RAIRO. OPERATIONS RESEARCH

DALILA TAYACHI
PHILIPPE CHRÉTIENNE
KHALED MELLOULI

Une méthode tabou pour l'ordonnancement multiprocesseur avec délais de communication

RAIRO. Operations Research, tome 34, nº 4 (2000), p. 467-485
<http://www.numdam.org/item?id=RO_2000__34_4_467_0>

RAIRO Operations Research
RAIRO Oper. Res. **34** (2000) 467-485

UNE MÉTHODE TABOU POUR L'ORDONNANCEMENT MULTIPROCESSEUR AVEC DÉLAIS DE COMMUNICATION (*)

par Dalila TAYACHI ([1]), Philippe CHRÉTIENNE ([2]) et Khaled MELLOULI ([1])

Résumé. – *Cet article traite des problèmes d'ordonnancement de n tâches sur m processeurs identiques, en présence de délais de communication. Une nouvelle approche de modélisation par un graphe d'arbitrage et de résolution par une méthode tabou est proposée. Des solutions initiales sont construites par des algorithmes de liste, qui sont ensuite améliorées par un algorithme tabou opérant en deux phases. Des expérimentations effectuées sur des graphes générés aléatoirement montrent que notre méthode est performante et qu'elle se comporte mieux que les principales heuristiques existantes.*

Mots clés : Problèmes d'ordonnancement, délais de communication, graphe d'arbitrage, méthode tabou, algorithme de liste.

Abstract. – *This paper deals with the problem of scheduling n tasks on m identical processors in the presence of communication delays. A new approach of modelisation by a decision graph and a resolution by a tabu search method is proposed. Initial solutions are constructed by list algorithms, and then improved by a tabu algorithm operating in two phases. The experiments carried on arbitrary graphs show the efficiency of our method and that it outperformed the principle existent heuristics.*

Keywords: Scheduling problems, communication delays, decision graph, tabu search, list algorithm.

INTRODUCTION

L'étude des problèmes d'ordonnancement avec délais de communication constitue un nouveau domaine de recherche qui prend de plus en plus d'importance, surtout avec le développement actuel d'ordinateurs à architecture répartie. Un problème d'ordonnancement avec délais de communication est représenté par un ensemble de tâches reliées devant s'exécuter sur un réseau de processeurs. Outre les contraintes de précédence classiques existant entre les tâches, des délais de communication entre des tâches s'exécutant sur des processeurs différents doivent être pris en compte. L'objectif de ce problème d'ordonnancement est la minimisation du makespan. Ce problème est NP-complet puisque même pour des durées

(*) Reçu en juin 1999.
([1]) Institut Supérieur de Gestion, 41 rue de Liberté, Cité bouchoucha, Le Bardo, Tunis, Tunisie.
([2]) Laboratoire d'Informatique de Paris, Université de Paris VI, Paris, France.

Recherche opérationnelle/Operations Research, 0399-0559/00/04/$ 7.00

d'exécution et des délais de communication unitaires, le problème est déjà NP-complet (Smith, 1987). Des algorithmes polynomiaux spécifiques à des instances très particulières ont été développés. Par exemple, Colin et Chrétienne (1991) ont développé un algorithme exact pour le cas de petits délais de communication, de duplication et d'un nombre infini de processeurs, Chrétienne (1989) a proposé un algorithme exact pour le cas d'un nombre infini de processeurs, petits délais de communication et où le graphe des précédence est une arborescence. Des approches heuristiques avec garantie de performance et avec moins d'hypothèses restrictives sur le problème ont été également développées, à savoir l'algorithme de Hanen et Munier (1997), les algorithmes de Hwang *et al.* (1989) et une méthode exacte (Daddi-Moussa, 1997).

Dans cet article, nous nous intéressons à la modélisation et la résolution du problème général avec délais de communication par une méthode tabou. Les tests que nous avons effectués sur des anti-arborescences générées aléatoirement prouvent que notre approche est assez efficace et se compare avantageusement aux méthodes déjà existantes.

Nous commençons dans la section 1 par la présentation du problème et les notations utilisées. Dans la section 2 nous passons en revue les principales méthodes de résolution existantes, puis dans la section 3, nous définissons les différents paramètres de la méthode tabou que nous proposons pour la résolution du problème. L'évaluation numérique de l'approche est l'objet de la section 4.

1. DÉFINITION DU PROBLÈME ET NOTATION

Soit $I = \{1, ..., n\}$ un ensemble de n tâches non préemptives. Chaque tâche i est caractérisée par sa durée d'exécution P_i, par l'ensemble de ses successeurs (respectivement prédécesseurs) $S(i)$ (respectivement $P(i)$). Ces tâches forment un graphe sans circuits, valué sur ses sommets et ses arcs $G = (I, U, P, C)$. Chaque sommet de G correspond à une tâche et chaque arc (i, j) correspond à une contrainte de précédence. Un nombre entier positif $\eta(i, j)$, qui représente la longueur du message envoyé de la tâche i vers la tâche j, est associé à chaque arc (i, j). On dispose de m processeurs $\{\Pi_1, ..., \Pi_m\}$ reliés entre eux par un réseau complet d'interconnexion (tous les processeurs sont reliés et les communications sont bidirectionnelles comme l'indique la figure 1). Un paramètre $d(\Pi_i, \Pi_j)$ désignant la durée de transfert d'une unité d'information est associé à chaque couple de processeurs Π_i et Π_j. Si un arc relie deux tâches i et j, alors

la durée du transfert de données est définie par $C_{ij} = \eta(i,j) * d(\Pi_i, \Pi_j)$ où Π_i (respectivement Π_j) est le processeur sur lequel s'exécute la tâche i (respectivement j). Lorsque les deux tâches s'exécutent sur un même processeur, la durée C_{ij} est nulle $(d(\Pi_i, \Pi_j) = 0)$.

On distingue dans la littérature plusieurs types de réseaux qui varient selon la topologie des liens entre les processeurs, la capacité des liens de communication dans le réseau, le mode d'acheminement des données entre les tâches à travers le réseau. On peut citer par exemple le réseau par bus, en anneau, étoile, etc.

