0.75 (3.63)	286	0 (4.82)	246	0	97
P11	0.61	267	0.61	169	0.61	143
P12	0.59 (3.83)	90	0 (1.03)	235	0	35
MOY	0.2625		0.0858		0.0767	

Tableau 7.24 Simulations où $r = \lceil \frac{(p \times n)-1}{2r} \rceil$

Chapitre 8
Conclusion

Il existe encore relativement peu de travaux de recherche portant de manière spécifique sur les stratégies de parallélisation appliquées aux méthodes heuristiques. D'une façon générale, la présente étude contribue à définir une perspective d'ensemble de ce domaine de recherche et à faire apparaître la spécificité du traitement parallèle appliqué aux méthodes heuristiques. C'est ainsi que nous avons pu montrer que l'information heuristique constitue une source importante de parallélisme des méthodes heuristiques, source qu'on ne peut pas utiliser pour la parallélisation des algorithmes. Cette exploitation du traitement spéculatif des méthodes de recherche pour obtenir du parallélisme se traduit par des comportements inhabituels des procédures parallèles. Pour cette raison, la conception, l'analyse et la mesure des performances des heuristiques parallèles,à partir des méthodes développées pour les algorithmes parallèles, semblent mal adaptés et trop simples pour tenir compte de tous les raffinements de stratégies de parallélisation basées sur l'information heuristique.

En plus de montrer l'importance de l'information heuristique comme source de parallélisme des méthodes heuristiques, la présente recherche introduit une taxonomie des stratégies de parallélisation applicables à la méthode tabou. Cette taxonomie montre la diversité des approches de parallélisation lorsqu'on tient compte, non seulement de facteurs reliés à l'architecture de l'ordinateur parallèle et au style de décomposition, mais également de l'impact de l'information heuristique sur le comportement de la procédure parallèle. C'est ainsi qu'on peut distinguer entre des stratégies de parallélisation basées sur une ou plusieurs sources d'information heuristique, sur la présence ou l'absence d'échange de connaissances entre les tâches concurrentes, sur différents niveaux de partage de l'information heuristique, etc.

Notre étude a également tenté d'apporter des réponses et des solutions à certains problèmes propres aux procédures parallèles avec recherches multiples coopérantes. Nous avons élaboré un cadre général en vue de guider la conception de procédures parallèles par recherches multiples coopérantes appliquées à la méthode tabou. Ce cadre vise à définir une démarche structurée de conception de la phase d'échange d'information entre les procédures séquentielles, phase qui n'existe pas pour les autres classes de stratégies de parallélisation. Ce cadre développé pour la recherche tabou pourra éventuellement être adapté pour des procédures parallèles par recherches multiples coopérantes appliquées à des méthodes comme le recuit simulé ou les algorithmes génétiques.

Sur le plan théorique, nous avons montré, dans le cas des recherches multiples coopérantes, que l'exploration de l'espace de solutions exécutée par chaque procédure séquentielle est le résultat d'une interaction complexe entre les stratégies d'exploration de la procédure parallèle. Sur la base de cette observation, nous avons démontré que l'augmentation du nombre de procédures séquentielles pouvait provoquer la destruction de la dynamique d'interaction ayant généré le chemin de la meilleure solution, causant ainsi une diminution des performances de la procédure parallèle si une autre solution aussi bonne n'est pas trouvée.

Le fait que le parcours de l'espace de solutions dépende d'une interaction complexe entre les stratégies d'exploration permet d'utiliser des modèles où le comportement des recherches multiples coopérantes trouve une approximation à partir de modèles qualitatifs de la théorie des systèmes dynamiques. À partir de ce type de modèles, nous avons simulé des interactions entre les stratégies d'exploration qui n'avaient pas pour objet d'échanger de l'information sur l'espace de solutions. Ces simulations se sont comportées de façon similaire aux procédures où il y a échange de connaissances acquises. Nous avons conclu que le contenu de "connaissances acquises" des messages ne joue pas un rôle important dans la dynamique d'interaction entre les stratégies d'exploration. C'est sans doute ce qui explique que

la dynamique d'interaction ayant conduit à une bonne solution puisse être facilement détruite: l'interaction entre les stratégies d'exploration n'obéit pas à la logique d'optimisation de la méthode de recherche des procédures séquentielles.

