Methode Tabou massivement parallele pour le probleme d'affectation quadratique

N. August, Thierry Mautor

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Méthode Tabou massivement parallèle pour le problème d'affectation quadratique

Nicolas AUGUST
Thierry MAUTOR

N° 2182
Décembre 1993

PROGRAMME 1

Architectures parallèles,
bases de données,
réseaux et systèmes distribués

Méthode Tabou massivement parallèle pour le problème d'affectation quadratique

---

Massively parallel Tabu search for the quadratic assignment problem

Nicolas AUGUST et Thierry MAUTOR

Programme 1 - Décembre 1993

Laboratoire PRiSM
Université de Versailles - St Quentin en Yvelines
45, Avenue des Etats Unis, 78000 Versailles

INRIA - Action Paradis
Domaine de Voluceau - BP 105 Rocquencourt
78153 Le Chesnay Cédex

RESUME

Nous étudions, dans cet article, l'adaptation de la méthode Tabou à la résolution du problème d'affectation quadratique. Dans un premier temps, nous expérimentons, en séquentiel, différentes stratégies de recherche pour la méthode Tabou et parvenons ainsi à obtenir des résultats supérieurs à ceux de tous les précédents algorithmes Tabou séquentiels. L'adaptation de notre algorithme à un environnement massivement parallèle est effectuée ensuite sur une Connection Machine CM-2. Des résultats de très bonne qualité, toujours à moins de 0.7% de la meilleure solution connue, sont ainsi obtenus en un faible nombre d'itérations.

MOTS-CLES : Problème d'affectation quadratique, Méthode Tabou, Algorithme massivement parallèle.

ABSTRACT

In this work, we study the adaptation of Tabu search to the solution of the Quadratic Assignment Problem. First, we test, in a sequential way, different tabu search strategies. This way, we achieve better results than all previous sequential Tabu algorithms. Next, we extend this program to a massively parallel environment, on a Connection Machine CM-2. Good quality results, always less than 0.7% far from best known results, are obtained with a small number of iterations.

KEY-WORDS : Quadratic Assignment Problem, Tabu Search, Massively parallel algorithm.

1 Introduction

Le problème d'affectation quadratique (P.A.Q.) a été formulé pour la première fois, il y a maintenant plus de trente ans et s'est révélé depuis lors comme ayant de très nombreuses applications pratiques, les plus intéressantes étant le placement de circuits électroniques et la répartition des services médicaux dans les grands centres hospitaliers.

C'est pourquoi il a logiquement suscité de nombreux et réguliers travaux au fil de ces trente années.

Cependant, ce problème s'est alors avéré particulièrement coriace à résoudre. La meilleure illustration de ce fait est, qu'à l'heure actuelle encore, les méthodes exactes les plus performantes ne permettent pas de résoudre la plupart des problèmes de taille supérieure à 20. Or des applications pratiques comme celle du placement des circuits électroniques, nécessitent souvent de placer plus d'une centaine d'éléments.

C'est la raison pour laquelle de nombreuses méthodes approchées ont été adaptées à la résolution du problème d'affectation quadratique. Parmi ces adaptations, les méta-heuristiques récentes que sont les algorithmes génétiques, le recuit simulé et la méthode Tabou se montrent les plus efficaces.

Plus précisément encore, c'est, à l'heure actuelle, la méthode Tabou qui permet d'obtenir les meilleurs résultats. C'est pourquoi nous nous sommes intéressés aux conditions d'adaptation de la méthode Tabou à la résolution du problème d'affectation quadratique de manière séquentielle tout d'abord puis dans un environnement massivement parallèle, sur une *Connection Machine*.

Nous présentons tout d'abord, dans le chapitre 2, le problème d'affectation quadratique, ses particularités, ses applications les plus importantes et ses principales méthodes de résolution aussi bien exactes qu'approchées.

Le chapitre 3 est ensuite l'occasion d'une description plus détaillée de la méthode Tabou. Nous nous intéressons plus particulièrement dans le chapitre 4 aux conditions d'application de cette méthode à la résolution du problème d'affectation quadratique et présentons plusieurs expérimentations que nous avons menées dans ce but.

Le chapitre 5, enfin, nous permet d'exposer les principes de notre implémentation massivement parallèle ainsi que les résultats obtenus.

2 Le problème d'affectation quadratique

2.1 Présentation du problème

Le problème d'affectation quadratique consiste en la détermination de la meilleure implantation de n unités (usines, services hospitaliers ...) sur n sites connus. Il s'agit de trouver quelle implantation des unités sur les sites permettra de minimiser le coût total de transit, le coût d'un transit étant supposé égal au produit de la distance entre les sites multiplié par le flux de matière transportée.

Par ailleurs, il est bien sur nécessaire d'implanter une et une seule unité sur chaque site.

Soient
- $D = (d_{ij})$, la matrice des distances entre les sites,
- $F = (f_{kl})$, celle des flux entre les unités,
- Π l'ensemble des permutations de $\{1 \ldots n\}$ dans lui-même.

Le problème peut alors s'écrire sous la forme suivante :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} d_{ij} f_{kl} x_{ik} x_{jl}$$

sous les contraintes:

$$x_{ik} \in \{0, 1\}$$

$$\forall k \in \{1 \ldots n\}, \sum_{i=1}^{n} x_{ik} = 1$$

$$\forall i \in \{1 \ldots n\}, \sum_{k=1}^{n} x_{ik} = 1$$

avec :
$x_{ik} = 1$ si l'unité k est placée sur le site i,
$x_{ik} = 0$ sinon.

Nous pouvons également définir ce problème de la manière suivante à l'aide des permutations (voir par exemple [16]):

$$\min_{\pi \in \Pi} C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f_{\pi(i)\pi(j)}.$$

2.2 Principales applications

Le problème d'Affectation Quadratique à été défini pour la première fois en 1957 par Koopmans et Beckmann [11], dans un contexte économique, afin de déterminer la meilleure implantation d'usines à des sites pré-déterminés.

Il a été depuis lors appliqué à de très nombreux cas concrets tels que :

- L'électronique, avec le placement de circuits sur une plaquette [20]. Dans le cas des circuits VLSI, on obtient des problèmes de très grande taille, mais peu denses, un circuit n'étant relié qu'à un petit nombre d'autres circuits.
- L'architecture, avec la répartition des services hospitaliers (David et Roucairol - Kremlin Bicêtre (1973) [5], Elshafei - Hôpital du Caire (1977) [6]).
- D'autres types d'application très variés comme l'ergonomie, la planification ou l'analyse des réactions chimiques.

Le problème du voyageur de commerce est un cas particulier du problème d'affectation quadratique.

La matrice des distances est celle des distances entre les villes, et la matrice de flux modèlise un circuit hamiltonien (passant une fois et une seule par toutes les villes):

$$F = \begin{pmatrix} 0 & 1 & 0 & & & & \\ 0 & 0 & 1 & & & 0 & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & 0 & & & & \ddots & \\ & & & & & & 1 \\ 1 & 0 & 0 & \cdots & & & 0 \end{pmatrix}$$

Cette propriété permet de montrer l'appartenance du problème d'affectation quadratique à la classe des problèmes NP-complets.

