COMMISSARIAT A L'ENERGIE ATOMIQUE
CENTRE D'ETUDES DE LIMEIL-VALENTON
DEPARTEMENT DE MATHEMATIQUES APPLIQUEES
94195 VILLENEUVE-SAINT-GEORGES CEDEX

CEA - N - 2738

Un algorithme rapide de partitionnement automatique de graphes.

Patrick Ciarlet et Françoise Lamour*

Septembre 1993

Ce travail a été réalisé dans le cadre du Projet Calcul Parallèle
du Département de Mathématiques Appliquées

* Cisi Ingénierie - sous contrat CEA/CEL-V/DMA

Résumé

Le problème abordé dans ce rapport est celui du partitionnement d'un graphe. Il s'agit de partitionner l'ensemble des sommets d'un graphe donné en sous-ensembles connexes de taille approximativement identique et tels que le nombre d'arêtes ayant leurs extrémités dans des sous-ensembles différents soit minimum. L'intérêt d'un tel partitionnement apparaît notamment dans le domaine du calcul parallèle lorsque les différents sous-ensembles sont affectés à des processeurs distincts d'une machine à mémoire distribuée. Nous proposons une heuristique efficace qui approche la solution de ce problème pour des graphes induits par des problèmes numériques.

Abstract

In this report we focus our interest on the graph partitioning problem. This problem consists in splitting the set of vertices of a given graph into balanced connected subsets. Moreover the overall number of edges which have their ends in differents subsets must be minimized. In particular such a partitioning is useful in the field of parallel computing. In this case different subsets are assigned to distincts processors of a distributed memory machine. We propose an efficient heuristic algorithm to approach the solution of this problem for graphs which come from numerical problems.

Table des Matières

Introduction

Le sujet du présent rapport est de présenter une méthode efficiente de partitionnement automatique de graphe. Le problème référencé sous le nom de *partitionnement de graphe* est le problème qui consiste à déterminer une partition de l'ensemble des sommets de ce graphe tout en optimisant un critère donné. Etant donné un graphe non orienté, notre préoccupation est d'une part de construire une partition équilibrée de l'ensemble de ses sommets, c'est-à-dire de partager cet ensemble en plusieurs sous-ensembles disjoints de taille approximativement identique, et d'autre part de minimiser le nombre d'arêtes ayant leurs extrémités dans des sous-ensembles différents. Nous savons bien sûr que ce problème est difficile (i.e. NP-complet voir par exemple [HyRi] pour une démonstration) et nous ne proposons ici qu'une heuristique simple et rapide qui tente d'approcher la solution de ce problème.

Le problème de partitionnement de graphe permet de formaliser de nombreux problèmes dès lors que les variables et les contraintes qui définissent le problème en question peuvent se résumer sous la forme d'un graphe. Notamment, dans le cadre du calcul parallèle, le partitionnement de graphe permet d'optimiser un calcul en exécutant concurremment les opérations définies sur les différents sous-ensembles. Etant donné une machine parallèle à p processeurs, il est intéressant de décomposer un graphe de tâches en p sous-graphes (ou en un multiple de p), chacun d'eux étant affecté à un processeur distinct. Evidemment, pour que le gain de l'optimisation soit réel il faut que les tailles des différents sous-graphes (i.e. le nombre de tâches par processeurs) soient à peu près équivalentes. De plus, sur une architecture à mémoire distribuée, il est absolument nécessaire de minimiser les communications entre processeurs.

Bien que le problème étudié soit d'intérêt beaucoup plus général, nos motivations sont de deux ordres : la parallélisation du produit matrice vecteur (pour des matrices creuses) et la définition de méthodes de décomposition de domaine parallèles. Le produit matrice vecteur joue, en effet, un rôle crucial lorsqu'il s'avère nécessaire de résoudre un problème linéaire par une méthode itérative. Ce produit bien optimisé sur les architectures vectorielles (c.f. [Erhe90]), a suscité de nouvelles réflexions lors de l'apparition des machines parallèles (voir [Laff93]). Les méthodes de décomposition de domaine sont également utilisées pour la résolution de problèmes linéaires par une méthode itérative. Elles visent à définir des préconditionneurs, c'est-à-dire des approximations de l'inverse du problème, en décomposant ce dernier en un ensemble de petits sous-problèmes que l'on peut résoudre localement. Ces préconditionneurs, qui sont soit algébriques (voir par exemple [CiaJ92] et [Meur91]) soit directs (voir [Smit91]), doivent eux-aussi être adaptés aux machines parallèles.

Quel que soit le problème considéré (produit matrice vecteur ou méthode de décomposition de domaine), la structure logique de ce problème peut se formaliser par un graphe dont l'ensemble des sommets est l'ensemble des points de discrétisation du domaine de calcul et l'ensemble des arêtes correspond aux éléments non nuls de la matrice A du système linéaire généré, c'est-à-dire qu'il existe une arête entre les sommets i et j du graphe si l'élément a_{ij} correspondant dans la matrice A est non nul.

Ainsi, il apparaît clairement pour ces deux problèmes que l'optimisation de la mise en œuvre

repose sur un "bon" partitionnement du graphe.

Les études sur le partitionnement automatique de graphe sont nombreuses. Parmi les articles les plus récents, on distingue aujourd'hui deux approches différentes à la résolution de ce problème.

La première approche est celle qui réunit les méthodes qui reposent sur un algorithme de type glouton, généralement simple à mettre en œuvre et raisonnablement rapide. Il en est ainsi dans [Farh88], [Farh89], et [FaLe93]. Dans ces trois articles C. Farhat propose le même algorithme simple de partitionnement de maillage qui consiste à construire récursivement les sous-ensembles de la partition en accumulant pour chacun d'eux le nombre requis de sommets et en donnant la priorité à des sommets d'un certain type. Dans le deuxième article, l'auteur associe à la décomposition de son maillage une méthode de placement pour une architecture massivement parallèle. Enfin, dans son troisième article, il démontre par comparaison avec des algorithmes gloutons de partitionnement basés sur des critères physiques, par exemple un algorithme récursif de partitionnement basé sur les moments d'inertie de l'objet considéré, que son algorithme est optimal.