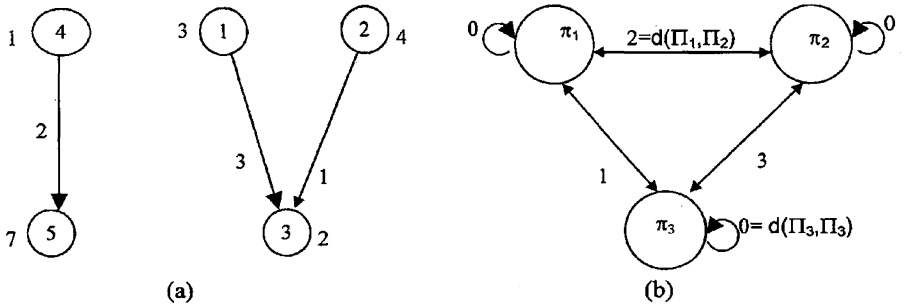


Figure 1. – (a) Graphe de précédence. (b) Réseau de processeurs.

Déterminer un ordonnancement réalisable pour ce problème consiste à associer à chaque tâche une date de début d'exécution t_i et un processeur Π_i (Chrétienne et Picouleau, 1995) tels que :

- pour chaque arc du graphe des précédence (i,j) : $t_j \geq t_i + P_i$ si $\Pi_i = \Pi_j$ et $t_j \geq t_i + P_i + C_{ij}$ si $\Pi_i \neq \Pi_j$;
- à chaque instant, un processeur exécute au plus une tâche et au plus m processeurs sont actifs.

L'objectif est de déterminer un ordonnancement réalisable qui minimise le makespan. Selon la notation définie par (Veltman, 1990), ce problème est représenté par $P/\mathrm{Prec}, P_j, C_{jk}/C_{\max}$.

Si on désigne par $P_{\min}$ (respectivement $P_{\max}$) la durée d'exécution minimale (respectivement maximale) et par $C_{\max}$ (respectivement $C_{\min}$) le plus grand (respectivement le plus petit) délais de communication alors on parle de cas avec petits délais de communication SCT (Small Communication Times) si le rapport $C_{\max}/P_{\min} \leq 1$. Lorsque $P_{\max}/C_{\min} \leq 1$, on parle de problèmes avec grands délais de communication.

2. TRAVAUX ANTÉRIEURS

Hanen et Munier (1997) ont proposé un algorithme de liste pour le cas de durées de tâches unitaires et de délais de communication unitaires. Leur algorithme présente une garantie de performance :

$$C_L/C^* \leq 7/3 - 4/3m$$

où C_L désigne la durée de l'ordonnancement fourni par cet algorithme et C^* est la durée minimale d'un ordonnancement. Les mêmes auteurs proposent une extension de cet algorithme pour le cas de petits délais de communication.

Hwang *et al.* (1989) ont développé un algorithme nommé ELS (Extended List Schedule) qui opère en deux étapes. Dans la première étape, les délais de communication sont ignorés et une solution est calculée par un algorithme de liste classique, puis dans une seconde étape, les délais de communication sont intégrés en conservant l'affectation et l'ordre des tâches sur les processeurs.

L'algorithme ELS offre la performance suivante :

$$C(\text{ELS}) \leq (2 - 1/m)C^* + d_{\max} \sum_{(i,j) \in U} \eta(i,j)$$

où C^* désigne la durée optimale pour le problème correspondant sans délais de communication, $d_{\max}$ désigne la durée de transfert maximale entre des processeurs. Les même auteurs ont développé une autre heuristique ETF (Earliest Task First) de performance meilleure que celle de ELS. ETF constitue une généralisation de l'algorithme de liste de Smith (1987) pour le cas de durées et délais de communication unitaires. Comme tout algorithme de liste qui affecte une tâche prête à un processeur dès qu'il est libre, ETF affecte la tâche de plus petite date au plus tôt sur le processeur libre qui permet son exécution le plus tôt possible. Elle offre la possibilité à l'instant courant t de retarder la décision d'ordonnancement d'une tâche à l'instant $\theta > t$ si une tâche se termine entre t et θ. Cette heuristique présente la garantie de performance suivante :

$$C(\text{ETF}) \leq (2 - 1/m)C^* + D$$

où D représente la durée maximale de communication le long d'une chaîne de G.

Plus récemment, Daddi-Moussa (1997) a proposé une première méthode exacte pour la résolution du problème général d'ordonnancement avec délais de communication lorsque le réseau des processeurs est complet. Les durées

des tâches, les délais de communication et le graphe de précédence sont arbitraires. La recherche des solutions optimales se fait selon la méthode par séparation et évaluation. À chaque nœud, est associé un sous-ensemble de tâches affectées, les dates de début d'exécution de ces tâches et les processeurs sur lesquels elles sont affectées. On définit alors l'ensemble C des tâches dites candidates c'est-à-dire qui ne sont pas encore ordonnancées et telles que les prédécesseurs ont été déjà exécutés, Π_k le processeur k, r_k le nombre de tâches affectés à Π_k, et R_k l'ensemble des tâches retardées associées à Π_k. Une tâche est dite retardée si elle n'est pas affectée à Π_k au rang $r_k + 1$ ou elle est affectée à un autre processeur diffèrent de Π_k. Lors de la séparation, l'auteur construit pour toute tâche i de $C - R_k$ un fils en affectant la tâche i à Π_k à la position $r_k + 1$. Un fils supplémentaire pour lequel aucune tâche candidate ne sera de rang $r_k + 1$ dans Π_k est également construit. On cite parmi les bornes inférieures utilisées, la borne de Graham et la borne du plus long chemin, et parmi les bornes supérieures les algorithmes de liste. Vu la complexité du problème, les instances résolues par cette méthode en un temps raisonnable sont de petite taille ($n \leq 14$). Dans cette perspective et afin de résoudre des problèmes généraux de grande taille nous avons opté pour la résolution du problème par une méthode tabou.