Perspectives

Comme le montrent les résultats numériques au chapitre 4, les parallélisations de la méthode tabou par recherches multiples sont très performantes. Cependant, bien que le parallélisme théorique soit presqu'illimité pour ce type de stratégies de parallélisation, nous pensons qu'en pratique le nombre de recherches séquentielles est limité par une difficulté inhérente aux heuristiques, soit celle de créer plusieurs stratégies différentes "efficaces" pour un même problème. Plus le nombre de recherches séquentielles croît, plus il y aura redondance entre les chemins d'exploration et plus il y aura de stratégies d'exploration inefficaces qui vont utiliser la puissance de calcul en pure perte.

Les recherches multiples coopérantes sont moins affectées par ce genre de limitations. L'échange d'information permet une meilleure utilisation des ressources de calcul en corrigeant les stratégies inefficaces et en contrôlant le phénomène de redondance. Il y a également un espèce de phénomène d'apprentissage qui se manifeste à travers l'interaction entre les stratégies d'exploration donnant lieu à l'émergence de nouvelles stratégies d'exploration pendant l'exécution du processus d'exploration de l'espace de solutions. Donc, en terme de la capacité à obtenir des méthodes de résolution efficaces, les stratégies de parallélisation par recherches multiples coopérantes sont beaucoup plus prometteuses que les recherches multiples.

Cependant, l'échange d'information entre les recherches séquentielles ne génère pas automatiquement une amélioration des performances de la procédure parallèle comme nous avons pu le voir tout au long de cette recherche. En fait, le développement d'une stratégie de parallélisation efficace par recherches multiples coopérantes

est un processus d'une grande complexité. Nous pensons que la méthode structurée de conception de l'échange d'information et l'analyse théorique présentée dans cette thèse seront des outils très utiles pour faciliter le développement et l'analyse de ce type de procédures parallèles. En particulier nous croyons que notre approche théorique peut conduire à deux types de recherches: l'étude de stratégies de parallélisation par recherches multiples coopérantes où l'interaction entre les recherches séquentielles est basée sur des modèles qualitatifs des systèmes dynamiques; et l'application des nombreux résultats de la mécanique statistique, de la théorie des systèmes dynamiques et de la théorie de l'information pour formaliser et généraliser certaines conclusions qui se dégagent de l'expérimentation. Cependant, le caractère général de ces modèles pourra rendre difficile leur application directe à la parallélisation des méthodes heuristiques de recherche, il en demeura pas moins qu'un bon nombre de ces résultats peuvent servir de guide dans le développement de plans d'expérimentations. Par exemple certaines notions de la théorie de l'information risquent de trouver des applications presqu'immédiates dans la mesure de la redondance et du contenu "informatif" des contextes échangés entre les recherches séquentielles. Il n'est pas impossible que ces notions puissent être utilisées pour définir des méthodes permettant de prévenir une détérioration des performances des recherches multiples coopérantes.

Enfin, la présente recherche confirme l'existence de plusieurs niveaux simultanés de calcul à l'intérieur de la même exécution de certains types de procédures de traitement parallèles. Si nous faisons le lien avec les quelques notions portant sur les réseaux de neurones introduites au chapitre 2, certains résultats de cette recherche semblent constituer un encouragement à poursuivre l'analyse et l'expérimentation vers de nouvelles alternatives à la théorie classique du calcul et vers de nouveaux modèles de programmation tel le calcul émergent.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, 1989.

[2] E.H.L Aarts, F.M.J de Bont, J.H.A. Habers, and P.J.M van Laarhoven. Parallel Implementations of Statistical Cooling Algorithm. *Integration VLSI*, 4:209–238, 1986.

[3] R.H. Abraham. *Dynamics–The Geometry of Behavior.* Addison-Wesley, 1992.

[4] D. Abramson and J. Abela. A Parallel Genetic Algorithm for Solving the School Timetabling Problem. In G. Gupta and C. Keen, editors, *15th Australian Computer Science Conference*, pages 1–11. Department of Computer Science, University of Tasmania, 1992.

[5] V.M. Alekseev and M.V. Yakobson. Symbolic Dynamics and Hyperbolic Dynamic Systems. *Phys. Rep.*, 75:287., 1981.

[6] G.M. Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS Conference Proceedings*, pages 483–485, 1967.