La seconde approche, beaucoup moins intuitive, met en œuvre des algorithmes algébriques de partage de graphes complétés par des méthodes d'optimisation issues de la théorie de graphes. Les auteurs de [PoSL90] et [Simo91] proposent ainsi un algorithme basé sur le calcul d'une valeur propre de la matrice d'adjacence du graphe considéré. Des résultats théoriques sur les propriétés des valeurs propres et surtout de leurs vecteurs propres associés permettent de diviser, par 2, récursivement les sommets du graphe en 2^p sous-ensembles équilibrés. La minimisation du nombre des arêtes inter sous-ensembles est traitée localement. Les auteurs de [BaSi92] optimisent le calcul de la valeur propre et réduisent ainsi notablement la durée de l'algorithme de partitionnement. Enfin, l'algorithme décrit dans [HeLe92] est à peu près semblable aux précédents, si ce n'est que plusieurs valeurs propres sont simultanément calculées afin de diviser directement par 4 ou par 8 l'ensemble des sommets du graphe.

Il n'existe à ce jour aucune comparaison faite entre ces deux classes de méthodes. Remarquons tout de même que la première approche, quoique naive, est beaucoup plus rapide et qu'elle présente l'avantage d'un partitionnement en un nombre quelconque de sous-ensembles. A l'inverse, la seconde approche semble mieux fondée quant à la minimisation du nombre d'arêtes inter sous-ensembles et à l'aspect des sous-ensembles créés (on est toujours assuré de la connexité des divers sous-ensembles).

La méthode que nous proposons appartient à la première école et s'inspire des travaux de C. Farhat. Cependant notre démarche diffère en plusieurs points qui, nous semble t-il, conduisent à une meilleure optimisation du partitionnement. Ces points seront mis en évidence lors de la présentation détaillée de notre algorithme.

A la suite de cette introdution, ce rapport est organisé en 5 parties. La première partie présente les notations utilisées et les principes de la méthode proposée. L'algorithme de partitionnement est décrit dans la deuxième partie. Dans la troisième partie, nous détaillons quelques optimisations pratiques concernant la mise en œuvre et nous proposons une éventuelle version

parallèle de l'algorithme. Des exemples de partitionnement réalisés et des résultats de performances mesurées sont exposés dans la partie 4. Enfin, nous concluons en avançant quelques idées pour des optimisations possibles de notre méthode de partitionnement.

1 Notations et principes

Rappelons brièvement les principales définitions et notations utilisées pour la suite de l'exposé.

Le degré d'un sommet s d'un graphe $G = (S, E)$, noté $d(s)$ est le nombre d'arêtes de E ayant une extrémité en s. On dit aussi que $d(s)$ est le nombre de voisins de s. On dit d'un graphe G qu'il est de *degré maximum* d si tous ses sommets sont de degré inférieur ou égal à d.

Le problème qui consiste à partitionner un graphe $G = (S, E)$ en p sous-ensembles, de sorte que les différents sous-ensembles soient de taille équilibrée et que le nombre d'arêtes ayant leurs extrémités dans des sous-ensembles distincts soit le plus petit possible s'écrit formellement comme suit :

Déterminer $S_1, S_2, ..., S_p$ tels que :

(a) $S_i \bigcap S_j = \emptyset$ pour $i \neq j$, et $\bigcup_{i=1}^{p} S_i = S$,

(b) $cardinal(S_i) \simeq cardinal(S_j)$ pour $1 \leq i, j \leq p$ et $i \neq j$,

(c) $cardinal(\{e = (s_i, s_j) \mid e \in E \text{ et } s_i \in S_i,\ s_j \in S_j \text{ avec } i \neq j\})$ est le plus petit possible.

Le *graphe induit* par la partition $S_1, S_2, ..., S_p$ est alors le graphe dont l'ensemble des sommets est $\{1, 2, ..., p\}$ et tel qu'il existe une arête entre un sommet i et un sommet j, s'il existe, au moins, une arête entre un sommet de S_i et un sommet de S_j.

Rappelons que le graphe que nous considérons est celui engendré par l'ensemble des points de discrétisation du domaine de calcul (sommets) et par les élements non nuls de la matrice du système linéaire à résoudre (arêtes). C. Farhat considère quant à lui uniquement des discrétisations par éléments finis. De plus le graphe qu'il partitionne a pour sommets les éléments du maillage; les arêtes relient les éléments ayant un côté en commun. Dans la suite, le mot graphe fait donc toujours référence à cette définition. De plus, par abus de langage, nous appelons *frontière* du graphe l'ensemble des sommets qui se trouvent effectivement sur la frontière du domaine discrétisé.

Si la construction d'un partitionnement qui respecte la condition (b), c'est-à-dire dont les sous-ensembles possédent à peu près le même nombre de sommets, est facilement réalisable, il n'en est pas de même pour la condition (c). Quelle que soit la partition retenue, les sous-ensembles de sommets engendrés sont liés entre eux par l'intermédiaire de certains sommets : les sommets de l'*interface*. Ces sommets ont la particularité d'avoir certains de leurs voisins dans des sous-ensembles différents. Pour respecter la condition (c), c'est ce nombre de voisins

qu'il faut minimiser. L'objectif à réaliser est donc de choisir pour l'interface, lorsque cela est possible, des sommets de faible degré. De plus, en réduisant le nombre de sommets des interfaces on peut espérer aller dans le sens de la dernière condition requise.

La méthode que nous proposons est basée sur ces observations et tente donc d'intégrer les trois principes suivants : (1) équipotence des sous-ensembles, (2) minimisation du nombre des sommets d'interface et (3) minimisation du degré maximum du graphe induit. Dans le cadre d'une mise en œuvre parallèle des calculs sous-jacents, le principe (1) assure l'équilibrage de la charge de travail sur les différents processeurs tandis que les principes (2) et (3) garantissent la réduction des synchronisations et des communications entre les processeurs.

Nous proposons de plus, une version parallèle de notre méthode de partitionnement. Dans ce cas, les différents sous-ensembles sont construits simultanément.

2 L'algorithme séquentiel

Soit G un graphe à partitionner en n_p sous-ensembles tout en respectant les trois principes précédemment énoncés, nous présentons dans ce paragraphe un algorithme qui calcule successivement ces différents sous-ensembles.

