Chapitre II

Parallélismes des métaheuristiques

Chapitre II Parallélisme des métaheuristique : vue générale et classification

2.1 Introduction

Dans un domaine comme l'informatique, les besoins de performance sont toujours croissants. Les programmes informatiques doivent gérer de plus en plus de données et ce, de plus en plus rapidement. Il est toutefois possible de rendre un programme plus performant en améliorant l'algorithme ou la machine sur laquelle il s'exécute. Une autre approche consiste à combiner les deux options et ainsi adapter un programme à une architecture matérielle plus performante. C'est pour répondre à une demande de plus en plus pressante qu'est apparu le parallélisme[Noe07].

Dans la plus parts des cas le calcul parallèle est considéré comme un moyen d'augmanter la performance (réduire le temps d'exécution) des applications qui nécessitent une grande quantité de calcul.

Le parallélisme de métaheuristiques est la technique généralement utilisée pour traiter les grandes instances des problèmes difficiles ; le but était d'utiliser le calcul parallèle pour réduire le temps d'exucution afin de traiter des grandes instance des problèmes difficils, mais on va voire que l'apparition des modèles basés sur la notion de multi-cherche coopérative a conduit non seulement à réduire le temps de calcul mais aussi à trouver des solutions de qualité supérieure.

Dans ce chapitre on va donner un aperçu sur les stratégies et les modèles du parallélisme des métaheuristiques.

2.2 Le parallélisme

On peut définir le ***calcul parallèle*** comme l'utilisation simultanée des ressources multiples de calcul (un ordinateur simple avec les processeurs multiples, un nombre arbitraire d'ordinateurs s'est relié par un réseau ou nombre arbitraire d'ordinateurs s'est relié par un réseau) pour résoudre un problème informatique, et on peut imaginer un programme qui s'exécute à l'aide des unités centrales de traitement multiples. Un problème décomposé en parties discrètes qui peuvent être résolues concurremment chaque partie est une série d'instructions. Ces instructions s'exécutent simultanément sur différentes unités centrales de traitement.

2.2.1 Motivations

- La vitesse des processeurs ne continuera pas augmenter à cause des contraintes physiques (vitesse de transfert des donneés ne peut pas depasse la vitesse de la lumiere).
- Le calcul parallèle peut être :
    - la seule solution pour traiter certains gros calculs en un temps donné.
    - la solution la plus simple ou la moins onéreuse pour traiter certains calculs en un temps donné.
- Le calcul parallèle apporte une solution pour des problèmes nécessitant une haute tolérance aux fautes.
- L'univers est hautement parallèle.

2.2.2 Modèles de programmations parallèles

Il y a plusieurs modèles de programmation parallèle d'usage courant citant :

- Le modèle de Données parallèles.
- Le modèle de programmation a mémoire partagée.
- Le modèle de passage de message.

2.2.2.1 Parallélisme De Données

Ce modèle de parallélisme est utilisé pour l'exploitation de la simultanéité qui se produite de l'application du même flux d'instructions aux différents éléments d'une structure de données. Par exemple, l'addition de 2 à tous les éléments d'une rangée, ou l'incrémentation du salaire de tous les employés avec 5 ans de service.

Les programmes de parallélisme de données sont caractérisés par les caractéristiques suivantes :

- chaque opération sur chaque élément d'informations peut être considérée comme une tâche indépendante.
- La granularité normale d'un calcul donnée-parallèle est petite.
- Le concept de localité des données ne manifeste pas naturellement.

Noté que les compilateurs de parallélisme de données exigent souvent du programmeur de fournir des informations au sujet de la façon dont les données doivent être

réparties sur les processeurs, c.-à-d. de la façon dont les données doivent être divisées sur les tâches. Après le compilateur peut traduire le programme de données parallèles sous forme de SIMD, et produire un code de communication automatiquement.

2.2.2.2 **Mémoire Partagée**

Dans le modèle de programmation a Mémoire Partagée les tâches partagent un espace d'adressage commun. Chaque tache (processus) peut exécuter les opérations de lecture ou d'écriture d'une manière asynchrone. Le control d'accès à la mémoire partagée est assuré avec les mécanismes d'exclusion mutuelle tels que les verrous et les sémaphores.

L'avantage de ce modèle est que pour le programmeur il n'y a aucun besoin d'indiquer explicitement comment faire communiquer les données des producteurs aux consommateurs, ceci est à cause de l'absence de la notion de l'ownership de données. Cependant la compréhension et le maintien de la localité des données est plus difficile, par conséquent il est plus difficile d'écrire des programmes déterministes.