Par ailleurs, il a également été démontré que la détermination d' une solution "ϵ-approchée" (dont l'erreur relative reste inférieure à ϵ pour n'importe quel problème), est aussi un problème NP-difficile [17].

2.3 Les méthodes de résolution

Résoudre un problème NP-difficile comme le problème d'affectation quadratique peut se faire de deux façons différentes :

- soit en cherchant la meilleure solution possible, ce qui oblige à prouver que la solution obtenue est bien optimale,
- soit en cherchant à obtenir une "bonne" solution, dont rien ne prouve qu'elle est la meilleure possible.

La première manière de procéder est celle des méthodes dites exactes. Elle répond exactement au problème posé mais, malheureusement, ne permet le plus souvent que de résoudre des problèmes de petite taille.

La seconde voie est donc très utile pour pouvoir aborder des problèmes de taille plus importante.

Nous allons examiner les différentes méthodes de résolution utilisées pour le P.A.Q.

2.3.1 Les méthodes exactes

Les méthodes permettant une résolution exacte du problème d'affectation quadratique sont des méthodes d'énumération implicite.

Les méthodes par linéarisation formalisent le problème comme un programme linéaire en variables 0-1, puis utilisent des méthodes de troncature ([1]). Si elles se sont montrées efficaces pour un certain nombre de problèmes d'optimisation combinatoire comme celui du voyageur de commerce, ce n'a pas été le cas pour le problème d'affectation quadratique. Elles ne sont d'ailleurs pratiquement plus utilisées pour la résolution de ce problème.

Les méthodes Branch & Bound se sont révélées les plus efficaces en ce qui concerne la résolution exacte du P.A.Q.

En particulier, l'algorithme de T. Mautor et C. Roucairol [13] a permis de résoudre de manière exacte l'ensemble des problèmes classiques de la littérature de taille inférieure à 20, dont certains pour la première fois.

2.3.2 Les méthodes approchées

Nous ne citons ici que les plus efficaces d'entre elles. Nous omettons en particulier de nombreuses méthodes constructives aux résultats très moyens.

- **Les procédures de descente** partent d'une solution initiale qui est progressivement améliorée par des échanges de placement des unités. Les échanges les plus couramment utilisés sont le 2-échange qui permute les placements de deux unités (si l'unité i était affectée au site j et l'unité k au site l, après l'échange on obtient, i affectée à l et k à j), et le 3-échange qui effectue la même opération avec 3 unités.

  Le principe de la méthode est d'effectuer à chaque étape l'échange qui améliore le plus la fonction de coût et de s'arrêter lorsqu'aucun échange ne permet plus d'amélioration.

  Si on note N_s le voisinage d'une solution s, qui correspond à l'ensemble des solutions atteignables à partir de s grâce à un mouvement admissible (comme le 2-échange), l'algorithme est le suivant :

  1. *Choisir une solution initiale s.*

  2. *Rechercher* $s' \in N_s$ *telle que* $f(s') = \min_{x \in N_s} f(x)$.

  3. *Si* $f(s') \geq f(s)$ *alors FIN .*

  (plus d'amélioration : nous sommes sur un optimum local).

  sinon $s := s'$ *et revenir en 2. .*

  L'inconvénient de ce type de méthodes est le fait d'atteindre un optimum local, dont la qualité est incertaine, et de ne pas pouvoir "en sortir".

  C'est pourquoi ont été mises au point ces dernières années un certain nombre de métaheuristiques essayant de dépasser ce bloquage.

- **Le recuit simulé** est directement inspiré de certaines méthodes utilisées en métallurgie :

  Suivant la façon dont on abaisse la température d'un métal en fusion on obtient des structures finales différentes. Si on la baisse brutalement (technique de la "trempe"), on fige la structure de départ en restant du point de vue énergétique à un *minimum local* : on obtient un verre.

Si par contre on baisse la température très lentement en laissant aux atomes le temps de se réorganiser, on atteint l'état du cristal qui correspond au *minimum absolu* de l'énergie interne. C'est la méthode dite du recuit.

L'algorithme du recuit simulé a été mis au point à partir de cette idée, le but étant de parvenir à l'optimum global en surmontant l'optimum local.

C'est un algorithme stochastique permettant une déterioration de la valeur courante suivant une certaine probabilité dépendant de l'importance de la détérioration entre la nouvelle solution et l'ancienne, et du degré d'avancement de la recherche.

En pratique, l'algorithme est le suivant:

Choisir une solution initiale s
Choisir une température initiale T.
Tant que *le système n'est pas gelé* faire
  Tant que *l'équilibre à T n'est pas atteint* faire
    Choisir aléatoirement un mouvement élémentaire applicable à partir de s amenant à la solution s'
    Calculer Δ : la variation de cout engendrée par l'application de ce mouvement
    Si $\Delta \leq 0$ (amélioration de la solution)
      alors s := s' (modification acceptée)
      sinon (dégradation de la solution)
      faire s := s' avec la probabilité $exp^{-\frac{\Delta}{T}}$
    fin Si
  fin Tant que
  Réduire la température
Fin Tant que

Les algorithmes génétiques sont fondés sur le croisement itératif d'une population de solutions. A chaque itération les solutions sont évaluées et les meilleures, en terme de coût sont croisées afin d'obtenir une nouvelle population. L'idée de cette méthode est que les meilleurs "génes" (affectations dans le cas du P.A.Q.) vont finir par s'imposer, et que la valeur moyenne des solutions va ainsi progressivement s'améliorer.

La méthode TABOU est une méta-heuristique mise au point par Glover en 1986 et de manière indépendante par Hansen, la même année (voir [8], [9] pour une description complète de la méthode).

Son objectif est de surmonter le problème des optima locaux tout en évitant au maximum de cycler. Pour cela on effectue à chaque étape **le meilleur déplacement possible**, même si celui-ci conduit à une solution moins bonne que celle trouvée précedemment. De plus, afin d'éviter les cycles, les n derniers mouvements sont interdits ou tabous d'où le nom de cette méthode.

Nous allons maintenant l'étudier plus en détail, ainsi que son adaptation au problème d'affectation quadratique.

3 La méthode Tabou

Cette méthode a été utilisée pour des applications très diverses telles que le problème du voyageur de commerce, l'affectation quadratique, les ensembles stables de cardinal maximal, la coloration ou la partition de graphes ainsi que des problèmes de télécommunication ou d'emploi du temps.

3.1 Principes de la méthode

3.1.1 Description générale

A chaque itération, tous les mouvements possibles sont examinés et le "meilleur", ou plus exactement le moins mauvais est sélectionné.