[7] D.E. Arvind and R.A. Iannucci. A Critique of Multiprocessing Von Neumann Style. In *Proceedings of the 10th Symposium on Computer Architecture*, 1983.

[8] R. Azencott. Parallel Simulated Annealing: An Overview of Basic Techniques. In Robert Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 37–46. John Wiley & Sons, Inc., 1992.

[9] S. Baluja. Structure and Performance of Fine-Grain Parallelism in Genetic Algorithm. In Stephanie Forrest, editor, *Proceedings of the fifth International Conference on Genetic Algorithms*, pages 155–162. Morgan Kaufmann Publishers, 1993.

[10] R. Battiti and G. Tecchiolli. Parallel Biased Search for Combinatorial Optimization: Genetic Algorithms and TABU. *Microprocessors and Microsystems*, 16:351–367, 1992.

[11] R. Battiti and G. Tecchiolli. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[12] T. Belding. The Distributed Genetic Algorithm Revisited. In D. Eshelmann, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers, 1995.

[13] A. J. Bernstein. Analysis of Program for Parallel Processing. *IEEE Transactions on Computers*, pages 746–757, 1966.

[14] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation, Numerical Methods*. Prentice Hall, 1989.

[15] G. Bilardi. Some Observations on Models of Parallel Computation. In Jorge L.C. Sanz, editor, *Opportunities and Constraints of Parallel Computing*, pages 11–13. Springer-Verlag, 1989.

[16] Peter van Emde Boas. Machine Models and Simulations. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*, pages 3–66. Elsevier Science, 1990.

[17] F. Bonfatti, G. Gadda, and P.D. Monari. Simulation of Dynamic Phenomena by Cellular Automata. *Comput. & Graphics*, 18(6):831–836, 1994.

[18] G. Brassard and P. Bratley. *Algorithmics: Theory and Practice*. Prentice Hall, 1988.

[19] F.W. Burton. Speculative Computation, Parallelism, and Functional Programming. *IEEE Transactions on Computers*, C-34(12):1190–1193, 1985.

[20] E. Cantú-Paz. A Summary of Research on Parallel Genetic Algorithms. Report 95007, University of Illinois at Urbana-Champain, 1995.

[21] J. Chakrapani and J. Skorin-Kapov. A Connectionist Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 19(3/4):287–295, 1992.

[22] J. Chakrapani and J. Skorin-Kapov. Mapping Tasks to Processors to Minimize Communication Time in a Multiprocessor System. Report, Harriman School for Management and Policy State University of New York at Stony Brook, 1993.

[23] J. Chakrapani and J. Skorin-Kapov. Massively Parallel Tabu Search for the Quadratic Assignment Problem. *Annals of Operations Research*, 41:327–341, 1993.

[24] S.H. Clearwater, T. Hogg, and B.A. Huberman. Cooperative Problem Solving. In B.A. Huberman, editor, *Computation: The Micro and the Macro View*, pages 33–70. World Scientific, 1992.

[25] S.H. Clearwater, B.A. Huberman, and T. Hogg. Cooperative Solution of Constraint Satisfaction Problems. *Science*, 254:1181–1183, 1991.

[26] J. Cohoon, W. Martin, and D. Richards. A Multi-population Genetic Algorithm for Solving the k-Partition Problem on Hyper-cubes. In Richard K. Belew and Lashon B. Booker, editors, *Proc. Fourth Int. Conference on Genetic Algorithms and their Applications*, pages 134–144. Morgan Kaufmann Publishers, 1991.

[27] T.G. Crainic, L. Delorme, and P.J. Dejax. A Branch-and-Bound Method for Multicommodity Location with Balancing Requirements. *European Journal of Operational Research*, 65(3):368–382, 1993.

[28] T.G. Crainic, M. Gendreau, P. Soriano, and M. Toulouse. A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research*, 41:359–383, 1993.

[29] T.G. Crainic, M. Toulouse, and M. Gendreau. Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements. *OR Spektrum*, 17(2/3), 1995.

[30] J.P. Crutchfield. The Calculi of Emergence: Computation, Dynamics, and Induction. *Physica D*, 75:11–54, 1994.

[31] J.P. Crutchfield and J.E. Hanson. Turbulent Pattern Bases for Cellular Automata. *Physica D*, 69:279–301, 1993.