Dans un environnement avec mémoire distribuée on trouve une autre variété c'est le modèle de la mémoire virtuelle partagée (*Virtual Shared Memory VSM*).

La mémoire distribuée partagée (DSM) est une extension du modèle de programmation à mémoire partagée sur les systèmes qu'ils n'ont pas une mémoire physiquement partagée. L'accès se fait à l'aide des opérations de lecture et écriture habituelles.

Contrairement au passage de message, dans un système de DSM un processus qui veut effectuer des opérations sur quelques données n'a pas besoin de connaître son endroit ; le système les cherchera et trouvera automatiquement.

Dans la plupart des systèmes de DSM, des données partagées peuvent être repliées pour augmenter le parallélisme et l'efficacité des applications.

Tandis que les machines parallèles scalable sont la plupart du temps basées sur la mémoire distribuée, beaucoup d'utilisateurs trouvent que plus facile d'écrire des programmes parallèles en utilisant un modèle de programmation de shared-mémoire. Ceci fait à **DSM** un modèle très prometteur, s'il est mis en application avec efficacité[Fos95].

2.2.2.3 **Passage de message**

Le modèle de programmation parallèle passage de message est probablement le modèle aujourd'hui le plus répandu de programmation parallèle.

Les programmes Message-passing créent des tâches multiples, chaque tâche encapsule ses données locales et chaque tâche est identifiée par un nom unique, et les tâches agissent l'un sur l'autre en envoyant et en recevant des messages à et des tâches appelées.

Théoriquement ce modèle permet la création dynamique des tâches, l'exécution des plusieurs tâches par un processeur, ou l'exécution de différents programmes par différentes tâches. Cependant, dans la pratique la plupart des systèmes passage de message créent un nombre fixe de tâches identiques au démarrage de programme et ne permettent pas aux tâches d'être créées ou détruites pendant l'exécution du programme, P2PMPI est un exemple typique.

On dit que ces systèmes mettent en application un modèle de programmation programme simple données multiples (SPMD) parce que chaque tâche exécute le même programme sur des données différentes. Ce modèle est suffisant pour un éventail des problèmes de programmation parallèle mais gêne quelques développements d'algorithme parallèles.

De point de vu programmation ce modèle immerge ce forme des bibliothèques des sous programmes. Ces bibliothèques fournissent des routines pour l'initialisation et la configuration de l'environnement de transmission de messages aussi bien que l'envoi et la réception des paquets des données [Fos95].

Actuellement, deux bibliothèques de passage de message de haut niveau sont les plus populaires pour les applications scientifiques et industrielles :

- Les PVM (machine virtuelle parallèle) du laboratoire national d'Oak Ridge[WD96].
- MPI (message passant l'interface) définis par le forum de MPI[BDJM94].

2.3 Le parallélisme des métaheuristiques

Le parallélisme des métaheuristiques est l'une des deux techniques utilisées pour augmenter l'efficacité des métaheuristiques, une idée simple est d'utiliser le calcul parallèle pour exécuter les instructions d'un algorithme d'une métaheuristique d'une manière parallèle

(parallélisme bas niveau) ou décomposer l'espace de recherche en parties et en affecter chacune à un processus de recherche (approche par décomposition de l'espace de recherche).

2.4 Classification sur le parallélisme des métaheuristiques

Il existe plusieurs stratégies pour paralléliser les métaheuristiques ; et plusieurs classifications des métaheuristiques parallèles, chaque classification décompose les métaheuristiques parallèles selon une vision indépendante ; par exemple dans [CT03]carnic et al décomposent les métaheuristiques parallèle en 3 types : parallélisme de bas niveau, parallélisme par décomposition de l'espace de recherche et parallélisme haut niveau ; on remarque que cette classification engendre des classes non exclusives, on parle de type 1et2 et le type 2 et 3 car le critère de cette classification n'est pas unique ; par conséquent on trouve la décomposition de l'espace et le parallélisme bas niveau qui appartiennent à la même classe dans une autre classification [CMRR02]cette dernière décompose les métaheuristiques parallèles selon les modèles suivants : parallélisme avec trajectoire unique et parallélisme avec trajectoire multiple ; celle-ci est décomposée en deux : parallélisme avec trajectoire multiple indépendant et parallélisme avec trajectoire multiple coopérative ; cette classification est mieux adaptée au métaheuristiques basées solution unique que les métaheuristiques basées population, car même la version séquentielle de ces dernières ne définit pas une sorte de trajectoire. Des classifications ont été expliquées dans [AT02] et [Can98]pour les modèles parallèles des métaheuristiques évolutionnaires ; on trouve le modèle maitre-travailleur, le modèle insulaire et le modèle cellulaire.