3. MODÉLISATION ET RÉSOLUTION DU $P/\mathrm{Prec}, P_j, C_{jk}/C_{\max}$

Nous proposons une extension de la méthode proposée pour la résolution du même problème sans délais de communication (Tayachi *et al.* 1998). Dans notre modélisation du problème, outre les relations de précédence existantes entre les tâches, nous tenons compte de relations supplémentaires qui expriment les positions relatives des tâches indépendantes sur un même processeur. Ainsi, nous modélisons une instance donnée du problème $P/\mathrm{Prec}, P_j, C_{jk}/C_{\max}$ par un graphe, appelé graphe d'arbitrage, $G_c(A) = (I, U \cup A)$ où U représente les arcs de précédence tandis que A représente les arcs d'arbitrage. Un arc $(i, j) \in A$ est dit arc d'arbitrage si les tâches i et j sont indépendantes et consécutives sur un même processeur. La valuation d'un arc d'arbitrage (i, j) est égale à la durée P_i de la tâche i, puisque $d(\Pi_i, \Pi_j)$ est nulle. Le graphe d'arbitrage $G_c(A)$ ainsi construit est sans circuit et les dates de l'ordonnancement sont celles de l'ordonnancement au plus tôt. La durée de l'ordonnancement n'est autre que la longueur du chemin critique dans le graphe d'arbitrage $G_c(A)$. Calculer un ordonnancement optimal revient à chercher un arbitrage A tel que le chemin critique de $G_c(A)$ soit minimal.

Étant donné cette modélisation, nous proposons une méthode tabou qui à partir d'un arbitrage initial, agit sur les tâches critiques pour construire un arbitrage voisin.

Nous illustrons cette modélisation sur un problème à 9 tâches et 3 processeurs (Fig. 2). Les chiffres à coté des sommets représentent les durées des tâches, ceux à coté des arcs représentent les délais de communication, les arcs en pointillé représentent les arcs d'arbitrage et ceux en gras représentent le chemin critique. Les tâches en gras dans (b) sont les tâches critiques. W représente la durée de l'ordonnancement.

Nous présentons dans ce qui suit les éléments de la méthode tabou à savoir, le voisinage, la solution initiale, la liste tabou et les critères d'arrêt.

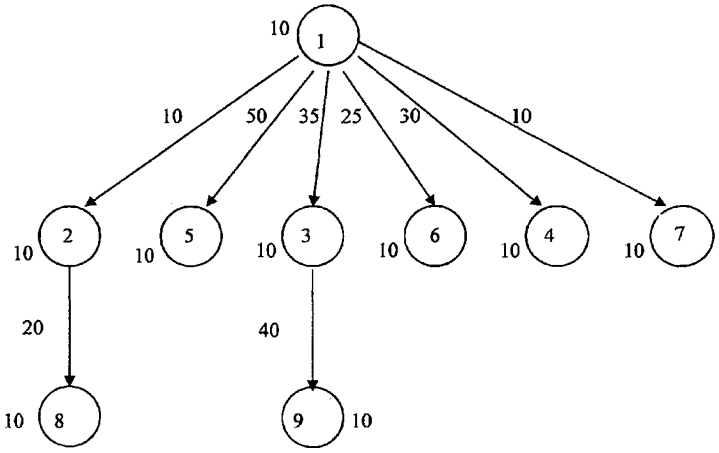
Solution initiale

Nous avons développé deux versions de la méthode tabou qui ne différent que par le choix de la solution initiale. Dans la première que nous notons TLPT (Tabu Largest Processing Time), nous construisons une solution initiale par la méthode ELS (Extended List Schedule). Dans la première étape de ELS, nous avons adopté la liste de priorité qui classe les tâches selon l'ordre décroissant de leurs durées. Dans la deuxième version tabou TETF (Tabu Earliest Task First), une solution initiale est construite par l'heuristique ETF (Earliest Task First).

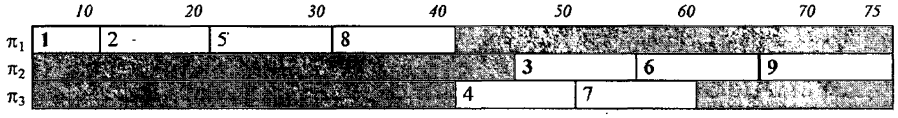
Voisinage

Notre méthode tabou est composée de deux algorithmes opérant successivement que nous dénotons par Tabou 1 et Tabou 2. Le deuxième algorithme Tabou 2, démarre de la meilleure solution obtenue par le premier algorithme Tabou 1, et chaque algorithme possède un voisinage spécifique :

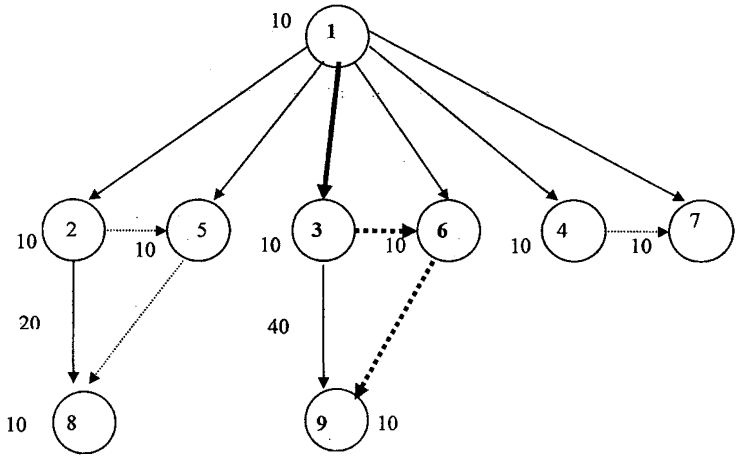
(1) dans le voisinage de Tabou1, nous tentons de regrouper les tâches critiques du graphe d'arbitrage initial sur un même processeur afin de diminuer l'effet des communications. En effet, ce sont essentiellement les tâches critiques qui entrent dans le calcul de la durée de l'ordonnancement. Par conséquent, si on place certaines de ces tâches sur un même processeur, ceci permet d'éviter les délais de communication et diminuer la durée totale. Nous calculons à chaque itération de l'algorithme le chemin critique du graphe d'arbitrage courant. Soit $i_1 - i_2 - \ldots - i_l - i_{l+1} - \ldots - i_p$



(a) Graphe de précédence



(b) Ordonnancement associé au graphe (a) pour 3 processeurs, $W = 75$



(c) Un graphe d'arbitrage associé au problème $P_3/\mathrm{Prec}, P_j, C_{jk}/C_{\max}$

Figure 2. – Illustration d'un graphe d'arbitrage sur un problème à 9 tâches et 3 processeurs.