[32] J.P. Crutchfield and M. Mitchell. The Evolution of Emergent Computation. Report SFI-94-03-012, Santa Fe Institute, 1994.

[33] F. Damera-Rogers, S. Kirkpatrick, and V.A. Norton. Parallel Algorithms for Chip Placement by Simulated Annealing. *IBM Journal of Research and Development*, 31:391–402, 1987.

[34] S. Devadas and A.R. Newton. Topological Optimization of Multiple Level Array Logic: on Uni and multi-processors. In *Proceedings of the IEEE Int. Conf. on Computer-Aided Design*, pages 38–41, 1986.

[35] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems.* Benjamin-Cummings, 1986.

[36] E. Felten, S. Karlin, and S.W. Otto. The Traveling Salesman Problem on a Hypercube, MIMD Computer. In *Proc. 1985 of the Int. Conf. on Parallel Processing*, pages 6–10. CRC Press, 1985.

[37] C.-N. Fiechter. A Parallel Tabu Search Algorithm for Large Travelling Salesman Problems. *Discrete Applied Mathematics*, 51:243–267, 1994.

[38] M.J. Flynn. Very High-Speed Computing Systems. *Proceedings of the IEEE*, 54:1901–1909, 1966.

[39] T. C. Fogarty and R. Huang. Implenting the Genetic Algorithm on Transputer Based Parallel Systems. In H.-P. Schwefel and R. Männer, editors, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 145–149. Springer-Verlag, 1990.

[40] S. Forrest. Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Network. In Stephanie Forrest, editor, *Emergent Computation*, pages 1–11. MIT/North-Holland, 1991.

[41] B. Gendron and T.G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066, 1994.

[42] B. Gendron and T.G. Crainic. A Branch-and-Bound Algorithm for Depot Location and Container Fleet Management. *Location Science*, 1995.

[43] P.B. Gibbons. Towards Better Shared Memory Programming Models. In Jorge L.C. Sanz, editor, *Opportunities and Constraints of Parallel Computing*, pages 55–58. Springer-Verlag, 1989.

[44] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 20(4):74–94, 1990.

[45] F. Glover, E. Taillard, and D. de Werra. A User's Guide to Tabu Search. *Annals of Operations Research*, 41:3–28, 1993.

[46] V. Gordon and D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In Stephanie Forrest, editor, *Proceedings of the fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann Publishers, 1993.

[47] V. Gordon, D. Whitley, and A. Bohm. Dataflow Parallelism in Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 533–542. North-Holland, 1992.

[48] C. Graffigne. Parallel Annealing by Periodically Interacting Multiple Searches: an Experimental Study. In Robert Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 47–79. John Wiley & Sons Inc., 1992.

[49] A. Grama and V. Kumar. Parallel Search Algorithms for Discrete Optimization Problems. *ORSA Journal on Computing*, 7(4):365–385, 1995.

[50] P. Grassberger. Problems in Quantifying Self-Organized Complexity. *Helvetica Physica Acta*, 62:498–511, 1989.

[51] D.R. Greening. Parallel Simulated Annealing Techniques. *Physica D*, 42:293–306, 1990.

[52] J.R. Gurd, C. Kirkham, and J. Watson. The Manchester Prototype Dataflow Computer. *Communication of the ACM*, 28(1):36–45, 1985.

[53] J. L. Gustafson. Reevaluating Amdahl's Law. *Comm. ACM*, 31(5):532–533, 1988.

[54] R. Hauser and R. Männer. Implementation of Standard Genetic Algorithm on MIMD Machines. In Y. Davidor, H.-P. Schwefel, and R Männer, editors, *Proceedings of the third Workshop on Parallel Problem Solving from Nature*, pages 504–514. Springer-Verlag, 1994.

[55] T. Hogg and C. Williams. Solving the Really Hard Problems with Cooperative Search. In *Proc. of the 11th Netl. Conf. on Artificial intelligence (AAAI93)*, pages 231–236. AAAI Press, 1993.

[56] B.A. Huberman. The Performance of Cooperative Processes. *Physica D*, 42:38–47, 1990.

[57] K. Hwang. *Advance Computer Architecture.* McGraw-Hill, 1993.