Comme cette manière d'agir peut conduire à cycler c'est à dire à répéter indéfiniment la même suite de mouvements, les t derniers mouvements effectués sont considérés comme interdits ("tabous"), t représentant la taille de la liste Tabou. Le mouvement effectivement choisi est donc le meilleur mouvement non tabou.

L'algorithme Tabou simple (séquentiel) est donc le suivant:

1- *Choisir une solution initiale s*
2- Initialisations :
   s^* := s *(s^* représente la meilleure solution obtenue jusque là)*
   k := 0 *(k est le compteur d'itérations effectuées depuis la dernière amélioration de s^*)*
   T := $\emptyset$ *(T est la liste Tabou)*
3- **Tant que** *le critère d'arret n' est pas vérifié* faire
   Si $N_s - T \neq \emptyset$ **alors** *(il existe au moins un mouvement effectuable à partir de s et non Tabou)*
      k := k+1
      Rechercher s' $\in$ (N_s-T) telle que s' = $\min_{x \in N_s - T} f(x)$
      s := s'
      Si $f(s) < f(s^*)$ alors s^* := s
                                k := 0
      fin Si
   fin Si
   Mettre à jour la liste T
Fin Tant que

3.1.2 Choix des paramètres

Le choix de la solution initiale n'a pas une grande influence sur l'efficacité de l'algorithme, comme l'a montré Skorin-Kapov [19].

Par contre, la taille de la liste Tabou ainsi que le critère d'arrêt ont une grande importance. En particulier, la taille de la liste Tabou apparaît comme un facteur primordial. Si celle-ci est trop petite, elle ne permet pas d'éviter les cycles alors que si elle est trop grande on risque de manquer très fréquemment des mouvements intéressants. Le critère d'arrêt est, lui, généralement fonction du nombre d'itérations k effectuées depuis la dernière amélioration de la solution.

Il parait toutefois impossible de déterminer a priori les valeurs de ces paramètres. Ce réglage ne peut s'effectuer que de manière empirique nécessitant donc plusieurs expérimentations. Notons que, d'après Taillard [21], il semble plus efficace de faire varier la taille de la liste Tabou au cours de la recherche.

3.1.3 Mouvements possibles

Il semble préférable que les mouvements effectués fassent passer d'une solution à une autre. Le fait de sortir de l'ensemble des solutions, c'est à dire de ne pas respecter les contraintes du P.A.Q., ne permet pas d'améliorer la recherche et diminue même fortement les performances malgré les perspectives de diversification (voir en particulier le mémoire de S.Riche [15]).

Le mouvement le plus simple est le 2-échange qui permute les affectations de 2 sites, ou de 2 unités. Il n'y a alors que $\frac{n*(n-1)}{2}$ mouvements possibles et le calcul du gain peut se faire de manière relativement simple (voir plus loin : 4.3.2).

Le 3-échange qui permute les affectations de 3 sites, est également envisageable, mais demande plus de calculs tant vis à vis du nombre de mouvements possibles que pour le calcul du gain occasionné par chacun de ces mouvements.

En revanche, le k-échange, avec $k > 3$, s'avère totalement rédhibitoire en termes de temps de calcul.

3.1.4 L'intensification

L'idée de l'intensification est d'approfondir la recherche dans un secteur contenant de bonnes solutions. Plusieurs méthodes peuvent être envisagées pour aller dans cette voie.

Ainsi, Chakrapani et Skorin-Kapov [4] proposent d'effectuer une intensification en repartant de la meilleure solution avec une liste TABOU vide.

Une autre possibilité consiste à fixer un seuil de fréquence et à ne prendre en compte que les mouvements au dessus de ce seuil. On retrouve un peu ici l'idée d'affectations prometteuses ou génétiquement intéressantes qu'il convient de privilégier, employées dans les algorithmes génétiques.

3.1.5 La diversification

L'objectif de la diversification, opposé à celui de l'intensification, est d'orienter la recherche vers de nouvelles régions non encore explorées et, peut-être, très intéressantes.

Là encore, plusieurs procédés peuvent être envisagés. Le plus courant consiste à utiliser une matrice des fréquences contenant la fréquence de l'affectation de l'unité i au site j.

La diversification s'effectue alors en empêchant d'utiliser les affectations les plus fréquentes.

3.1.6 Le critère d'aspiration

Il arrive qu'un mouvement Tabou, donc en principe interdit, se révèle intéressant. L'objet du critère d'aspiration est de nous permettre de l'effectuer quand même, malgré son statut Tabou. Cette idée est développée par F. Glover dans [10].

L'aspiration la plus couramment utilisée et la plus évidente consiste à regarder si le mouvement ne conduit pas à une solution de coût inférieur à celui de la meilleure solution trouvée jusqu'à présent.

D'autres critères d'aspiration, un peu plus complexes, peuvent également être envisagés. Ainsi, l'aspiration peut s'effectuer si la nouvelle solution est meilleure qu'un optimum local. L'aspiration par direction favorable a lieu, quant à elle, si le mouvement courant conduit à une amélioration, et si le mouvement précédent avait également conduit à une amélioration ($f(x_3) \leq f(x_2) \leq f(x_1)$).

L'inconvénient de recourir trop souvent à l'aspiration est qu'elle détruit la protection offerte par la liste Tabou et favorise donc le cyclage.

3.1.7 Enchaînement des diverses parties

La méthode TABOU posséde donc un grand nombre d'aspects différents. Au même titre qu'il doit régler les paramètres, l'utilisateur doit décider ce qu'il va utiliser et comment il va gérer le compromis intensification/diversification.

L'agencement des différentes phases de recherche et le choix des valeurs des paramètres doivent, de toute façon, s'effectuer en tenant compte du problème à résoudre.

3.2 Les applications de la méthode Tabou à la résolution du P.A.Q.

3.2.1 Présentation des algorithmes existants

Du fait de la nouveauté de la méthode, ses applications à la résolution du problème d'affectation quadratique sont toutes extrèmement récentes. Elles sont dues d'une part à Skorin-Kapov et d'autre part à Taillard.

- En 1990, Skorin-Kapov a proposé une première application de la recherche Tabou à l'affectation quadratique [19]. Ce programme séquentiel sans véritable intensification de la recherche, si ce n'est une possibilité de redémarrage à partir de la meilleure solution jusque là trouvée, a obtenu des résultats déjà excellents. Sur les problèmes standard de petite taille, la meilleure solution est trouvée très rapidement. La meilleure solution connue pour le problème de Steinberg de taille 36 est améliorée.

- Skorin-Kapov a proposé ensuite une version améliorée de son premier programme en y incluant une approche appelée analyse cible [18]. Celle-ci, très proche de la stratégie d'intensification de la recherche, permet classiquement

de restreindre l'espace de recherche en fixant les affectations communes aux bonnes solutions.

- Enfin, en 1991, Chakrapani et Skorin-Kapov ont développé une version massivement parallèle de l'algorithme de recherche Tabou, implémentée sur une Connection Machine CM-2, avec 16K processeurs [4].