Soit N le nombre de sommets de G, on note n_s le nombre de sommets souhaités par sous-ensemble, n_s étant défini par sa valeur $\frac{N}{n_p}$. Le partitionnement du graphe G s'effectue en construisant, successivement, les différents sous-ensembles de sommets, eux-mêmes construits de manière séquentielle. Chaque sous-ensemble S_k, pour $1 \leq k \leq n_p$, est déterminé au moyen d'un processus itératif qui sélectionne dans le graphe le nombre désiré de sommets (nombre aussi proche que possible de n_s). Chaque sommet est alors *marqué* d'un numéro (de 1 à n_p) qui l'identifie au sous-ensemble considéré. Ainsi, on appelle *degré courant* d'un sommet le nombre de ces voisins non marqués à l'étape courante. On étend de même la notion de frontière de la manière suivante : on appelle *frontière courante* du graphe l'ensemble des sommets non marqués de la frontière. Dans le cas où tous les sommets de la frontière sont marqués, la frontière courante est alors définie comme l'ensemble des sommets non marqués qui sont voisins de sommets marqués. La figure 1 explique les deux définitions de la frontière courante.

La procédure de construction d'un sous-ensemble se compose de 2 étapes. La première étape correspond à l'initialisation du procédé itératif par le choix d'un sommet particulier. La deuxième étape met effectivement en œuvre le processus récursif d'accumulation et de marquage de sommets. L'ensemble, $V(i)$, des sommets sélectionnés à l'étape i est l'ensemble des voisins, non encore marqués, de ceux choisis à l'étape $i-1$. Plus précisément, le choix du sommet initial n'est pas arbitraire. Le sommet de départ doit vérifier, dans cet ordre, les trois conditions suivantes :

- (a) appartenir à la frontière courante,
- (b) être le voisin d'un sommet du sous-ensemble (s'il existe) précédemment construit,

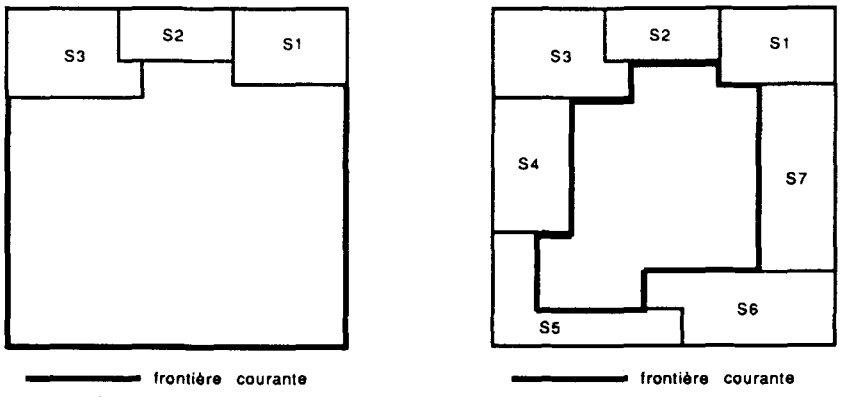

Figure 1: Illustration de la notion de frontière courante.

- (c) être de degré courant minimal.

Si plusieurs sommets vérifient ces trois conditions, nous choisissons arbitrairement un de ces sommets. Les conditions initiales (a) et (b) assurent la régularité des différents sous-ensembles : du fait de la définition de la frontière courante, les sous-ensembles vont progresser tout au long de la frontière, dans un mouvement concentrique. De plus, intuitivement, la condition (c) contribue à la minimisation du nombre de sommets de l'interface.

Enfin, au cours de la phase itérative d'accumulation des sommets, si on constate que l'ajout de tous les sommets voisins, potentiellement candidats, grossit excessivement la taille du sous-ensemble final, on choisit de compléter le sous-ensemble précédemment déterminé en retenant seulement le nombre prescrit de sommets (n_s) et en privilégiant ceux de degré courant minimal.

Cette dernière propriété va dans le sens de la minimisation du degré maximum du graphe induit. En effet, les sommets choisis à la dernière itération sont les sommets de l'interface. S'ils sont de degré courant minimal ils ont, par définition, le moins de voisins possibles à l'extérieur du sous-ensemble considéré, ce qui conduit nécessairement à minimiser le degré maximum du graphe induit.

La procédure globale de partitionnement est réalisée en itérant le processus décrit de construction d'un sous-ensemble.

Par rapport à l'algorithme proposé par C. Farhat et qui a inspiré nos travaux, notre méthode diffère principalement en 3 points. D'une part nous considérons directement le graphe de la matrice du problème et non celui du maillage. D'autre part, les conditions que nous imposons à nos sommets de départ, pour l'initialisation de la construction de chaque sous-ensemble, sont beaucoup plus fortes et assurent ainsi de bonnes propriétés à notre partitionnement. Enfin, les optimisations de notre algorithme de base, que nous développons dans le paragraphe suivant, tentent de pallier aux anomalies inhérentes aux méthodes heuristiques alors qu'aucune proposition à ce sujet n'est faite dans les articles de référence.

Nous donnons ci-après en pseudocode, une description de la procédure de partitionnement d'un graphe G.

Partitionnement_sequentiel(G)

1. début
2. $k = 1$;
3. initialiser la frontière courante à la frontière de G;
4. **tant que** $k < n_p$ **faire**
5. **début**
6. **si** la frontière courante est vide **alors** la réinitialiser[1];
7. $i = 1$;
8. choisir un sommet s de G vérifiant :
(a) s appartient à la frontière courante de G;
(b) si $k > 1$, s est voisin d'un sommet marqué à l'étape $k - 1$;
(c) s est de degré courant minimal;
9. marquer s;
10. $S_k(i) = \{s\}$;
11. **tant que** $cardinal(S_k(i)) < n_s$ **faire**
12. **début**
13. $i = i + 1$;
14. construire $V(i)$;
15. **si** $cardinal(V(i)) \leq n_s - cardinal(S_k(i-1))$ **alors**
16. **début**
17. marquer les sommets de $V(i)$;
18. $S_k(i) = S_k(i-1) \bigcup V(i)$;
19. mise à jour de la frontière courante;
20. mise à jour des degrés courants;
21. **fin**
22. **sinon**
23. **début**
24. marquer $(n_s - cardinal(S_k(i-1)))$ sommets de degré courant minimal dans $V(i)$;
25. $S_k(i) = S_k(i-1) \bigcup \{les\ sommets\ marqués\ de\ V(i)\}$;
26. mise à jour de la frontière courante;
27. mise à jour des degrés courants;
28. **fin**
29. **fin**
30. $S_k = S_k(i)$;
31. $k = k + 1$;
32. **fin**
33. marquer les sommets restants;
34. accumuler ces sommets dans S_{n_p};
35. fin

[1] c'est-à-dire que la frontière courante est redéfinie comme l'ensemble des sommets non marqués et voisins de sommets marqués.