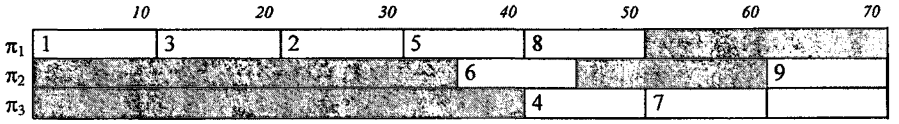


Figure 3. – Une solution voisine obtenue par le déplacement de la tâche 3 sur π_1, $w = 70$.

Tabou 1

Étape 0.

Construire une solution initiale S_o.

La liste tabou est initialisée à vide.

La meilleure solution S^* est S_o.

Étape 1. La solution courante est S_i

- déterminer le chemin critique du graphe d'arbitrage associé à S_i.

Étape 2. Construire le voisinage $V(S_i)$ comme suit :

- pour chaque couple de tâches critiques successives (i, j) qui s'exécutent sur deux processeurs π_i et π_j distincts faire :
  - * regrouper i avec j sur le processeur π_i et déterminer l'ordonnancement associé ;
  - * regrouper j avec i sur le processeur π_j, et déterminer l'ordonnancement associé.

Étape 3. Soit S_v la meilleure solution voisine non tabou ou satisfaisant les critères d'aspiration obtenue par le placement de i sur π_j

- la solution courante devient $S_{i+1} = S_v$;
- stocker le triplet (i, π_j, π_i) dans la liste tabou pendant k itérations où k désigne la taille de la liste ;
- si S_v est meilleure que S^* alors $S^* = S_v$;
- $i := i + 1$.

Étape 4. Si les critères d'arrêt sont satisfaits aller à Tabou 2, sinon revenir à l'Étape 1.

ce chemin, alors une solution voisine est obtenue en plaçant i_l sur le processeur de i_{l+1}, $\pi(i_{l+1})$ après le dernier de ses ascendants.

La figure 3 illustre une solution voisine de la solution présentée dans la figure 2b. Cette solution voisine est obtenue en regroupant la tâche critique 3 avec la tâche critique 1 sur le processeur II_1 ce qui a permis d'éviter un délai de communication $C_{13} = 35$ entre les tâches 1 et 3. Ceci a ramené la durée de l'ordonnancement de 75 à 70.

(2) le deuxième algorithme constitue une étape de post-optimisation dans notre méthode tabou. En effet, après les améliorations effectuées au cours de tabou 1, une solution assez bonne est obtenue. À partir de cette solution, nous essayons encore d'améliorer en équilibrant les charges des processeurs (la charge d'un processeur désigne la date de fin de la

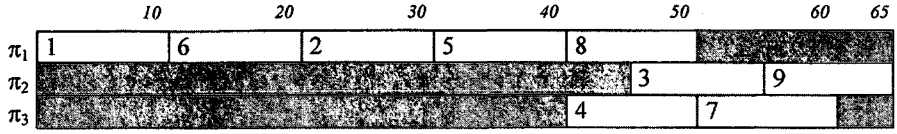


Figure 4. – Une solution voisine obtenue suite au déplacement de la tâche 6 sur π_1, $w = 65$.

Tabou 2

Étape 0.

La solution initiale est la meilleure solution obtenue à la fin de l'algorithme tabou1, soit S^*

- la liste tabou est initialisée à vide
- la meilleure solution S_2^* est S^*.

Étape 1. La solution courante est S_i

- déterminer le chemin critique du graphe d'arbitrage associé à S_i,
- déterminer le processeur de charge minimale $\pi_{\min}$.

Étape 2. Construire le voisinage $V(S_i)$ comme suit :

- déplacer chaque tâche critique sur $\pi_m in$ et déterminer l'ordonnancement associé.

Étape 3. Soit S_v la meilleure solution voisine non tabou ou satisfaisant les critères d'aspiration obtenue par le placement de i sur $\pi_{\min}$.

- la solution courante devient $S_{i+1} = S_v$
- stocker le triplet $(i, \pi_{\min}, \pi_i)$ dans la liste tabou pendant k itérations où k désigne la taille de la liste.
- si S_v est meilleure que S_2^* alors $S_2^* = S_v$
- $i := i + 1$.

Étape 4. Si les critères d'arrêt sont satisfaits arrêt, sinon revenir à l'Étape 1.

dernière tâche qui s'exécute sur ce processeur). Ainsi une solution voisine de celle de la figure 2b est obtenue en affectant une tâche critique au processeur de charge minimale. Pour le problème présenté précédemment, nous déplaçons par exemple la tâche 6 sur le processeur de charge minimale qui est π_1 dans ce cas.

L'algorithme TETF (respectivement TLPT) consiste alors à exécuter successivement tabou 1 et tabou 2, en démarrant dans Tabou 1 d'une solution initiale construite par l'heuristique ETF (respectivement ELS).

Notons qu'il est aussi possible de fusionner les deux algorithmes tabou 1 et tabou 2 en un seul algorithme, en considérant à chaque itération l'union du voisinage de tabou 1 et de celui de tabou 2.

Critères d'arrêt

L'algorithme s'arrête lorsqu'une des conditions suivantes est satisfaite :

- le nombre d'itérations effectuées atteint 10 n où n est la taille du problème (5 n pour chacun des algorithmes Tabou 1 et Tabou 2);
- le nombre d'itérations successives sans amélioration atteint 2 n pour chacun des algorithmes Tabou 1 et Tabou 2;
- le voisinage de l'algorithme tabou 2 est vide;
- la durée de l'ordonnancement obtenu par l'un des algorithmes Tabou 1 ou Tabou 2 est égale à une borne inférieure. Nous considérons deux bornes suivant le type de délais de communication des instances étudiées :

  * pour le cas avec petits délais de communication, nous considérons le maximum entre le rapport $\sum_{i=1}^{n} P_i/m$ et la durée de l'ordonnancement fourni par l'algorithme de Chrétienne (1989) adapté au cas d'un nombre limité de processeurs. Il est à noter que ce dernier est optimal lorsque le graphe de précédence est une anti-arborescence et le nombre de processeurs est illimité;

  * pour le cas de grands délais de communication, nous considérons comme borne le maximum entre le rapport $\sum_{i=1}^{n} P_i/m$ et la longueur du chemin critique du graphe de précédence. Remarquons que cette borne déjà utilisée dans le cas sans délais de communication est de plus mauvaise qualité lorsque l'on considère les délais de communication.