- De son côté, Taillard a proposé, en 1991, une autre adaptation de la méthode Tabou à l'affectation quadratique, parallèle et testée sur un réseau de dix transputers, qui se montre très efficace [21]. En effet, cet algorithme obtient, pour les problèmes de taille élevée (49 à 90), des résultats encore meilleurs que ceux atteints par Skorin-Kapov dans ses deux premières implémentations et d'ordre comparable à ceux obtenus par l'algorithme massivement parallèle de celle-ci.

Toutes ces applications, et tout particulièrement les deux dernières, ont obtenu d'excellents résultats surpassant nettement ceux de toutes les heuristiques antérieures, même ceux du recuit simulé, et faisant incontestablement de Tabou la méthode approchée "leader" à cette date pour la résolution du problème d'affectation quadratique.

3.2.2 Algorithmes parallèles et massivement parallèles

Nous voudrions détailler ici les deux plus récents travaux (Taillard et Chakrapani - Skorin-Kapov) qui sont ceux qui obtiennent les meilleurs résultats et qui sont des implémentations respectivement parallèle et massivement parallèle.

- *Utilisation des processeurs :*
  Toutes les méthodes citées ci-dessus utilisent comme seuls mouvements les 2-échanges. C'est au moment du calcul de tous les différents gains occasionnés par les $\frac{n*(n-1)}{2}$ 2-échanges possibles et de la sélection du meilleur d'entre eux que le parallélisme est utilisé.

  Taillard répartit ainsi l'étude de ces échanges sur ses 10 transputers.

  Chakrapani et Skorin-Kapov utilisent un processeur par couple site/unité, soit n^2 processeurs. L'évaluation des mouvements et la sélection du meilleur peuvent être réalisés en $O(log\ n)$ opérations, permettant un speed-up de l'ordre de $O(\frac{n^2}{log\ n})$.

- *Calcul du gain :*
  Le calcul du gain est donc la partie essentielle de l'algorithme ou, tout du moins, celle qui se prête le mieux à une parallélisation éventuellement massive. C'est pourquoi nous détaillerons ultérieurement cet aspect (4.3.2 et 5.3).

- *Enchainement des différentes phases :*
  Les différentes phases sont enchainées de manière séquentielle. Ainsi, l'algorithme de Chakrapani et Skorin-Kapov effectue successivement :

  – une phase de recherche Tabou simple pendant 25n itérations,

- une phase d'intensification à partir de la meilleure solution trouvée jusque là (50n itérations),
- une phase de diversification (10n itérations).

4 Expérimentations séquentielles

4.1 Idées développées

Quelques aspects de l'algorithme Tabou ne sont pas précisés de façon explicite. Ainsi, à chaque étape on retient le meilleur mouvement non tabou, en faisant eventuellement jouer un critère d'aspiration. Mais que fait-on lorsqu'il y a *plusieurs mouvements de même gain* ?

Pour que le choix ne s'effectue pas au hasard, nous pourrions effectuer une étape de calcul supplémentaire afin de déterminer quel est celui qui conduit à la meilleure solution ultérieure.

Ceci permettrait d'avoir une certaine "vision en avant", sans pour autant être obligé de calculer systématiquement tous les mouvements sur une profondeur de 2, ce qui conduirait à des calculs excessifs.

De même, les définitions de l'intensification et de la diversification supposent la définition préalable de régions qu'il convient soit de privilégier, soit de quitter. Etant donné la nature du problème, nous pourrions partitionner l'espace des solutions en considérant par exemple les solutions pour lesquelles une unité donnée est placée sur un site particulier.

Toutefois, le fait de fixer certaines affectations n'est hélas pas très pertinent pour le P.A.Q. car, compte-tenu de la fonction économique, les zones correpondant aux diverses affectations fixées ont des valeurs moyennes très proches.

Enfin, nous avons déjà vu que le 2-échange présentait un grand nombre d'avantages surtout en ce qui concerne le calcul du gain, la taille du voisinage et le nombre d'opérations. C'est pourquoi nous utiliserons ce mouvement dans notre programme. Cependant, comme le 3-échange a donné également de bons résultats avec les procédures de descente, nous effectuerons un test comparatif sur un programme simple afin de voir ce qu'il en est concernant la méthode Tabou.

4.2 Versions séquentielles

Plusieurs versions séquentielles de Tabou ont été mises au point afin de comparer les résultats obtenus à l'aide des différentes stratégies de calcul :

- Une recherche **Tabou classique** avec une aspiration simple (aspiration si la valeur est inférieure au minimum global). Lorsque le nombre fixé d'itérations sans amélioration est atteint, on enchaîne les phases suivantes :
  - une intensification en repartant de la meilleure solution avec une liste Tabou vide,
  - une diversification en repartant d'une solution minimisant la fréquence d'affectation des sites aux unités.

- Un programme effectuant les mêmes opérations, mais qui **en cas d'égalité** entre plusieurs mouvements occasionnant le même gain minimal, sélectionne celui qui donnera **le meilleur résultat à l'itération suivante.**

- Un programme qui au lieu de se limiter à l'étude de toutes les permutations possible de 2 unités (2-échanges), examine tous les **3-échanges** possibles (plus grand voisinage).

- Un programme qui calcule les gains de tous les mouvements possibles sur 2 itérations puis effectue celle qui **minimise la somme des gains sur ces 2 itérations.**

4.3 Programme Tabou classique

4.3.1 Principe de l'algorithme

Classiquement, les deux paramètres du programme que l'utilisateur doit fixer sont la taille de la liste Tabou et le nombre d'itérations sans améliorations à l'issue duquel le programme passe à la phase suivante.

La solution de départ est simplement la permutation identité (1,2,3, ... n). Le coût initial est donc : $\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f_{ij}$ (1).

Deux matrices $n * n$ (sites-unités) sont utilisées par le programme.

- La matrice de mémoire contient la dernière itération où l'unité j a été affectée au site i. Cette matrice est utilisée pour déterminer si un mouvement est Tabou. En l'occurrence, un mouvement est Tabou s'il affecte les deux unités échangées à deux emplacements qu'elles ont occupés au cours des t dernières itérations.

- La matrice de fréquence mémorise le nombre d'itérations où l'unité j a été placée sur le site i. Cette matrice est utilisée lors des phases d'intensification et de diversification.

4.3.2 Calcul des gains

Nous adoptons les notations suivantes :

- p représente la solution courante et associe à chaque site l'unité qui s'y trouve $(p(s) = u)$.

- p_{st} est la solution obtenue, à partir de p, en échangeant l'affectation des unités se trouvant sur les sites s et t.