Dans la suite on va donner une classification rassemblant les différents points de vue. En premier lieu on maintient le typage de Crainic[CT03] et [CT98],en second lieu, au sein de chaque type on citera les modèles de parallélisme selon la nature de la métaheuristique :

- **le niveau du parallélisme :** on trouve une classification basée sur un tel critère dans [CT03] et [CT98] , ce point de vue décompose le parallélisme des HM en deux types :
    - **parallélisme bas niveau** : dans ce type le parallélisme se fait au niveau des opérations de chaque itération, le seul but de cette stratégie est de faire accélérer le calcul, les solutions obtenues ont la même qualité des solutions obtenues par les algorithmes séquentiels.
    - **parallélisme haut niveau** : cette stratégie vise une bonne exploration de l'espace de recherche par le lancement de plusieurs processus explorent l'espace de recherche, d'une manière synchrone ou asynchrone, coopérative ou indépendante.

- **la nature de la métaheuristique de base :** c'est-à-dire si la méthode basée population ou solution unique on trouve une classification basée sur ce critère dans [RPE01].

L'application de ces deux repères donne un ensemble des modèles de parallélisme appliqués sur les métaheuristiques.

2.4.1 Parallélisme type I : parallélisme bas niveau

2.4.1.1 Le modèle d'accélération :

Les méthodes de recherche locale sont généralement construites d'un ensemble d'instructions exécutées itérativement. Le modèle d'accélération repose sur un paradigme maître-travailleur ; le processus maître décompose l'ensemble d'instructions en parties qui peuvent être exécutées simultanément, et il envoie chaque partie à un processus esclave ; puis chaque processus esclave exécutera la partie qui lui est assignée indépendamment sur des contraintes de synchronisation entre-parties [CT98].

Ce modèle peut être appliqué aussi si l'évaluation de la fonction de cout est parallélisable et/ou les entrées sorties sont coûteuses en temps. La fonction de cout sera remplacée par des fonctions des couts partielles évaluées simultanément ; la fonction globale atteinte par l'application d'une fonction d'agrégation[CT98].

2.4.1.2 Le modèle d'évaluation parallèle du voisinage

Le voisinage d'une solution est l'ensemble des solutions atteignables à partir de celle-là après l'application d'un mouvement. L'étape d'évaluation des solutions voisines est une étape pertinente dans les méthodes basées solution unique, et peut consommer la plus part de temps d'exécution, donc nous reposons sur un modèle *maître- travailleurs* pour établir une évaluation parallèle d'un tel voisinage. Le processus maître engendre l'ensemble des solutions voisines de la solution courante et envoie chaque partie à un processus travailleur, qui à son tour fait les calculs pour évaluer la partie qui lui est associée, après il renvoie le résultat au processus maître ; ce dernier compare les résultats reçus et choisit le meilleur (voire la figure 10) [Mal05].

Ce modèle exploite la puissance de calcul parallèle mieux que le modèle précédent, surtout si la fonction à évaluer était coûteuse en termes de calcul [Mal05].

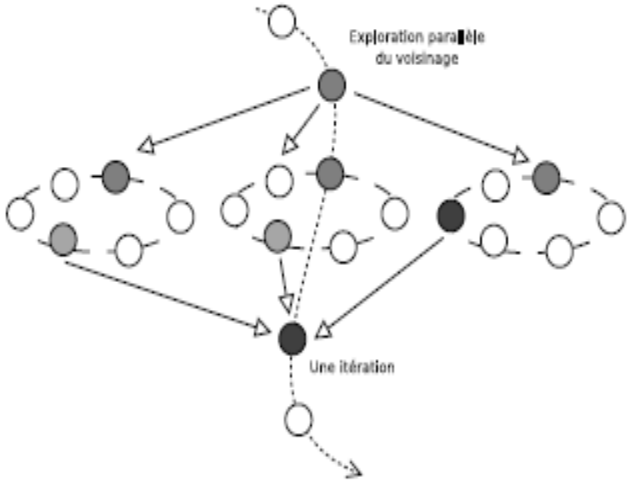

Figure 10 : le modèle d'évaluation parallèle de voisinage.