Liste tabou

Nous gardons la même liste que dans le cas sans délais de communication (Tayachi *et al.* 1998). Si à une itération nous déplaçons une tâche i d'un processeur Π_i vers un processeur Π_j, alors le mouvement qui consiste à déplacer i de Π_j vers Π_i est déclaré tabou pendant un certain nombre d'itérations. La longueur de la liste tabou varie dans l'intervalle [3, 7] suivant n.

Critères d'aspiration

Nous utilisons le critère d'aspiration le plus simple. Il consiste à révoquer le statut tabou au triplet (i, Π_i, Π_j) si cela permet d'obtenir une solution meilleure que toutes les solutions rencontrées jusqu'à présent.

4. PROBLÈMES TRAITÉS ET COMPARAISON

Étant donné qu'un problème avec délais de communication est caractérisé par le graphe de précédence, les délais de communication, le nombre de processeurs, nous avons classé les problèmes traités selon ces paramètres. Ainsi nous avons traité le cas d'anti- arborescence avec grands délais de communication et durées de tâches constantes et le cas d'anti-arborescence avec petits délais de communication et durées de tâches constantes. Les anti-arborescences sont générées aléatoirement conformément à l'algorithme décrit dans Alonso et Schott (1995). Cet algorithme conçu pour la génération des arborescences, que nous avons adapté trivialement au cas des anti-arborescences, présente l'avantage d'équiprobabilité entre les arborescences ayant le même nombre de sommets. Il est à noter que notre approche est applicable pour tous type de graphes de précédences, et que nous avons choisi de nous limiter aux anti-arborescences, car nous disposons de bornes spécifiques. L'hypothèse de durées constantes est considérée afin de diminuer le nombre de paramètres intervenant dans notre étude. Nous avons choisi un nombre de processeurs fonction du nombre de feuilles afin de tenir compte de la largeur des anti-arborescences. Nous avons alors considéré les trois cas suivants :

- un nombre de processeurs égal au tiers du nombre de feuilles : $m = 1/3f$;
- un nombre de processeurs égal au deux tiers du nombre de feuilles : $m = 2/3f$;
- un nombre de processeurs égal au nombre de feuilles: $m = f$ où f désigne le nombre de feuilles.

Nous avons appliqué nôtre méthode à environ 840 problèmes de tailles et caractéristiques différentes. Ces tests ont été effectués sur un PC pentium II cadencé à 200 MHz, et le programme a été implémenté en Turbo Pascal 7.0.

Nous présentons les résultats obtenus par classe de problèmes dans les tableaux qui suivent où nous indiquons :

- la moyenne des gains par rapport à chacune des heuristiques ELS et ETF. Le gain d'une solution S_t obtenue par une méthode tabou par rapport à la solution S_i générée par une heuristique quelconque est donnée par :

$$100^*(W_k(S_t) - W_k(S_i))/W_k(S_i) \qquad (1)$$

où W_k désigne la durée de l'ordonnancement obtenu pour l'instance numéro k. La formule (1) donne un nombre négatif pour un gain.

- la proportion PA_1 (respectivement PA_2) des expériences pour lesquelles une amélioration est apportée par rapport à l'heuristique ELS (respectivement ETF);
- la proportion PO_1 (respectivement PO_2) des expériences pour lesquelles la borne est atteinte parmi les expériences améliorés par TLPT (respectivement TETF);
- la moyenne des écarts relatifs E/B entre le makespan des solutions trouvées et la borne inférieure. L'écart relatif d'une solution S et la borne B pour l'instance numéro k est donnée par :

$$100^*(W_k(S) - B)/B.$$

a) Anti-arborescences avec grands délais de communication et durées constantes

Nous avons généré 140 anti-arborescences de tailles différentes, 20 pour chaque nombre de tâches n où $n = 10, 20, 30, 40, 60, 80$ et 100. Pour chacune de ces anti-arborescences, nous avons considéré trois cas pour le nombre de processeurs soit au total 420 tests.

Nous supposons dans cette classe de problèmes que les durées P_i sont constantes afin de faciliter l'interprétation des résultats. P_i est fixé arbitrairement à 10. Les délais de communication C_{ij} sont supérieurs ou égaux aux durées des tâches. C_{ij} est une variable aléatoire uniformément distribuée dans l'intervalle $\left[P_i, \left(\sum_{i=1}^{n} P_i\right)/2\right]$ soit dans l'intervalle $[10, 5n]$.

Nous synthétisons les résultats dans le tableau 1 qui suit :

Remarque 1 : Nous n'avons pas présenté dans le tableau 1 les écarts par rapport à la borne, car dans ce cas de grands délais de communication, les bornes disponibles sont de très mauvaises qualités. Ceci explique la faiblesse des proportions d'optimalité.

Interprétation des résultats

Les résultats montrent que les deux variantes de la méthode tabou, TLPT et TETF sont meilleures que les deux heuristiques ELS et ETF. En effet, concernant ELS, TLPT permet d'améliorer nettement les solutions calculées par l'heuristique ELS quelque soit le nombre de processeurs et le nombre de tâches puisque la proportion des cas pour lesquels la méthode tabou a amélioré la solution obtenue par ELS est de 100 % sauf pour le cas de 10 tâches où cette valeur atteint 90 %. Le gain moyen maximal atteint 69 % dans

TABLEAU 1
Comparaison de TLPT et TETF avec les heuristiques ELS et ETF dans le cas de grands délais de communications

n	m	npg	G1/ELS	PA1	PO1	G1/ETF	G2/ETF	PA2	PO2
10	f/3	20	17	90	11	9	11	70	7
	2f/3	20	19	90	11	11	11	79	7
	f	20	21	90	11	2	2	37	0
20	f/3	20	32	100	0	16	18	95	0
	2f/3	20	37	100	0	18	18	90	0
	f	20	32	100	0	4	6	50	0
30	f/3	20	47	100	0	27	25	95	0
	2f/3	20	48	100	0	28	25	95	0
	f	20	42	100	0	8	12	70	0
40	f/3	20	53	100	0	29	31	100	0
	2f/3	20	50	100	0	21	23	95	0
	f	20	47	100	0	9	12	85	0
60	f/3	20	62	100	0	40	42	100	0
	2f/3	20	56	100	0	27	30	100	0
	f	20	50	100	0	12	19	90	0
80	f/3	20	66	100	0	40	41	100	0
	2f/3	20	63	100	0	25	31	100	0
	f	20	58	100	0	12	23	100	0
100	f/3	20	69	100	0	43	45	100	0
	2f/3	20	64	100	0	30	34	100	0
	f	20	55	100	0	13	25	100	0
	total	420							