- $\Delta_p(st)$ correspond à la variation de coût occasionnée par cet échange :

$$\Delta_p(st) = Cout(p_{st}) - Cout(p)$$

Les gains des différents mouvements possibles sont calculés à l'aide des coûts quadratiques d'affectation d'une unité sur un site. Le coût quadratique de placement

de l'unité k sur le site i vaut pour la solution p :

$$\gamma_p(ik) = 2\sum_{j=1}^{n} d_{ij} f_{kp(j)} \quad (2)$$

La variation de coût occasionnée par l'échange des unités sur les sites s et t, avec initialement l'unité u sur le site s et l'unité v sur le site t, peut alors se calculer grâce à la formule suivante :

$$\Delta_p(st) = \gamma_p(sv) + \gamma_p(tu) - \gamma_p(su) - \gamma_p(tv) + 4d_{st}f_{uv} \quad (3)$$

Le programme calcule donc tout d'abord les coûts quadratiques, puis en déduit les gains en utilisant la formule précédente.

Enfin, le programme sélectionne le plus petit gain non Tabou, en tenant compte de l'aspiration, puis permute les unités sélectionnées. Cet échange des unités entraîne une modification des coûts quadratiques. Le coût quadratique de placement de l'unité v sur le site s devient, après l'échange de placement des unités sur les sites m et n :

$$\gamma_{p_{mn}}(s,v) = \gamma_p(s,v) + 2(d_{sm} - d_{sn})(f_{vp(n)} - f_{vp(m)}) \quad (4)$$

Le programme modifie donc la valeur des coûts quadratiques à l'aide cette formule.

4.3.3 Structure de l'algorithme

Entrée des paramètres.
Lecture des données (Flux, Distances).
Initialisations.
Calcul du cout initial (cf (1)).
Calcul des couts quadratiques initiaux (cf (2)).

Boucle principale
Tant que compteur $\leq$ nb max d'itérations faire
    Pour tous les mouvements possibles
        Calculer les Gains (cf (3))
    fin Pour
    Sélectionner le meilleur échange non Tabou.
    Si cet échange améliore la meilleure solution **alors**
        Mémoriser cette solution.
        Remettre le compteur à 0.
    fin Si
    Calculer les nouveaux couts quadratiques (cf (4)).
Fin Tant que

Intensification.
Diversification.
Affichage des résultats.

4.3.4 Intensification

Après cette recherche Tabou classique, le programme effectue ensuite une intensification en repartant avec une liste tabou vide de la meilleure solution obtenue jusque là.

Cette intensification est répétée tant que l'on trouve une solution meilleure que celle obtenue précédemment, durant le nombre fixé d'itérations.

4.3.5 Diversification

La diversification est effectuée de façon similaire à l'intensification, si ce n'est que l'on repart d'une solution qui pour chaque site affecte une unité non déjà affectée ayant une fréquence minimale.

On n'obtient pas forcément ainsi une solution minimisant la somme des fréquences des affectations, ce qui nécessiterait la résolution d'un problème d'affectation linéaire, mais on obtient bien une solution différente de celles visitées auparavant ce qui est bien l'objectif de la diversification.

4.4 Programme avec recherche en cas d'égalité

4.4.1 Principe

Dans le programme précédent, le mouvement retenu était celui de gain minimal. Cependant, en cas d'égalité le premier échange minimal rencontré était celui finalement retenu.

Pour que ce choix ne soit pas effectué "au hasard", on va calculer les gains à plus long terme pour tous les mouvements minimaux. Ceci permet de comparer les gains de tous les échanges minimaux, et de retenir le plus intéressant. On va ainsi sélectionner le 2-échange qui donne le gain minimal à cette itération et à l'itération suivante.

Evidemment, en cas d'égalité à l'itération suivante on va choisir au hasard, car effectuer les calculs un cran plus loin prendrait trop de temps.

4.4.2 Application

Pour déterminer le meilleur 2-échange, nous allons donc effectuer les opérations suivantes pour tous les mouvements de gain minimal :

- modification des coûts quadratiques,
- calcul des gains,
- mémorisation du gain minimal à l'issue des 2 itérations.

Ce programme est celui qui obtient les meilleurs résultats en un temps raisonnable (cf tableaux des résultats: 4.7).

4.5 2-échange généralisé

Ce programme généralise ce qui a été fait dans le programme précédent.

L'idée est de calculer ici l'ensemble des 2-échanges possibles sur 2 itérations. On ne se limite donc plus au développement de ceux qui sont minimaux à l'issue de la première itération. Ceci permet d'explorer un voisinage beaucoup plus important, l'espoir étant d'obtenir en un nombre plus faible d'itérations de meilleurs résultats. Cependant, on doit calculer les gains d'un nombre de l'ordre de n^4 solutions différentes à chaque itération alors que pour un 2-échange classique, le nombre de voisins à explorer est de l'ordre de n^2.

Classiquement, on sélectionne le mouvement ayant le gain minimal à l'issue des deux itérations.

4.6 Le 3-échange

4.6.1 Généralités

Parmi les échanges possibles entre 3 éléments i, j *et* k, il existe trois 2-échanges (i-j; i-k; j-k) et deux "3-échanges purs" qui correspondent aux permutations circulaires :

- le premier, à partir de ijk, donne jki,
- le second, à partir de la me6me solution, amène à kij.

L'existence de ces 3-échanges amène à modifier le calcul du gain.

4.6.2 Application

Afin de réutiliser les formules précédemment établies, on peut décomposer le calcul du gain pour les deux 3-échanges en deux étapes :
- on calcule tout d'abord le gain de l'échange de i et j (on passe de ijk à jik),
- puis, dans un second temps, on calcule les gains des échanges de i et k (passage de jik à jki) et de j et k (passage de jik à kij).

La somme des gains de ces deux étapes permet donc bien d'obtenir le gain des deux 3-échanges.

Classiquement, le 3-échange, non Tabou, ayant le plus petit gain est sélectionné.

Nous verrons ultérieurement, lors de la présentation des résultats (cf 4.7) que le 3-échange permet de trouver des bonnes solutions en un nombre plus faible d'itérations, mais que la quantité de calculs nécessaires à chaque itération est nettement plus élevée.

4.7 Tableaux de résultats

Les programmes séquentiels que nous venons de décrire sont notés de la manière suivante dans les tableaux de résultats qui suivent :

- Tab pour le programme Tabou simple,
- Taeg pour le programme avec recherche en cas d'égalité,

- Tab3 pour le programme qui utilise le 3-échange,
- Tab2g pour le programme avec le 2-échange généralisé.

Les deux premiers tableaux présentent les résultats obtenus sur les problèmes les plus classiques de la littérature : ceux de Nugent et al [14] de tailles 15,20 et 30. Les deux tableaux suivants (tableaux 3 et 4) résument, quant à eux, les résultats sur des problèmes de plus grande taille : ceux introduits par Skorin-Kapov [19] et dont la dimension varie de 42 à 90.