3 Quelques optimisations de l'algorithme

3.1 Connexité d'un sous-ensemble

Il peut s'avérer au cours de la construction itérative d'un sous-ensemble que celle-ci s'arrête faute de candidats, c'est-à-dire faute de voisins non encore marqués, et ce avant d'avoir atteint la taille souhaitée. Il serait alors possible de poursuivre la construction du sous-ensemble en cours à partir d'un nouveau sommet. Cependant cette solution n'assure pas la connexité du sous-ensemble. Afin de préserver cette propriété, le traitement particulier suivant est mis en œuvre : l'ensemble des sommets déja construit est réparti sur les sous-ensembles avoisinants suivant la règle qu'un sommet est affecté au sous-ensemble avec lequel il partage le plus grand nombre de voisins.

Dans l'algorithme présenté page 6, la détermination du dernier sous-ensemble se fait trivialement en affectant à ce dernier tous les sommets non encore marqués. De ce fait, nous ne pouvons aucunement être assurés de la connexité de ce sous-ensemble. Pour remédier à cela, nous avons affiner le processus de construction du dernier sous-ensemble. Nous avons ainsi choisi, dans le cas où ce dernier est constitué de plusieurs composantes connexes, de réaffecter celles de plus petites tailles à d'autres sous-ensembles suivant le même procédé que décrit précédemment. Lorsque les différentes composantes sont de tailles équivalentes nous acceptons alors de perdre la propriété de connexité pour le dernier sous-ensemble. D'après les tests réalisés, ce cas de figure ne se produit que très rarement, et uniquement pour un partitionnement en petits sous-ensembles (faible nombre de sommets).

Remarque
Pour assurer la connexité du dernier sous-ensemble on peut envisager de ne conserver, dans tous les cas, qu'une seule des composantes. Afin de ne pas privilégier la connexité au détriment de l'équilibrage des sous-ensembles, on peut alors compléter la composante retenue par des sommets récupérés dans les sous-ensembles avoisinants.

3.2 Equilibrage de la taille d'un sous-ensemble

Nous avons appelé n_s le nombre de sommets souhaités par sous-ensemble. Ce nombre est défini comme étant le nombre total de sommets du graphe, N, divisé par le nombre de sous-ensembles désirés, n_p. La division de N par n_p n'étant pas nécessairement entière, il se produit inévitablement un déséquilibre de la taille des sous-ensembles au long des itérations de l'algorithme de partitionnement. Pour prévenir ce phénomène, nous avons donc réestimé la

valeur de n_s à chaque initialisation du procédé de construction d'un nouveau sous-ensemble suivant la formule :

$$n_s = \frac{N - \textit{nbre de sommets déjà marqués}}{n_p - \textit{nbre de sous ensembles déjà créés}}.$$

De plus, cette réévaluation prend ainsi en compte la réaffectation des sommets pour des raisons de connexité évoquée dans le paragraphe précédent.

3.3 Une version parallèle

Une optimisation possible de la méthode de partitionnement de graphe est la parallélisation du procédé de construction des différents sous-ensembles. Contrairement aux optimisations précédemment décrites cette optimisation n'a pas encore été mise en œuvre.

Une parallélisation de la méthode de partitionnement peut consister en la détermination simultanée de différents sous-ensembles. Un tel algorithme parallèle de partitionnement de graphe repose alors, essentiellement, sur le choix des différents sommets qui initialisent le processus accrétif de construction des différents sous-ensembles. Concrètement, pour éviter tout recouvrement des sous-ensembles entre eux, il faut que les points de départ soient suffisamment éloignés les uns des autres. Pour ce faire, il faut estimer le nombre maximum d'itérations qui conduit à la détermination complète d'un sous-ensemble.

Considérons le cas d'un graphe régulier infini de degré d (tout sommet possède exactement d voisins) et obtenu par pavage du plan. Si l'on appelle s_i le nombre de nouveaux sommets atteints lors de la $i$ème étape de construction, on a (le résultat a été démontré pour un degré valant 3, 4 et 6) :

$$s_i = d \cdot i.$$

Le nombre de sommets n_l après l étapes est donc :

$$n_l = 1 + \sum_{i=1}^{i=l} s_i = 1 + \frac{1}{2} d \cdot l(l+1).$$

Ainsi, après l étapes, au moins $\frac{1}{2} d \cdot l^2$ sommets sont accumulés. Comme on veut n_s sommets par sous-ensemble, $l_{max} = E(\sqrt{\frac{2n_s}{d}}) + 1$ est un majorant du nombre maximum d'itérations.

Si on part d'un sommet situé sur la frontière d'un graphe régulier, on obtient, en raisonnant de la même façon, un minorant de la forme $\frac{1}{4} d \cdot l^2$. Dans ce cas, $l'_{max} = E(\sqrt{\frac{4n_s}{d}}) + 1$ est un majorant du nombre maximum d'itérations.

Remarque

Dans le cas d'un graphe régulier infini de degré 6 obtenu par pavage de l'espace (pavage de l'espace par des cubes), on a le résultat suivant :

$$s_i = 4i^2 + 2.$$

et

$$n_l = 1 + 2l + \frac{2}{3}l(l+1)(2l+1) \simeq \frac{4}{3}l^3.$$

L'estimation que nous proposons, dans le cas d'un graphe quelconque, est une approximation empirique du résultat énoncé plus haut. Dans ce cas, c'est le *degré moyen* du graphe (c'est-à-dire le degré moyen d'un sommet du graphe) qui est pris en considération et le nombre maximum d'itérations nécessaire pour la construction d'un sous-ensemble est : $l^f_{max} = E(\sqrt{\frac{4n_s}{d_{moy}}}) + 1$.