[58] K. Hwang and P.A. Briggs. *Computer Architecture and Parallel Processing.* McGraw-Hill, 1984.

[59] R. Karp and Y. Zhang. Randomized Parallel Algorithms for Backtrack Search and Branch-and-Bound. *Journal of the Association for Computing Machinery,* 40(3):765–789, 1993.

[60] R.M. Karp and Ramachandran V. Parallel Algorithms for Shared-Memory Machines. In Jan Van Leeuwen, editor, *Handbook of Theoretical Computer Science Vol. B: Formal Models and Semantics,* pages 869–941. Elsevier, 1990.

[61] D.E. Knuth. *The Art of Computer Programming.* Addison-Wesley, 1973.

[62] R. Kravitz and R. Rutenbar. Multiprocessor-Based Placement by Simulated Annealing. In *Proceedings of the 25th ACM/IEEE Design Automation Conference,* 1986.

[63] S.A. Kravitz and R. Rutenbar. Placement by Simulated Annealing on a Multiprocessor. *IEEE Transactions on Computer-Aided Design,* 6:534–549, 1987.

[64] L. Lamport and N. Lynch. Distributed Computing: Models and Methods. In Jan Van Leeuwen, editor, *Handbook of Theoretical Computer Science Vol. B: Formal Models and Semantics,* pages 1157–1199. Elsevier, 1990.

[65] C.G. Langton. Studying Artificial Life with Cellular Automata. *Physica D,* 22:120–149, 1986.

[66] D. Levine. A Parallel Genetic Algorithm for the Set Partitioning Problem. Report (Phd Thesis) ANL-94/23, Argonne National Laboratory, 1994.

[67] M. Malek, M. Guruswamy, M. Pandya, and H. Owens. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research,* 21:59–84, 1989.

[68] B. Manderick and P. Spiessens. Fine-Grained Parallel Genetic Algorithm. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann Publishers, 1989.

[69] J.B. Marion. *Classical Dynamics of Particules and Systems.* Academic Press, 1970.

[70] W.S. McCulloch and W. Pitts. A logical Calculus of the Ideas Immanent in Nervous Activity. *Bull. Math. Biophys.*, 5:115–133, 1943.

[71] Grigoriadis M.D. and Hsu T. RNET - The Rutgers Minimum Cost Network Flow Subroutines. Technical report, Rutgers University, New Brunswick, New Jersey, 1979.

[72] H. Mühlenbein. Evolution in Time and Space - The Parallel Genetic Algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems.* Morgan Kaufman, 1991.

[73] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithm in Combinatorial Optimization. *Parallel Computing*, 4:269–279, 1987.

[74] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. New Solutions to the Mapping Problem of Parallel Systems - the Evolution Approach. *Parallel Computing*, 6:269–279, 1987.

[75] M. Munetomo, Y. Takai, and Y. Sato. An efficient Migration Scheme for Subpopulation-Based Asynchronous Parallel Genetic Algorithms. In Stephanie Forrest, editor, *Proceedings of the fifth International Conference on Genetic Algorithms*, page 649. Morgan Kaufmann Publishers, 1993.

[76] G. Nicolis and Prigogine I. *Self-Organization in non-Equilibrium Systems.* John Wiley & Sons, 1977.

[77] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[78] C. Pettey, M. Leuze, and J. Grefenstette. A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proc. Second Int. Conference on Genetic Algorithms and their Applications*, pages 155–161, 1987.

[79] C.J Prez, A. Corral, A. Daz-Guilera, K. Christensen, and A. Arenas. On Self-Organized Criticality and Synchronization in Lattice Models of Coupled Dynamical Systems. *Int. J. Modern Physics*, to appear, 1996.

[80] F. Rosenblatt. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Rev.*, 65:386., 1958.

[81] P. Roussel-Ragot and G. Dreyfus. Parallel Annealing by Multiple Trials: An Experimental Study on a Transputer Network. In Robert Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 91–108. John Wiley & Sons, Inc., 1992.

[82] R. Rutenbar and S. Kravitz. Layout by Annealing in a Parallel Environment. In *Proceedings IEEE Conf. Computer Design*, pages 434–437, 1986.

[83] E. Sacks. A Dynamic Systems Perspective on Qualitative Simulation. *Artificial Intelligence*, 42:349–362, 1990.