	Tab	Taeg	Tab3	tab2g
Nugent15 (opt = **1150**)	**1150**	**1150**	**1150**	**1150**
Temps calcul (mn)	**0.03**	0.06	1.58	0.27
Iteration du resultat	577	519	**174**	**38**
Temps / 1000 it.(mn)	**0.04**	**0.06**	3.47	3.15
Taille liste	15	15	15	15
Nugent20 (opt = **2570**)	**2570**	**2570**	2574	**2570**
Temps calcul (mn)	**0.12**	**0.24**	8.19	5.51
Iteration du resultat	606	1056	**321**	**160**
Temps / 1000 it.(mn)	**0.06**	**0.10**	11.30	10.26
Taille liste	15	20	35	15
Nugent30 (opt = **6124**)	6128	**6124**	6128	6128
Temps calcul (mn)	0.54	0.38	66.07	31.55
Iteration du resultat	1740	226	507	372
Temps / 1000 it.(mn)	**0.16**	**0.22**	59.43	55.47
Taille liste	20	20	40	20

Tableau 1 : Résultats de nos algorithmes séquentiels (problèmes de Nugent)

Pour ces deux types de problèmes, le premier tableau (1 et 3) détaille les résultats obtenus par nos différents programmes séquentiels :

- le temps de calcul, exprimé en minutes,
- le nombre d'itérations nécessaires à l'obtention de la meilleure solution trouvée,
- le temps moyen pour effectuer 1000 itérations,
- la valeur donnée au paramètre "taille de la liste Tabou".

Le second tableau (2 et 4), quant à lui, compare les valeurs des solutions trouvées par ces différents programmes avec celles fournies par les principales méthodes approchées séquentielles :

- celle de Nugent, Vollmann et Ruml, 1968 [14],
- celle d'Elshafei, 1977 [6],

- la méthode du recuit simulé, appliquée en 1984, par Burkard et Rendl, au P.A.Q. [3],
- les deux méthodes Tabou séquentielles développées par Skorin-Kapov, en 1990 [19], [18], respectivement notées SK1 et SK2.

Optimum connu	Nugent 15 1150	Nugent 20 2570	Nugent 30 6124
Nugent	1206	2678	6378
Elshafei	1228	2598	6250
Recuit simulé	1150	2570	6148
Tabou SK1	1150	2570	6124
Tabou SK2	1150	2570	6194
Tab	1150	2570	6128
Taeg	1150	2570	6124
Tab3	1150	2574	6128
Ta2g	1150	2570	6128

Tableau 2 : Meilleures solutions trouvées par les principaux programmes séquentiels sur les problèmes de Nugent

4.8 Analyse des résultats

Les programmes séquentiels que nous venons de décrire permettent d'atteindre des résultats extrémement satisfaisants.

Certes, les programmes utilisant le 3-échange (Tab3) ou le 2-échange généralisé (Tab2g) ont des temps d'éxécution prohibitifs et il n'est donc pas raisonnable de les utiliser pour les problèmes de taille supérieure à 30. Mais les 2 programmes plus simples (Tab et Taeg) obtiennent de très bonnes performances avec des durées de calcul tout à fait raisonnables.

D'une part, la meilleure solution connue de tous les problèmes de taille inférieure à 50 est systématiquement trouvée par l'algorithme (Taeg). Même sur des applications de taille plus élevée, la solution trouvée demeure à moins de 0,5 % de la meilleure solution connue.

D'autre part, les résultats obtenus par ces algorithmes séquentiels, sont meilleurs que ceux des algorithmes séquentiels de la littérature (cf tableaux 2 et 4).

On peut noter que, de ces 2 programmes, celui avec recherche en cas d'égalité s'avére la plupart du temps le plus performant. Toutefois, dans certains cas, le programme le plus simple obtient de meilleures performances. C'est par exemple le cas pour les problèmes de Skorin-Kapov de taille 56 et 81. Ceci peut s'expliquer par le fait que ces programmes n'explorent pas de la même façon l'espace des solutions

et peuvent un peu "par hasard", trouver des solutions qu'un autre programme ne trouvera pas.

Notons, enfin, que les phases d'intensification et de diversification s'avérent indispensables. En effet, la plupart des résultats sont obtenus au cours de l'intensification qui suit la recherche Tabou classique. Cependant, le résultat optimum de Skor49 et celui de Skor56 sont obtenus au cours de la phase de diversification. Une nouvelle intensification suivant la diversification pourrait, peut-être, dans ce cas permettre d'améliorer encore les résultats obtenus.

Skorin-Kapov	Tab	Taeg
SK42 (opt = **15 812**)	15 816	**15 812**
Temps calcul (mn)	**1.08**	1.58
Iteration du resultat	1 111	1 182
Temps / 1000 it.(mn)	**0.21**	**0.42**
Taille liste	28	24
SK49 (opt = **23 386**)	23 402	**23 386**
Temps calcul (mn)	**2.01**	**4.28**
Iteration du resultat	2 080	3 363
Temps / 1000 it.(mn)	**0.24**	**1.01**
Taille liste	24	28
SK56 (opt = **34 458**)	34 494	34 518
Temps calcul (mn)	3.28	13.05
Iteration du resultat	3399	7413
Temps / 1000 it.(mn)	**0.32**	1.33
Taille liste	44	44
SK64 (opt = **48 498**)	48 758	48 684
Temps calcul (mn)	3.49	4.06
Iteration du resultat	1939	687
Temps / 1000 it.(mn)	**0.42**	1.52
Taille liste	44	40
SK72 (opt = **66 256**)	66 650	66 454
Temps calcul (mn)	7.28	16.18
Iteration du resultat	4894	4923
Temps / 1000 it.(mn)	**0.54**	2.19
Taille liste	42	42
SK81 (opt = **91 008**)	91 130	91 150
Temps calcul (mn)	6.38	10.03
Iteration du resultat	2464	2865
Temps / 1000 it.(mn)	1.10	2.49
Taille liste	44	44

Tableau 3 : Résultats de nos algorithmes séquentiels (problèmes de Skorin-Kapov)

Skorin-Kapov	42	49	56	64	72	81
Opt.connu [4]	15 812	23 386	34 458	48 498	66 256	91008
Recuit simulé	15 956	23 362	34 916	49 020	66 924	91432
Tabou SK1	15 864	23 536	34 736	48 964	66 756	91 778
Tabou SK2	15 864	23 472	34 788	48 928	66 794	91 770
Tab	15816	23 402	**34 494**	48 758	66 650	**91 130**
Taeg	**15 812**	**23 386**	34 518	**48 684**	**66 454**	91 150

Tableau 4 : Meilleures solutions trouvées par les principaux programmes séquentiels

5 Parallélisation massive de la méthode Tabou

5.1 Présentation

Nos travaux ont été effectués sur une *Connection Machine CM2* avec 16K processeurs et 2 séquenceurs. Cette machine se trouve à l'INRIA - Sophia Antipolis.

L'ensemble des développements a été effectué en C* (C-star), qui est un langage de haut niveau inspiré du C et capable de gérer des variables parallèles.