Ainsi la méthode parallèle de partitionnement proposée s'articule autour de deux points. D'une part, la première phase du processus de partitionnement consiste en la détermination de plusieurs sommets de départ. Ces sommets doivent être distants deux à deux, au minimum, d'une longueur l^f_{max}, c'est-à-dire que tout chemin reliant deux de ces sommets doit être constitué d'au moins l^f_{max} arêtes. D'autre part, la seconde phase met en œuvre le processus de partitionnement, séquentiel, qui calcule successivement en chacun des sommets de départ les sous-ensembles formant une partie de la partition finale.

Pour ce qui des sommets de départ respectant la distance imposée, des algorithmes de chemins classiques de complexité polynomiale, tels que ceux proposés dans [Berg87] ou [GoMi85], doivent permettre de les déterminer.

4 Exemples et résultats

Nous présentons ici cinq exemples de graphes : les deux premiers sont obtenus par une discrétisation par différences finies pour un schéma à 5 points ou un schéma à 9 points, le troisième graphe offre un exemple qui mèle une discrétisation par éléments finis P1 et Q1, le quatrième exemple est celui d'un domaine compris entre 2 sphères de rayons respectifs 1 et 10 et discrétisé par des éléments finis P1, enfin le dernier exemple est une représentation de la discrétisation par éléments finis P1 de l'extérieur d'un profil d'aile d'avion.

Exemple 1

La Figure 2 représente un carré discrétisé par différences finies pour le schéma à 5 points avec 10.000 sommets que l'on a partitionné en 50 sous-ensembles. Le tableau 1 récapitule les différents paramètres du partitionnement pour 10.000 sommets : le nombre de sous-ensembles (n_p), le nombre moyen de sommets par sous-ensemble ($n_{s_{moyen}}$), le pourcentage moyen de sommets d'interface ($n_{s_{inter}}$), le pourcentage moyen des arêtes d'interface ($n_{a_{inter}}$), le degré moyen du graphe induit ($d_{G^i_{moyen}}$), le nombre de composantes connexes du dernier sous-ensemble (n_{cc}), et enfin le temps en seconde du partitionnement sur une Sun SparcStation IPX (T_p). Pour les valeurs moyennes, les valeurs minimales et maximales sont précisées.

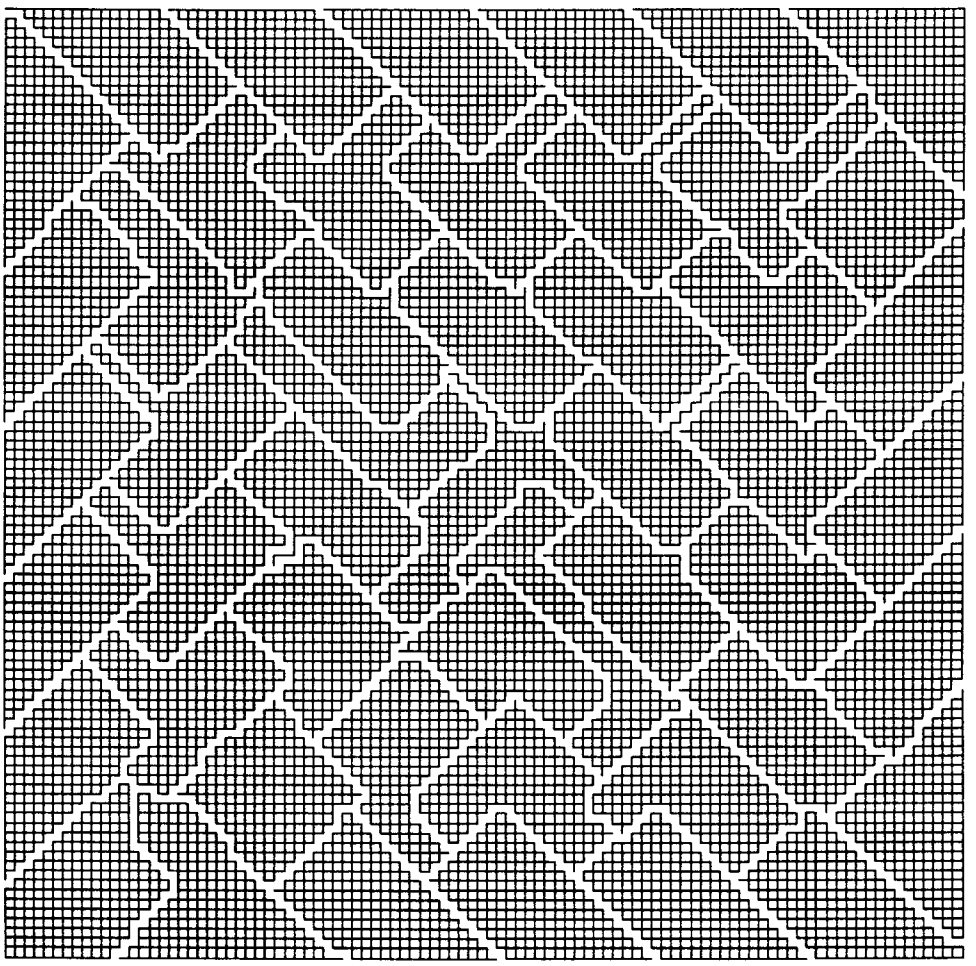

Figure 2: Carré de 10.000 sommets en 50 sous-ensembles.

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
5	2000	$4\ {}^{5}_{3}$	$1\ {}^{2}_{1}$	$2\ {}^{4}_{2}$	1	0.60
15	$666\ {}^{667}_{666}$	$9\ {}^{18}_{5}$	$4\ {}^{9}_{2}$	$4\ {}^{8}_{3}$	1	0.59
50	$200\ {}^{215}_{193}$	$20\ {}^{40}_{10}$	$10\ {}^{22}_{5}$	$4\ {}^{9}_{3}$	3	0.70
90	$111\ {}^{121}_{106}$	$27\ {}^{58}_{13}$	$15\ {}^{36}_{7}$	$5\ {}^{13}_{3}$	1	0.74

Table 1: Partitionnement pour 10.000 sommets.