npg : nombre de problèmes générés pour le couple (n, m) correspondant ;
G1/ELS : la moyenne des gains de TLPT par rapport à l'heuristique ELS ;
PA1 : la proportion des expériences pour lesquelles une amélioration est apportée la proportion des expériences pour lesquelles une amélioration est apportée ;
PO1 : la proportion des expériences pour lesquelles la borne est atteinte parmi les expériences améliorées par TLPT ;
G1/ETF : la moyenne des gains de TLPT par rapport à l'heuristique ETF ;
G2/ETF : la moyenne des gains de TETF par rapport à l'heuristique ETF ;
PA2 : la proportion des expériences pour lesquelles une amélioration est apportée par rapport à l'heuristique ETF ;
PO2 : la proportion des expériences pour lesquelles la borne est atteinte parmi les expériences améliorées par TETF.

le cas de 100 tâches avec un nombre de processeurs égal au tiers du nombre de feuilles tandis que le gain moyen minimal est de 17 % obtenu pour 10 tâches et un nombre de processeurs égal au tiers du nombre de feuilles. Bien que ce gain soit relativement faible, il a conduit au moins 11 % des expérimentations à l'optimum. La mauvaise performance de ELS provient

du fait qu'elle ignore les délais de communication lors de la construction de l'ordonnancement contrairement à la méthode TLPT.

En ce qui concerne la méthode ETF, bien qu'elle présente une garantie de performance, nos deux versions tabou se comportent nettement mieux. Notre première version TLPT présente un gain par rapport à ETF qui est au moins égal à 9 % sauf dans le cas où le nombre de processeurs est égal à f où ce gain varie entre 2 et 13 %. Dans la version TETF, partant d'une solution initiale fournie par ETF, nous arrivons à améliorer les résultats d'un taux qui s'élève jusqu'à 45 % dans le cas de 100 tâches et $f/3$ processeurs. La proportion de cas où la borne inférieure est atteinte est de 7 %. Ce taux peu élevé peut s'expliquer par le fait que dans le cas de grands délais de communication, les bornes ne sont pas bonnes.

Nous présentons dans la figure 5 (respectivement figure 6) les courbes des gains de la méthode tabou par rapport à ELS (respectivement ETF) en fonction du nombre de tâches et du nombre de processeurs.

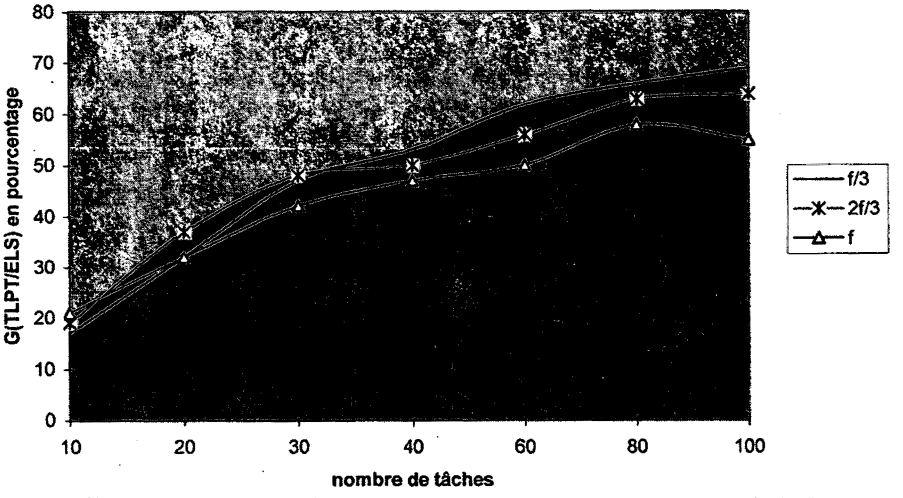


Figure 5. – Variation du gain de la méthode TLPT par rapport à ELS en fonction du nombre de tâches et du nombre de processeurs.

D'après les figures 5 et 6, quelque soit le nombre de processeurs, le gain de la méthode tabou par rapport aux heuristique ELS et ETF croit avec le nombre de tâches. D'autre part, ce gain décroît en fonction du nombre de processeurs. Ceci peut s'expliquer par le fait que notre voisinage qui essaye de regrouper les tâches critiques sur un même processeur a plus de chance d'améliorer la durée de l'ordonnancement lorsque m est faible. Cependant, nous devons noter que le nombre de processeurs n'est pas un facteur déterminant puisque

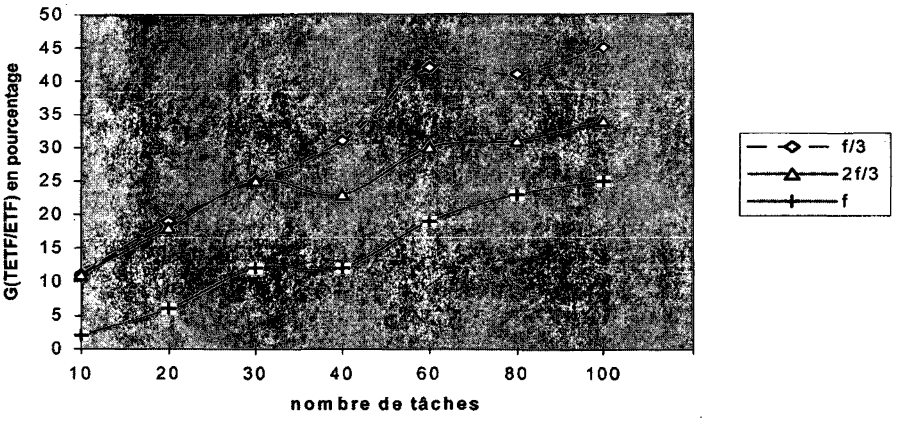


Figure 6. – Variation du gain de la méthode TETF par rapport à ETF en fonction du nombre de tâches et du nombre de processeurs.