2.4.1.3 Le modèle d'évaluation parallèle d'une population

Dans les algorithmes génétiques, un processus maître exécute l'opérateur de la sélection, le croisement et la mutation sur l'ensemble de la population ; après il envoie une partie de la population aux processus travailleurs, à leur tour, ces derniers évaluent les parties qui leur sont affectées , puis ils renvoient les résultats au maître, celui-ci calcule la moyenne totale ; car l'ordre employé pour calculer la fitness (ou la qualité) des individus est non pertinent à la moyenne finale de la population. La figure (11) montre la fonctionnalité de ce modèle.

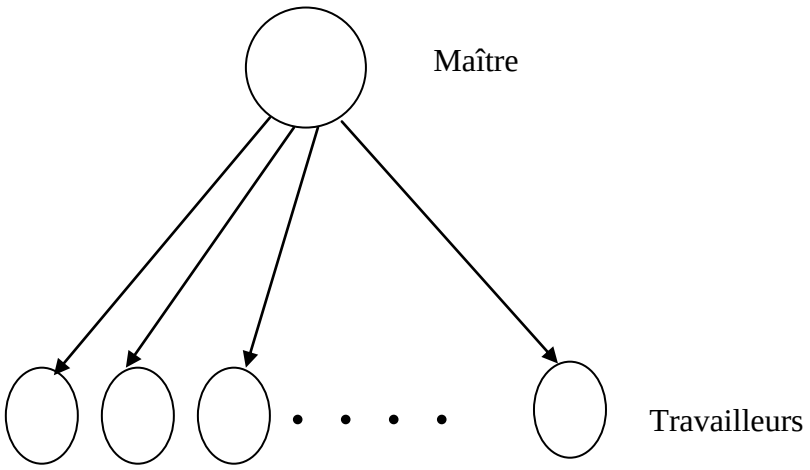

Figure 11 : le modèle d'évaluation parallèle d'une population.

2.4.1.4 Le modèle d'évaluation parallèle d'une fonction objectif

C'est un autre modèle basé sur le paradigme maitre-travailleurs, appelé aussi modèle d'évaluation parallèle d'un individu [Mal05]. Chaque individu est répliqué sur plusieurs machines et il reçoit une évaluation partielle de leur fonction objective, puis les parties calculées de la fonction objective sont envoyées à la machine maîtresse pour appliquer une fonction d'agrégation. Ce modèle de parallélisme sera utile si et seulement si la fonction objective est très coûteuse en fonction de calcul. Maleb [Mal05] dit que l'utilisation de ce modèle avec d'autres améliore le degré de parallélisme d'une manière significative ; il dit aussi que pour un modèle insulaire à 10 îles de 100 individus, la prise en compte du modèle d'évaluation parallèle d'une solution fait passer le nombre d'évaluation pouvant être effectuées simultanément de 1000 à 20000.

2.4.2 Parallélisme type II : avec décomposition de l'espace de recherche

Dans ce type, des sous problèmes sont résolus en parallèle, et la solution finale sera obtenue par une technique de composition ou d'extraction. La mise en application de ce type de parallélisme est généralement assurée par un modèle maître-esclave où un processus - maître divise l'espace de recherche avec la décomposition de l'ensemble des variables de décision en sous-ensembles disjoints ; les variables en dehors de chaque sous-ensemble sont considérées fixes, ensuite le maître envoie chaque partie à un processus ***esclave,***ce dernier applique l'heuristique à leur partition assignée concurremment et indépendamment puis il renvoie le résultat au maître qui composera le résultat final.

Le maître peut effectuer des modifications sur les partitions pendant la recherche, soit à des intervalles fixés en avant ou déterminés pendant l'exécution ; ou au moment de redémarrage de processus de la recherche.

Un inconvénient d'une telle approche basée sur la division des variables de décision peut laisser des grandes parties de l'espace de recherche inconnues. Par conséquent la plupart des applications répètent la division pour créer des différents segments du vecteur de variable de décision et redémarre la recherche [CT03].