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
50	$200\ {}^{215}_{186}$	$20\ {}^{37}_{10}$	$10\ {}^{20}_{5}$	$4\ {}^{9}_{3}$	1	0.72

Table 2: Avec traitement optimisé du dernier sous-ensemble.

On constate qu'en termes de pourcentage de sommets et d'arêtes d'interface, les meilleurs résultats sont obtenus pour un partitionnement en un petit nombre de sous-ensembles. C'est en effet dans ce cas que le rapport périmètre/surface est le plus petit. Pour une partition en 50 sous-ensembles, le dernier sous-ensemble n'est pas connexe mais est en trois parties. L'optimisation du traitement du dernier sous-ensemble permet une partition avec un dernier sous-ensemble connexe (voir Table 2). Bien que les valeurs moyennes ne soient pas modifiées par cette optimisation trop locale, les valeurs maximales du pourcentage de sommets et d'arêtes d'interface sont inférieures dans ce cas. Enfin, on observe que le degré moyen du graphe induit reste du même ordre de grandeur que celui du graphe d'origine malgré certains pics qui apparaissent pour une partition en un grand nombre de sous-ensembles et dans le cas de composantes multiples du dernier sous-ensemble. L'augmentation de ce degré en fonction du nombre de sous-ensembles s'explique par le fait que la méthode que nous avons mise en œuvre construit de manière concentrique les différents sous-ensembles. De ce fait, un partitionnement à grand nombre de sous-ensembles produit plusieurs couronnes de sous-ensembles et les sous-ensembles des couronnes les plus internes sont ceux qui ont nécessairement le plus de voisins.

Les temps de construction de la partition sont assez négligeables. Pour un nombre de sommets fixé à 10.000, le temps de partitionnement varie très peu en fonction du nombre de sous-ensembles et reste bien inférieur à 1 seconde pour une partition en 90 sous-ensembles. De même, pour un nombre de sous-ensembles fixé (ici 32), la variation du temps de partitionnement est sous-linéaire en fonction du nombre de sommets du graphe (voir Table 3). Les lignes précédées d'une flèche dans la Table 3 indiquent que l'on a choisi pour le calcul du dernier sous-ensemble la version optimisée. On constate que les temps de calcul ne varient pratiquement pas avec ou sans cette option.

N	n_{cc}	T_p (s)
2500	1	0.28
4096	3	0.39
→ 4096	1	0.43
6400	2	0.50
→ 6400	1	0.50
8100	1	0.58
10000	1	0.63
14000	2	0.75
→ 14000	1	0.75
19600	2	0.82
→ 19600	1	0.84

Table 3: T_p en fonction de N pour 32 sous-ensembles.

Exemple 2

Cet exemple est donné afin de montrer que dans les cas simples notre algorithme de partitionnement donne les résultats attendus : dans les trois cas les sous-ensembles sont totalement équilibrés et de forme carrée (voir Figure 3).

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
4	1024	6	2	3	1	0.49
64	64	$38\ \frac{63}{23}$	$18\ \frac{21}{10}$	$6\ \frac{3}{8}$	1	0.52
256	16	$71\ \frac{75}{43}$	$48\ \frac{52}{25}$	$7\ \frac{8}{3}$	1	0.63

Table 4: Carré de 4096 sommets.

Remarque

Comme nous discrétisons ici un carré, le nombre de sommets du graphe (N) est toujours de la forme n^2. Si on choisit de partitionner en n_p sous-ensembles où n_p est de la forme q^2 et q divisant n, on obtient nécessairement en utilisant notre algorithme des sous-ensembles carrés et équilibrés.

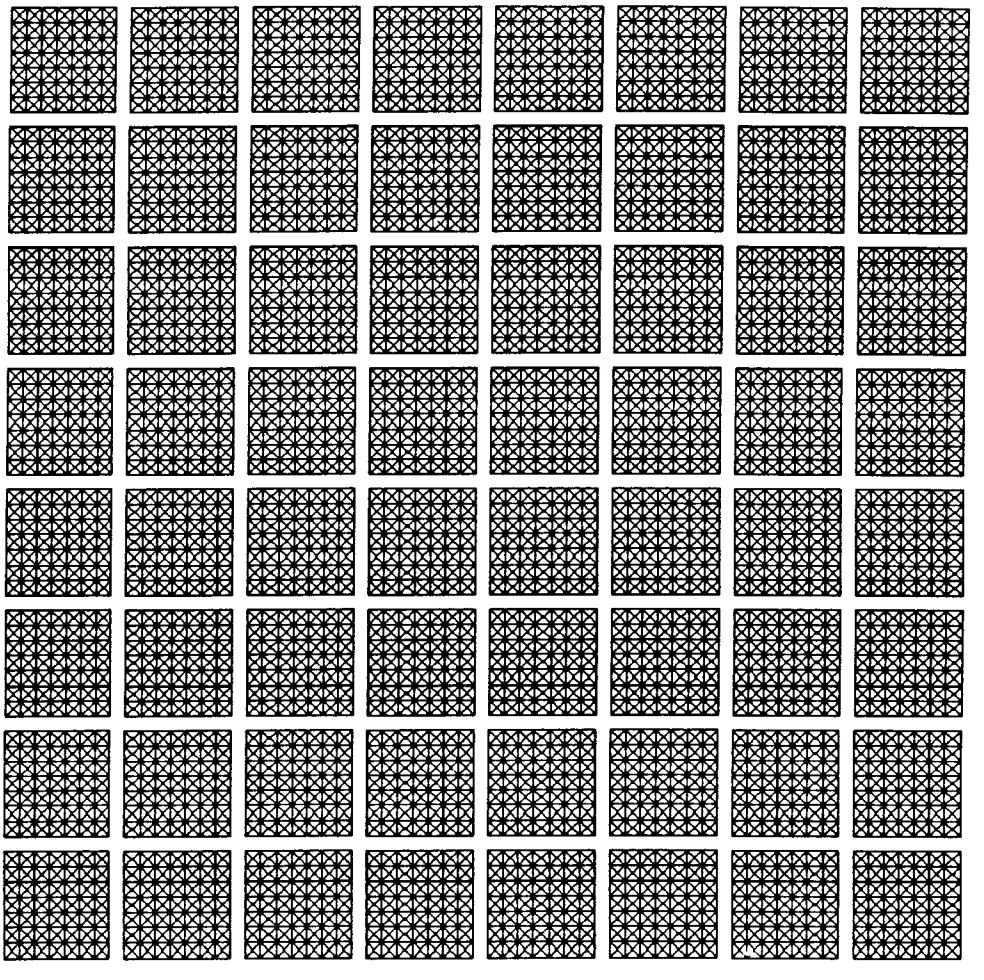

Figure 3: Carré de 4096 sommets en 64 sous-ensembles.