[84] C. Scheurich and M. Dubois. Correct Memory Operation of Cache-Based Multiprocessors. In *The 14th Annual International Symposium on Computer Architecture*, pages 234–243, 1987.

[85] M. Schwehm. Implementation of Genetic Algorithms on Various Interconnection Networks. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, editors, *Parallel Computing and Transputers Applications*, pages 195–203. Amsterdam: IOS Press, 1992.

[86] R. Serra and G. Zanarini. *Complex Systems and Cognitive Processes.* Springer-Verlag, 1990.

[87] B. Shapiro and J. Navetta. A Massively Parallel Genetic Algorithm for RNA Secondary Structure Prediction. *The Journal of Supercomputing*, 8:195–207, 1994.

[88] M. Snir. Parallel Computation Models – Some Useful Questions. In Jorge L.C. Sanz, editor, *Opportunities and Constraints of Parallel Computing*, pages 139–144. Springer-Verlag, 1989.

[89] T. Starkweather, D. Witley, and K. Mathias. Optimisation using Distributed Genetic Algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 176–185. Springer-Verlag, 1991.

[90] Yuba T. and al. Sigma-1: a Dataflow Computer for Scientific Computations. *Computer Physics Communications*, 37:141–148, 1985.

[91] E. Taillard. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455, 1991.

[92] E. Taillard. Parallel Iterative Search Methods for Vehicle Routing Problems. *NETWORKS*, 23:661–673, 1993.

[93] E. Taillard. *Recherches Itératives Dirigées Parallèles.* PhD thesis, École Polytechnique Fédérale de Lausanne, 1993.

[94] E. Taillard. Parallel Taboo Search Techniques for the Job Shop Sheduling Problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.

[95] R. Tanase. Parallel Genetic Algorithm for a Hypercube. In J.J. Grefenstette, editor, *Proc. Second Int. Conference on Genetic Algorithms and their Applications*, pages 177–183. Lawrence Erlbaum Associates, 1987.

[96] R. Tanase. Distributed Genetic Algorithms. In J.D. Schaffer, editor, *Proc. Third Int. Conference on Genetic Algorithms and their Applications.* Morgan Kaufmann Publishers, 1989.

[97] H.W.J.M. Trienekens and A. de Bruin. Towards a Taxonomy of Parallel Branch and Bound Algorithms. Report EUR-CS-92-01, Department of Computer Science, Erasmus University Rotterdam, 1992.

[98] A.M. Turing. On Computable Numbers with an Application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 42(2):230–265, 1937.

[99] R.J.M. Vaessens. *Generalized Job Shop Scheduling: Complexity and Local Search.* PhD thesis, Eindhoven University of Technology, 1995.

[100] M.G.A. Verhoeven. *Parallel Local Search.* PhD thesis, Eindhoven University of Technology, 1996.

[101] G.Y. Vichniac. Simulating Physics with Cellular Automata. *Physica D*, 10:96–116, 1984.

[102] B. Virot. Parallel Annealing by Mutiple Trials: Experimental Study of a Chip Placement Problem Using a Sequent Machine. In Robert Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 109–127. John Wiley & Sons, Inc., 1992.

[103] S. Voß. Tabu Search: Applications and Prospects. In D.-Z. Du and P.M. Pardalos, editors, *Network Optimization Problems*, pages 333–353. World Scientific Publishing Co., 1993.

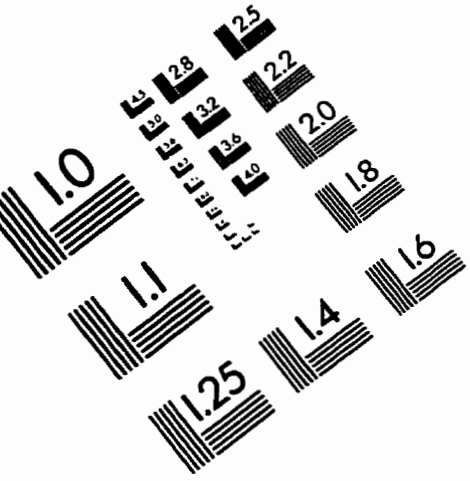
[104] B.W. Wah and G.-J. Li. Design Issues of Multiprocessors for Artificial Intelligence. In Kai Hwang and Douglas Degroot, editors, *Parallel Processing for Supercomputers*, pages 107–167. McGraw-Hill Series in Supercomputing & Parallel Processing, 1989.