2.4.3 Le parallélisme type III : parallélisme haut niveau

2.4.3.1 Le modèle multi cherche

Il consiste à lancer simultanément plusieurs processus de recherche locale pour trouver une meilleure et robuste solution. Les processus de recherche peuvent exécuter la même méthode heuristique, dans ce cas on a une approche dite *homogène ;* comme ils peuvent aussi

exécuter des heuristiques différentes, en ce moment, on a une approche dite hétérogène. Ainsi ils peuvent commencer à partir de la même solution initiale (*mono dép*art) ou à partir des différentes solutions (multi *départs*) ; et peuvent communiquer seulement à l'extrémité de la recherche pour identifier la meilleure solution globale, ainsi on l'appellera méthode *de recherche indépendante.* Comme ils peuvent inter-changer des informations pendant la recherche, on l'appellera *méthode de recherche coopérative*. Les communications peuvent être exécutées d'une manière synchrone ou asynchroneet peuvent être déclenchées par des événements ou exécutées aux moments prédéterminés ou dynamiquement décidés [CT03].

Ce modèle exige plus d'efforts de programmation et des maîtrises pour l'implémentation. Les processus échangent l'information rassemblée le long de la recherche, Cette information partagée est mise en application en tant que : variables globales stockées dans une mémoire partagée, ou par passage de message si on utilise une architecture de mémoire distribuée.

Dans ce modèle, où les processus de recherche coopèrent ; l'information rassemblée le long de chaque trajectoire est employée pour améliorer les autres. Cependant on compte non seulement à accélérer la convergence vers une solution optimale mais, aussi, à trouver une meilleure solution que la solution trouvée par les stratégies de recherche indépendantes avec le même temps de calcul.

L'aspect le plus difficile est la détermination de la nature d'information à partager ou à être échangée pour améliorer la recherche, sans prendre trop de mémoire ni de temps additionnel pour rassembler cette information. Généralement on utilise les meilleures solutions trouvées, des mouvements, des listes de tabou, et tailles de population, entre d'autres. Ces informations peuvent donner une vue plus générale sur l'espace de recherche et peuvent être employées pour renforcer les stratégies de diversification et d'intensification [CMRR02].

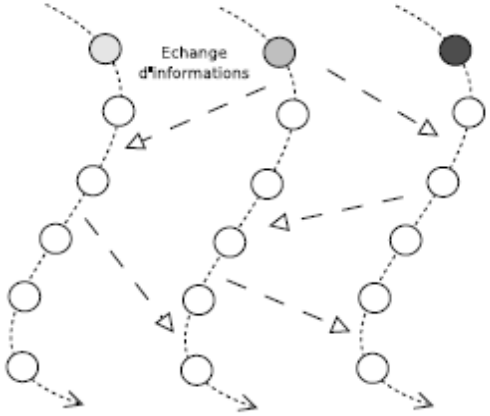

Figure 12 : le modèle multi-cherche.

Le calcul est accéléré avec l'emploi d'une stratégie à recherche multiple, on essaye généralement de réduire l'espace de recherche pour chaque processus par rapport à l'espace de recherche exploré par un algorithme séquentiel. La mise en application d'une telle technique varie selon la méthode métaheuristique. Si on a Ple nombre de processeurs ; pour la recherche de tabou par exemple, on peut effectuer à chaque processus T/P itérations, où T est le nombre d'itérations du processus séquentiel correspondant. Pour le recuit simulé, le nombre d'itérations de la boucle d'équilibrage est réduit proportionnellement de L à L/P [CT03].

2.4.3.2 Le modèle d'îles (Island model)

Ce modèle est appelé aussi le modèle parallèle des algorithmes génétiques distribués ou le modèle parallèle avec une granularité grossière [Can98].

Le modèle Island consiste à diviser la population globale en sous populations, où chaque sous population est affectée à un processeur. Les opérations de sélection, croisement et mutation sont faites au niveau de chaque sous population. La communication entre les sous population est assurée par la migration des individus entre elles (voire la figure) ; sans celle-ci les sous populations convergent rapidement vers des solutions non optimales.

Une bonne stratégie de migration peut donner des résultats comparatifs ; la migration est contrôlée par plusieurs paramètres tel que :

- la Structure de voisinage : détermine la topologie et la densité de la connexion inter sous population. Si la densité de la connexion est très forte l'ensemble se

comportera comme un algorithme unique ; et si la connexion est faible chaque sous population se comportera comme une population indépendante.