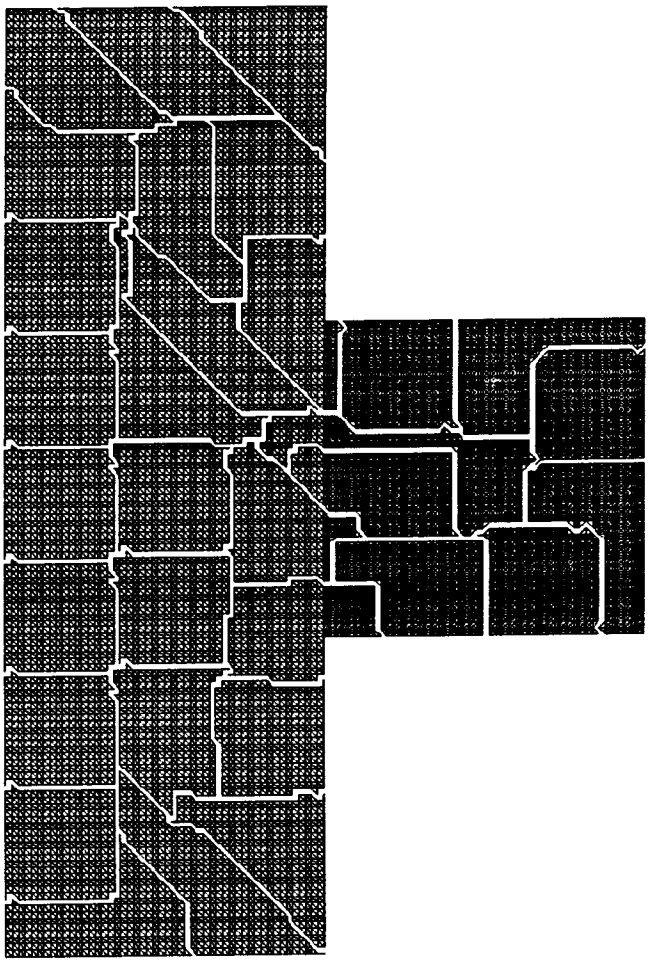

Figure 4: T de 10.251 sommets en 32 sous-ensembles.

Exemple 3

Il s'agit d'un T constitué d'un sous-domaine discrétisé par des éléments finis P1 et d'un autre discrétisé par des éléments finis Q1. La première discrétisation induit un sous-graphe de degré 6 alors que la seconde induit un sous-graphe de degré 8. Les résultats résumés dans la Table 5 portent sur un T de 10.251 sommets et dont le degré moyen est 6 (voir Figure 4 pour une partition en 50 sous-ensembles). Conformément à ce que l'on pouvait attendre, les pourcentages de sommets et d'arêtes d'interface ainsi que le degré moyen du graphe induit augmentent en fonction du nombre de sous-ensembles. Cependant, compte-tenu de l'exemple, ces valeurs demeurent tout à fait acceptables.

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
4	$2562\ {}^{2563}_{2562}$	$4\ {}^{6}_{3}$	$1\ {}^{2}_{1}$	$2\ {}^{3}_{2}$	2 *	0.70
8	$1281\ {}^{1282}_{1281}$	$7\ {}^{10}_{4}$	$2\ {}^{3}_{1}$	$3\ {}^{5}_{2}$	1	0.69
16	$640\ {}^{641}_{640}$	$13\ {}^{27}_{5}$	$4\ {}^{22}_{11}$	$3\ {}^{10}_{2}$	1	0.69
32	$320\ {}^{365}_{302}$	$19\ {}^{42}_{8}$	$7\ {}^{17}_{3}$	$4\ {}^{11}_{2}$	1	0.70
40	$256\ {}^{257}_{256}$	$21\ {}^{38}_{9}$	$8\ {}^{15}_{3}$	$4\ {}^{9}_{2}$	1	0.73
64	$160\ {}^{204}_{151}$	$28\ {}^{71}_{12}$	$11\ {}^{36}_{4}$	$5\ {}^{16}_{2}$	3*	0.75
128	$80\ {}^{112}_{55}$	$39\ {}^{58}_{17}$	$17\ {}^{28}_{7}$	$5\ {}^{9}_{2}$	2*	0.80

Table 5: T de degré moyen 6 et 10251 sommets.

Les valeurs de n_{cc} repérées par une * passent respectivement à 2, 2 et 1 dans le cas d'un traitement optimisé du dernier sous-ensemble.

Exemple 4

Cet exemple présente un domaine compris entre deux sphères avec 9020 sommets et un degré moyen égal à 14. On observe que sur ce cas, les temps de partitionnement sont particulièrement peu élevés et ne dépassent pas le $1/10^e$ de seconde.

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
2	4510	15	4	1	1	0.02
4	2255	$27\ ^{31}_{19}$	$8\ ^{10}_{16}$	3	1	0.02
8	$1127\ ^{1128}_{1127}$	$43\ ^{54}_{26}$	$15\ ^{20}_{8}$	$6\ ^{7}_{5}$	1	0.02
10	902	$44\ ^{53}_{25}$	$16\ ^{21}_{8}$	$7\ ^{8}_{6}$	1	0.02
20	451	$58\ ^{69}_{39}$	$24\ ^{32}_{13}$	$9\ ^{12}_{6}$	1	0.03
40	$225\ ^{232}_{196}$	$67\ ^{96}_{40}$	$33\ ^{94}_{16}$	$11\ ^{28}_{6}$	2	0.04
80	$112\ ^{116}_{108}$	$82\ ^{100}_{58}$	$51\ ^{104}_{28}$	$12\ ^{23}_{6}$	2	0.06

Table 6: Sphère de degré moyen 14 et 9020 sommets, traitement optimisé.