La version massivement parallèle de Tabou est directement inspirée du programme séquentiel Tab, décrit dans la section précédente. Certes, les résultats séquentiels de Taeg sont, dans la plupart des applications, légérement supérieurs à ceux de Tab. Mais, tout d'abord, cette légère amélioration s'effectue au prix d'un temps de calcul plus élevé. De plus les fonctions parallèles de haut niveau du C* ne permettent pas d'avoir une gestion fine de l'algorithme. Par exemple, la fonction qui renvoie le minimum d'une variable parallèle, ne se préoccupe pas des cas d'égalité et ne renvoie même pas les coordonnées de ce minimum. La parallélisation massive de Taeg s'avère donc particulièrement délicate.

5.2 Principes de la parallélisation massive

La structure générale de l'algorithme est donc directement inspirée du programme séquentiel *Tab*, précedemment décrit au paragraphe 4.3.

Toutefois, l'ensemble des calculs peut maintenant être effectué sur des variables parallèles. Seule la lecture des données, distances et flux, qui se trouvent dans des fichiers sur le frontal, ne peut que demeurer séquentielle. Plus précisément, le C* manipule les variables parallèles sous la forme de tableaux ou "shapes" de dimensions variables. A cause de contraintes dues à la géométrie des processeurs de la CM (hypercube), le nombre de positions d'un tableau dans chaque dimension doit être une puissance de 2. Ceci nous a amené, par exemple, à utiliser des tableaux de taille 128 x 128 pour la résolution des problèmes de Skorin-Kapov de tailles 72 et 81. Le nombre total de positions d'un tableau doit, quant à lui, être un multiple du nombre de processeurs de la partie de la CM que va utiliser le programme.

Utilisation des processeurs :

La méthode Tabou ne travaille que sur une solution unique qu'elle modifie progressivement par des échanges d'affectation. Il parait donc indispensable "d'éparpiller" autant que possible les composants de cette solution sur les processeurs. A moins de travailler sur des problèmes de taille gigantesque (plusieurs milliers d'unités), mettre une affectation par processeur ne suffit pas à une utilisation de la plupart des processeurs.

Il convient donc d'utiliser un processeur pour chaque couple site/unité. Un processeur sera donc affecté à l'étude de l'affectation d'une unité k sur un site i.

Ainsi un nombre n^2 de processeurs est-il utilisé, chacun d'entre eux comprenant :

- une variable x_{ik}, variable 0-1 indiquant si l'unité k est placée sur le site i,
- la i^{eme} ligne de la matrice D des distances,

- la k^{eme} colonne de la matrice F des flux.

5.3 Utilisation du parallélisme massif

Outre la possibilité d'utilisation de variables parallèles, c'est lors des étapes de calcul des gains des différents échanges et de sélection du meilleur d'entre eux que le parallélisme massif peut être pleinement utilisé.

Nous avons détaillé, dans le paragraphe 4.3.2, le calcul du gain de l'échange faisant passer l'unité u du site s au site t et l'unité v de t à s. Nous avons pu constater que ce calcul repose sur la détermination et la mise à jour des coûts quadratiques de placement d'une unité sur un site.

Or, dans le cadre d'un parallélisme massif, chaque processeur élémentaire, dédié à l'étude de l'affectation d'une unité k sur un site i, peut calculer et tenir à jour le coût quadratique correspondant. Chaque coût quadratique peut ainsi, à l'occasion d'un mouvement, être mis à jour, sur chaque processeur, en un calcul élémentaire (cf formule 4 du paragraphe 4.3.2).

Le calcul du gain d'un 2-échange, somme de 4 coûts quadratiques (cf formule 3 du paragraphe 4.3.2) peut, quant à lui, être calculé sur un processeur en O(1) opération, moyennant 3 communications inter-processeurs.

Ainsi, si, en séquentiel, il est nécessaire de calculer successivement les $\frac{n(n-1)}{2}$, soit $O(n^2)$, gains occasionnés par les différents échanges effectuables à partir d'une solution quelconque, dans un contexte massivement parallèle, chacun d'entre eux peut être traité par un processeur.

Il s'agit, ensuite, de déterminer quel est l'échange procurant le gain le plus important. Pour cela, $O(log\ n)$ comparaisons s'avèrent suffisantes.

Ces deux points justifient l'accélération en $O(\frac{n^2}{log\ n})$ pouvant être théoriquement atteinte par une version massivement parallèle de la méthode Tabou. Notons toutefois que cette accélération est celle correspondant à la comparaison du programme s'exécutant sur n^2 processeurs élémentaires et du programme ne s'exécutant que sur un seul de ces processeurs. Or, il n'est pas envisageable, vues la taille et la puissance de ces processeurs élémentaires, d'exécuter le programme sur un seul processeur.

5.4 Résultats obtenus

Les résultats sont décrits dans deux tableaux :

- le premier tableau (tableau 5) résume les valeurs des meilleures solutions trouvées, sur les problèmes de Skorin-Kapov, par les principales méthodes Tabou parallèles
  - celle de Taillard [21], notée Tab Tail,
  - celle de Chakrapani et Skorin-Kapov [4], notée Tab Ch-SK,
  - la notre enfin, notée Tab.cs,
- le second tableau (tableau 6) détaille les conditions d'obtention de ces différents résultats. Il indique ainsi :

- le temps de calcul, exprimé en minutes,
- l'itération à laquelle la meilleure solution a été trouvée,
- le temps moyen (en minutes) nécessaire au déroulement d'un millier d'itérations,
- le nombre total d'itérations effectuées,
- la valeur donnée à la taille de la liste Tabou.

Skorin-Kapov	42	49	56	64	72	81
Opt.connu [4]	15 812	23 386	34 458	48 498	66 256	91 008
Tab Tail	15 812	23 386	34 458	48 498	66 256	91 028
Tab Ch-SK	15 812	23 386	34 458	48 498	66 256	91 010
Tab.cs	15 812	23 412	34 696	48 778	66 512	91 176

Tableau 5 : Meilleures solutions trouvées par les méthodes Tabou parallèles

5.5 Analyse des résultats

Les résultats obtenus sont légèrement moins bons que ceux atteints par Taillard et Chakrapani-Skorin-Kapov. Ils restent toutefois d'une qualité tout à fait comparable, puisque, dans le pire des cas, la solution n'est dégradée que de 0,69 %.

D'autre part, les résultats de Taillard et de Skorin-Kapov ont été obtenus avec un nombre d'itérations beaucoup plus élevé :

- de l'ordre de 100 000 pour le programme de Skorin-Kapov,
- de l'ordre de plusieurs centaines de milliers pour le programme de Taillard.

et un temps d'exécution plus long (principalement pour Taillard). Les temps d'accès restreints à la CM2, de l'ordre du quart d'heure en temps partagé, n'ont pas permis d'améliorer significativement les résultats.

Notre programme reste relativement lent, d'une part à cause du mode de communication du *C** (il aurait fallu faire appel à des fonctions PARIS), d'autre part du fait des temps de communication élevés avec le frontal. Remarquons toutefois que les temps d'exécution n'augmentent que très peu avec la taille des problèmes.