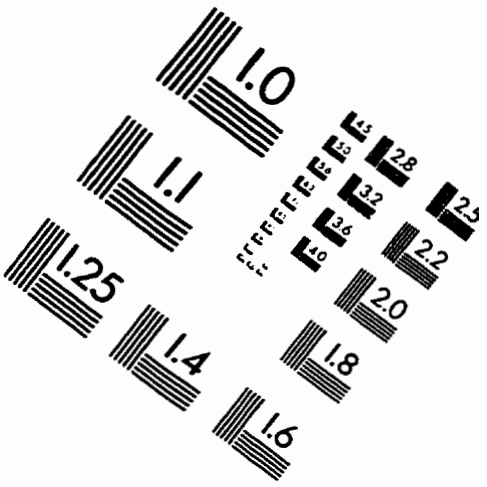
[105] S. Wolfram. Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics*, 55(3):601–644, 1983.

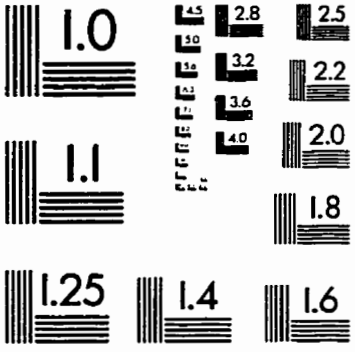
[106] S. Wolfram. Computation Theory of Cellular Automata. *Comm. Math. Phys*, 96:15–57, 1984.

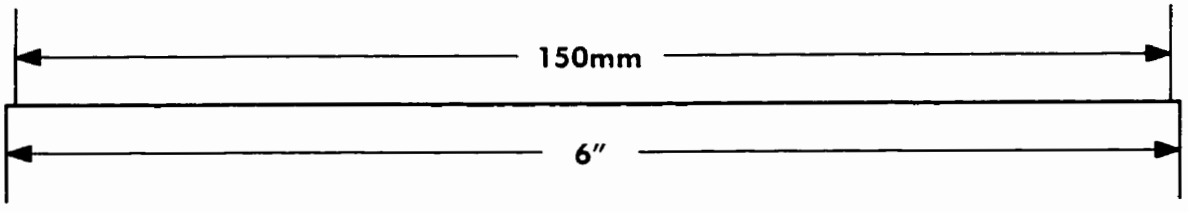
[107] S. Wolfram. Universality and Complexity in Cellular Automata. *Physica D*, 10:1–35, 1984.

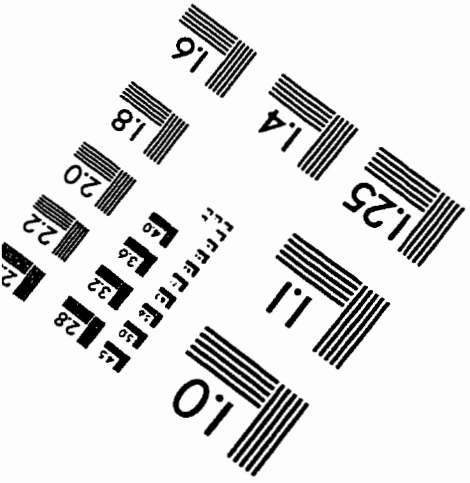
IMAGE EVALUATION
TEST TARGET (QA-3)

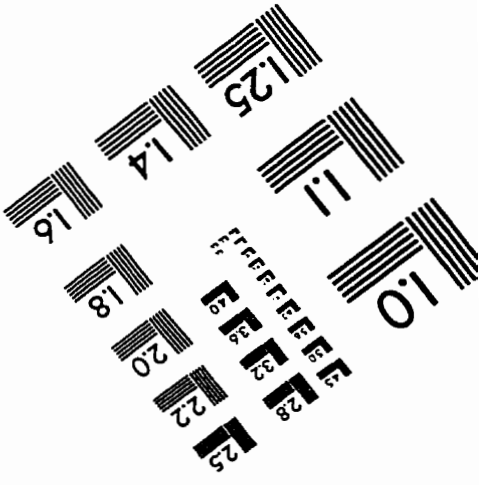












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989