d'une part le comportement des courbes reste le même pour les trois cas de m, d'autre part l'écart de la durée de l'ordonnancement pour un nombre fixé de tâches est faible lorsqu'on fait varier le nombre de processeurs. La performance de notre méthode provient du fait qu'elle travaille sur des solutions complètes contrairement aux approches gloutonnes à savoir les heuristiques ELS et ETF qui ne reviennent pas sur des décisions déjà prises.

b) Anti-arborescences avec petits délais de communication et durées constantes

Dans ce cas, les délais de communication sont inférieurs ou égaux aux durées des tâches. C_{ij} est une variable aléatoire discrète uniformément distribuée dans l'intervalle [1..10] tandis que P_i est fixé arbitrairement à 10. 420 problèmes sont générés de manière identique à celle du paragraphe précédent. La colonne 7 du tableau 2 indique la moyenne des écarts de la méthode TLPT par rapport à la borne tandis que la colonne 12 indique la moyenne des écarts de la méthode TETF par rapport à la borne. Nous rappelons que la borne considérée est le maximum entre la borne $\sum_{i=1}^{n} P_i/m$ et la durée de l'ordonnancement fourni par l'algorithme de Chrétienne (1989).

Nous synthétisons les résultats dans le tableau 2.

Remarque 2 : Les cases vides du tableau 2 indiquent que la méthode tabou n'a pas été appliquée car les solutions initiales obtenues par ETF sont déjà optimales.

Bien que les gains de TLPT par rapport ELS soient relativement faibles comparés au cas des grands délais de communication, il est important de

TABLEAU 2
Comparaison de TLPT et TETF avec les heuristiques ELS et ETF dans le cas de petits délais de communications

n	m	npg	G1/ELS	PA1	PO1	E1/B	G1/ETF	G2/ETF	PA2	PO2	E1/B
10	f/3	20	10	89	35	14	10	7	78	36	13
	2f/3	20	8	89	41	10	9	6	72	46	8
	f	20	8	83	87	1					
20	f/3	20	13	95	0	20	7	6	75	0	20
	2f/3	20	11	100	30	7	2	6	82	64	2
	f	20	11	100	80	1					
30	f/3	20	10	1000	5	24	3	7	85	6	18
	2f/3	20	11	100	20	5	3	5	90	28	3
	f	20	11	100	90	0.5					
40	f/3	20	11	100	0	21	2	5	85	0	16
	2f/3	20	12	100	30	5	1	5	88	40	2
	f	20	13	100	85	0.5					
60	f/3	20	9	100	0	17	2	6	90	6	2
	2f/3	20	11	95	37	4	0	2	70	67	1
	f	20	11	100	85	0.5					
80	f/3	20	10	100	0	15	1	6	95	11	7
	2f/3	20	12	100	5	3	0	3	100	43	1
	f	20	11	100	80	1					
100	f/3	20	11	90	0	13	1	6	100	5	5
	2f/3	20	13	95	21	3	1	3	89	62	1
	f	20	12	100	80	0.5					
	total	240									

npg : nombre de problèmes générés pour le couple (n,m) correspondant ;
G1/ELS : la moyenne des gains de TLPT par rapport à l'heuristique ELS ;
PA1 : la proportion des expériences pour lesquelles une amélioration est apportée la proportion des expériences pour lesquelles une amélioration est apportée ;
PO1 : la proportion des expériences pour lesquelles la borne est atteinte parmi les expériences améliorées par TLPT ;
E1/B : l'écart de TLPT par rapport à la borne ;
G1/ETF : la moyenne des gains de TLPT par rapport à l'heuristique ETF ;
G2/ETF : la moyenne des gains de TETF par rapport à l'heuristique ETF ;
PA2 : la proportion des expériences pour lesquelles une amélioration est apportée par rapport à l'heuristique ETF ;
PO2 : la proportion des expériences pour lesquelles la borne est atteinte parmi les expériences améliorées par TETF ;
E1/B : l'écart de TETF par rapport à la borne.

rappeler que dans presque 100 % des cas une amélioration est apportée par rapport à l'heuristique ELS. De plus, ces gains ont conduit à l'optimum dans des proportions qui varient entre 80 et 90 % pour un nombre de processeurs

égal au nombre de feuilles. De plus, la moyenne des écarts par rapport à la borne n'a pas dépassé dans ce cas le 1 %. Lorsque le nombre de processeurs est égal aux deux tiers du nombre de feuilles, la proportion des expériences pour lesquelles la borne est atteinte varie dans l'intervalle [20 %, 41 %] sauf pour le cas de 80 tâches où elle atteint 5 %. Le meilleur gain est de 13 % obtenu dans le cas de 20 tâches et $f/3$ processeurs, 40 tâches et f processeurs et 100 tâches et $2f/3$ processeurs. Les écarts par rapport à la borne décroissent avec l'augmentation du nombre de tâches dans le cas d'un nombre de processeurs égal aux deux tiers du nombre de feuilles et au nombre de feuilles. En ce qui concerne l'heuristique TETF, elle permet d'améliorer plus de 70 % des cas. Cette amélioration a conduit à la borne inférieure dans des proportions variant entre 28 % et 67 % pour un nombre de processeurs égal à $2f/3$. Même si la moyenne des gains est plus faible que celle de TLPT par rapport à ELS, la moyenne des écarts par rapport à la borne n'a pas dépassé les 3 % lorsque le nombre de processeurs est égal à $2f/3$, sauf pour le cas de 10 tâches. Lorsque le nombre de processeurs est égal au nombre de feuilles, ETF donne l'optimum dans la plupart des expérimentations, et par conséquent l'application de la méthode tabou est sans apport pour l'étude.

Comme le montre la figure 7, le comportement du gain de TLPT par rapport à l'heuristique ELS n'est pas stable. Ceci est dû à l'instabilité des solutions initiales fournies par ELS.