- le choix des individus à être migrés : définit la nature des individus à migrer ils peuvent être les meilleurs, les mauvais ou aléatoires.
- le temps de migration : définit quand les individus doivent être immigrés ; la migration inter populations est soit synchrone est déterminée à l'avance avec des intervalles fixes, ou asynchrone.

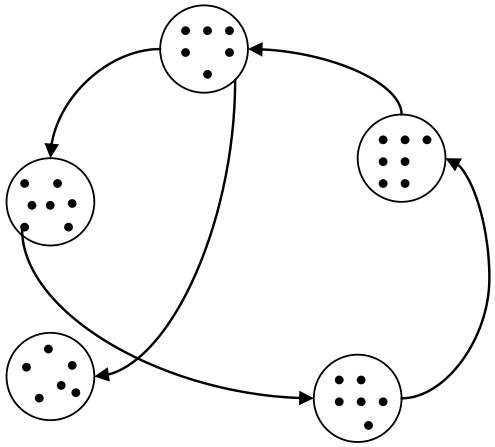

Figure 13 : le modèle d'îles.

2.4.3.3 Le modèle cellulaire

Dans ce modèle la population est divisée en sous populations où chacune contient un nombre très réduit d'individus. Chaque ensemble est affecté à un processeur, qui habituellement manipule un ou un nombre réduit d'individus avec une communication intensive entre les ensembles. On note que les ensembles sont rangés selon une topologie bien déterminée de 1, 2 ou 3 dimensions, La topologie 2D rectangulaire (la figure 14) est la plus utilisée parce que dans la plupart des ordinateurs massivement parallèles, les processus sont reliés selon cette topologie.

Chaque ensemble a un voisinage, dans ce modèle, la notion de voisinage est pertinente car les opérations de sélection, croisement et mutation sont faites localement.

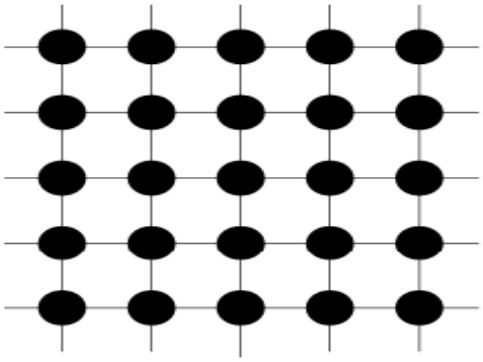

Figure 14 : le modèle cellulaire.

Ce modèle est mieux adapté à une structure informatique massivement parallèle SIMD, comme la machine Maspar200. En outre ce modèle peut être mis en application sur n'importe quel multiprocesseur. Ses paramètres principaux sont : la taille de population, la structure de population, la structure et la taille de voisinage [Can98].

2.5 Conclusion

Lorsque la taille des données d'entrée d'un problème difficile augmente, la taille de l'espace de recherche augmente aussi, dans ce cas-là l'utilisation des métaheuristiques dans leurs versions originales ne donne pas les résultats souhaités, pour augmenter la puissance de recherche des métaheuristiques le parallélisme est une technique qui est généralement utilisée.

Le parallélisme consiste à exécuter plusieurs instructions simultanément sur une ou plusieurs machine(s) capable (s) de faire ça. Le parallélisme des métaheuristiques est réalisé sur plusieurs manières qui engendrent par la suite une classification de différents types :

- le parallélisme bas niveau : basé généralement sur un modèle maitre-travailleur et tente toujours de réduire le temps d'exécution occupé par le processus de recherche, afin d'élaborer des grandes instances des problèmes difficiles, les solutions données par ce type de parallélisme sont les même données par les versions séquentielles mais avec un temps inferieur.
- Le parallélisme avec la décomposition de l'espace de recherche : basé aussi sur le modèle maitre-travailleur, il nécessite un effort supplémentaire pour avoir les résultats finaux.
- Le parallélisme haut niveau : consiste à lancer plusieurs processus de recherche qui tentent d'explorer la totalité de l'espace de recherche. Dans ce type de parallélisme on ne cherche pas seulement de réduire le temps d'exécution mais aussi de trouver des

solutions plus efficaces et robustes. Ce type comprend une grande variété des méthodes et il nécessite plus d'efforts soit dans la conception (choix et adaptation des paramètres), soit dans l'implémentation (synchronisation, modes de migration …). Ce type de parallélisme reste le plus promoteur pour donne+r des bons résultats.

Chapitre III

recherche locale a mémoire adaptative parallèle pour le problème de voyageur de commerce