Notons enfin que l'apport de la *Connection Machine* est validé dans la mesure où nous obtenons des résultats comparables à ceux fournis par nos programmes séquentiels, en un temps plus court.

Skorin-Kapov	Tab.cs	Tab Ch-SK	Tab Tail
SK42 (opt = **15 812**)	15 812	15 812	15 812
Temps calcul (mn)	4.34	env. 5.30	env. 4.30
Iteration du resultat	2 940	2 866	
Temps / 1000 it.(mn)	0.55	0.05	0.10
Total Itérations	**4 940**	**57 691**	**26 836**
Taille liste	24	24	29
SK49 (opt = **23 386**)	23 412	23 386	23 386
Temps calcul (mn)	2.01	env. 6.40	env. 1h10
Iteration du resultat	**1129**	17 295	
Temps / 1000 it.(mn)	1.02	0.06	0.15
Total Itérations	**2 129**	**71 801**	**280 662**
Taille liste	28	32	34
SK56 (opt = **34 458**)	34 696	34 458	34 458
Temps calcul (mn)	1.34	env. 10.00	env. 3h00
Iteration du resultat	**733**	4 127	
Temps / 1000 it.(mn)	1.16	0.06	0.20
Total Itérations	**1 233**	**105 345**	**537 061**
Taille liste	44	36	39
SK64 (opt = **48 498**)	48 778	48 498	48 498
Temps calcul (mn)	3.03	env. 11.40	env. 4h15
Iteration du resultat	1 237	4 517	
Temps / 1000 it.(mn)	1.20	0.06	0.25
Total Itérations	**2 237**	**116 878**	**687 652**
Taille liste	44	44	47
SK72 (opt = **66 256**)	66 512	66 260	66 256
Temps calcul (mn)	7.07	env. 15.00	env. 6h40
Iteration du resultat	3 716	30 470	
Temps / 1000 it.(mn)	1.23	0.06	0.30
Total Itérations	**5 116**	**150 188**	**env. 800 000**
Taille liste	40	48	
SK81 (opt = **91 008**)	91 176	91 010	91 028
Temps calcul (mn)	5.00	env. 19.00	env. 11h.
Iteration du resultat	2 524	132 549	
Temps / 1000 it.(mn)	1.25	0.06	0.40
Total Itérations	**3 524**	**189 097**	**env 1 000 000**
Taille liste	46	52	

Tableau 6 : Résultats détaillés des méthodes Tabou parallèles

6 Conclusion

Le problème d'Affectation Quadratique possède des applications pratiques très nombreuses et de taille relativement élevées.

Les méthodes exactes demeurant à l'heure actuelle, extrèmement limitées quant à la taille des problèmes qu'elles peuvent résoudre, il s'avère indispensable de faire appel à des heuristiques performantes.

Si les travaux antérieurs de Taillard ainsi que de Chakrapani - Skorin Kapov montraient déjà la supériorité de la méthode Tabou sur toutes les autres heuristiques, les résultats que nous avons obtenus au cours de cette étude entérinent cette supériorité.

En effet, en développant et en testant tout d'abord différentes versions de la méthode Tabou en séquentiel, nous avons d'ores et déjà pu obtenir des résultats extrèmement satisfaisants.

Via l'introduction d'une recherche "en cas d'égalité" qui consiste à sélectionner le mouvement qui donne le résultat minimal à une itération donnée mais aussi à la suivante, nous sommes parvenu à obtenir systématiquement la meilleure solution connue pour les problèmes de taille inférieure à 50 en des temps inférieurs à 5 minutes.

Même sur des problèmes de taille supérieure, l'erreur que commet ce programme demeure inférieure à 0,5 %, dans le pire des cas, en comparaison avec la meilleure solution connue.

On peut noter que ce programme dépasse ainsi les performances de toutes les précédentes heuristiques séquentielles.

Nous avons ensuite développé une version massivement parallèle de cet algorithme sur la Connection Machine CM2.

Les résultats obtenus ont montrés que si la vitesse de calcul est relativement lente, on a pu obtenir des solutions de bonne qualité en un nombre d'itérations beaucoup plus petit que dans les autres programmes parallèles.

Une gestion plus précise de la communication entre les processseurs à l'aide du langage parallèle de plus bas niveau PARIS, devrait permettre d'obtenir de meilleures performances en termes de vitesse de calcul.

Remerciements

Les auteurs sont très reconnaissants envers l'INRIA - Sophia Antipolis pour la mise à disposition de temps de calcul sur la Connection Machine CM-2.

Bibliographie

[1] E. Balas and J.B. Mazzola. Quadratic 0-1 programming by a new linearization. In *Presented at the TIMS/ORSA Meeting*, Washington D.C., 1980.

[2] R. Burkard, S. Karisch, and F. Rendl. Qaplib - a quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991.

[3] R. Burkard and F. Rendl. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17:169–174, 1984.

[4] J. Chakrapani and J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. Technical Report HAR-91-06, Harriman School for Management and Policy, NY 11794, 1991.

[5] A. David and C. Roucairol. Un algorithme de composition spatiale. *Notes méthodologiques en Architecture et Urbanisme*, 2:69–87, 1973. Centre MMI, Institut de l'Environnement.

[6] A. Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly*, 28:167–179, 1977.

[7] G. Finke, R. Burkard, and F. Rendl. Quadratic assignment problems. *Annals of Discrete Math.*, 28:61–82, 1987.

[8] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[9] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[10] F. Glover and M. Laguna. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*. University of Colorado at Boulder, 1992.

[11] T.C. Koopmans and M.J. Beckman. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[12] T. Mautor. *Contribution à la résolution des problèmes d'implantation : algorithmes séquentiels et parallèles pour l'affectation quadratique.* Thèse d'université, Université Paris VI, 4, place Jussieu, 75252 Paris Cedex 05, February 1993.

[13] T. Mautor and C. Roucairol. A new exact algorithm for the solution of quadratic assignment problems. *Discrete Applied Mathematics*, à paraitre, 1993. MASI-RR-92-09 - Université de Paris 6, 4 place Jussieu, 75252 Paris Cédex 05.

[14] C. Nugent, T. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150–173, 1968.

[15] J.S. Riche. Application de la recherche tabou au placement des circuits électroniques. Technical report, Université de Paris 6, 1992. Mémoire de D.E.A.

[16] C. Roucairol. *Du séquentiel au parallèle: la recherche arborescente et son application à la programmation quadratique en variable 0-1.* Thèse d'état, Université Paris VI, June 1987.

[17] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[18] J. Skorin-Kapov. Extensions of a tabu adaptation to the quadratic assignment problem. Technical Report HAR-90-006, Harriman School for Management and Policy, SUNY at Stony Brook, NY 11794, 1990.

[19] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2:33–45, 1990.

[20] L. Steinberg. The backboard wiring problem : a placement algorithm. *SIAM Review*, 3:37–50, 1961.

[21] E. Taillard. Robust tabu search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS-LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399