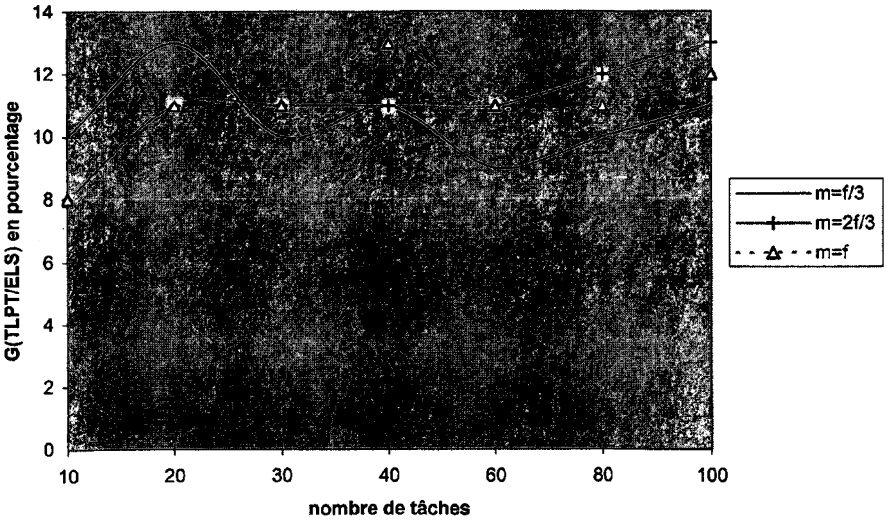


Figure 7. – Variation du gain de la méthode TLPT par rapport à ELS en fonction du nombre de tâches et du nombre de processeurs.

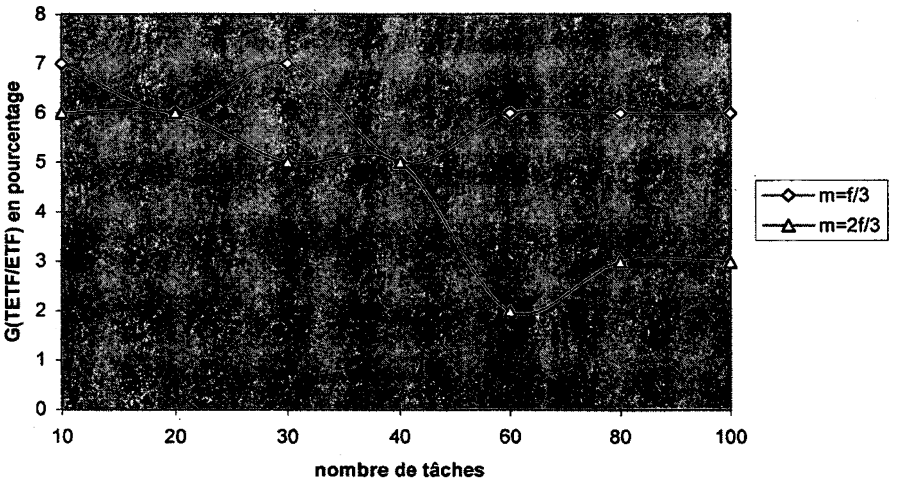


Figure 8. – Variation du gain de la méthode TETF par rapport à ETF en fonction du nombre de tâches et du nombre de processeurs.

La figure 8 montre que plus le nombre de tâches est élevé, plus le gain de TETF par rapport à ETF diminue. Ceci peut s'expliquer par le fait que l'effet des durées d'exécution induites par l'augmentation des tâches est plus important que celui des délais de communication pour ces instances avec petits délais. Ce gain reste une fonction décroissante du nombre de processeurs pour les mêmes raisons que dans le cas de grands délais de communication.

CONCLUSION

Dans cet article, nous avons proposé une nouvelle approche de résolution du problème d'ordonnancement de n tâches sur m processeurs identiques, avec délais de communication $P/Prec, P_j, C_{jk}/C_{\max}$. Notre approche modélise un ordonnancement par un graphe d'arbitrage ensuite le résout par une méthode tabou qui opère en deux phases. L'efficacité de notre approche provient de l'importance qu'on accorde aux délais de communication dans la définition du voisinage. La méthode proposée a été comparée avec les deux heuristiques ELS et ETF sur la base de problèmes générés aléatoirement. Les expérimentations ont montré que notre méthode est plus performante dans la plupart des cas. Des extensions de notre approche à des cas particuliers des problèmes d'ordonnancement tels que les processeurs hétérogènes, groupe de processeurs et bus feront l'objet de nos recherches ultérieures.

REMERCIEMENTS

Nous tenons à remercier Chams Lahlou et Ahmed Daddi-Moussa docteurs de l'Université de Paris VI de l'aide qu'ils nous ont apportée lors des travaux d'expérimentations. Nous remercions également les deux rapporteurs anonymes dont les commentaires ont permis d'améliorer certaines parties de ce papier.

RÉFÉRENCES

L. Alonso et R. Schott, *Random Generation of Trees*. Kluwer Academic Publishers (1995).

P. Chrétienne, A Polynomial Algorithm to Optimally Schedule Tasks on a Virtual Distributed System Under Tree-Like precedence constraints. *E.J.O.R.* **43** (1989) 225-230.

P. Chrétienne et C. Picouleau, *Scheduling with Communication Delays: A Survey, Scheduling Theory and its Applications*. John Wiley & Sons Ltd. (1995).

J.-Y. Colin et P. Chrétienne, CPM Scheduling With Small Communication Delays. *Oper. Res.* **39** (1995) 680-684.

A. Daddi-Moussa, *Méthode Exacte pour les Problèmes d'Ordonnancement avec Délais de Communication*. Thèse de Doctorat de l'Université de Paris VI (1997).

C. Hanen et A. Munier, *An Approximation Algorithm for Scheduling Dependent Tasks on m Processors With Small Communication Delays*. Rapport Technique, Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie (1997).

J.-J. Hwang, Y.-C. Chow, F.D. Angers et C.-Y. Lee, Scheduling Graphs in Systems with Interprocessor Communication Times. *SIAM. J. Comput.* **18** (1989) 244-257.

V.J.R. Smith, UET Scheduling With Unit Interprocessor Communication Delays. *Discrete Applied Math.* **18** (1987) 55-71.

D. Tayachi, K. Mellouli et P. Chrétienne, Modélisation du Problème $P_m/Prec/C_{max}$ et Résolution par une méthode Tabou, *Les Actes du Congrès des Deuxièmes Journées Francophones de Recherche Opérationnelle, FrancoroII*. Tunisie (1998).

B. Veltman, B.J. Lageweg et J.K. Lenstra, Multiprocessor Scheduling With Communication Delays. *Parallel Computing*. **16** (1990) 173-182.