Les pourcentages $n_{s_{inter}}$ et $n_{a_{inter}}$ sont élevés à cause du degré moyen élevé du graphe.

Exemple 5

La figure 5 représente un partitionnement en 16 sous-ensembles pour le graphe représentatif de l'extérieur d'un profil d'aile d'avion avec 4253 sommets. Le degré moyen de ce graphe est 6 (le degré minimum est 3 et le degré maximum est 9). La particularité de ce graphe est de présenter 3 trous. Les résultats des différents partitionnement effectués sont résumés dans la Table 7. On constate d'une part que l'irrégularité du graphe ne pose aucun problème quant à son partitionnement et d'autre part que celui-ci se réalise en un temps de l'ordre de $1/10^e$ de seconde. Comme pour les exemples précédents on observe que les performances en termes de pourcentages de sommets et d'arêtes d'interface s'altèrent dès lors que la taille des sous-ensembles est petite.

n_p	$n_{s_{moyen}}$	$n_{s_{inter}}$ (%)	$n_{a_{inter}}$ (%)	$d_{G^i_{moyen}}$	n_{cc}	T_p (s)
4	$1063\ ^{1064}_{1063}$	$11\ ^{13}_{9}$	$3\ ^{4}_{3}$	3	1	0.10
16	$265\ ^{363}_{245}$	$20\ ^{26}_{12}$	$7\ ^{10}_{4}$	$4\ ^{7}_{3}$	2*	0.10
32	$132\ ^{234}_{124}$	$28\ ^{44}_{14}$	$11\ ^{19}_{5}$	$4\ ^{7}_{3}$	3*	0.11
50	$85\ ^{105}_{83}$	$36\ ^{56}_{27}$	$15\ ^{27}_{10}$	$4\ ^{7}_{2}$	4*	0.14
128	$33\ ^{63}_{25}$	$58\ ^{96}_{36}$	$30\ ^{74}_{16}$	$4\ ^{9}_{2}$	2	0.20

Table 7: Profil d'aile de degré moyen 6 et 4253 sommets.

Les valeurs de n_{cc} repérées par une * passent toutes à 1 dans le cas d'un traitement optimisé du dernier sous-ensemble.

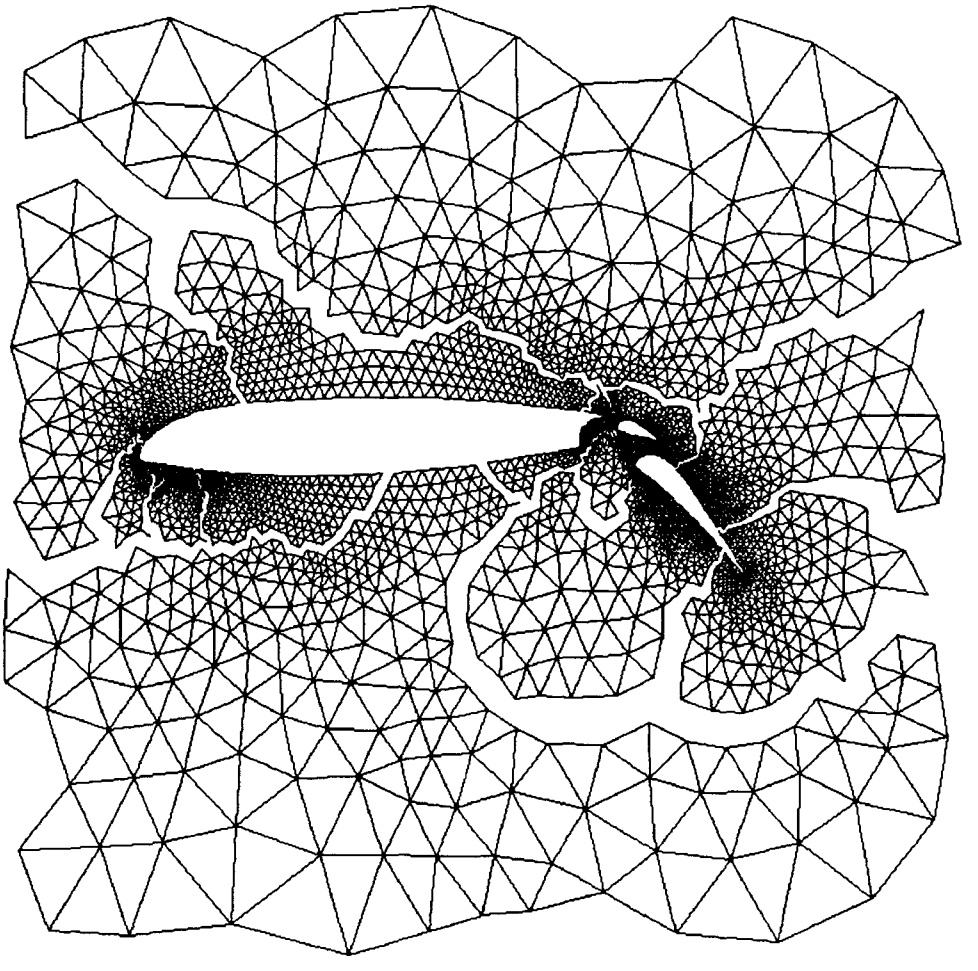

Figure 5: Profil d'aile de 4253 sommets en 16 sous-ensembles.

Conclusion

La méthode de partitionnement proposée offre de manière rapide une décomposition constituée de sous-ensembles de taille équilibrée et qui ne partagent pas trop de sommets entre eux. C'est dans ce sens une bonne méthode générale de partitionnement de graphe. Bien entendu, cette méthode est perfectible et ce dans plusieurs directions. Cependant, à ce niveau aucune optimisation de la méthode ne peut s'abstraire de l'environnement matériel.

Outre la mise en œuvre parallèle de notre méthode de partitionnement, nous envisageons de prolonger notre travail principalement suivant deux axes. Le premier consiste à travailler sur la minimisation du nombre des sommets de l'interface ainsi que sur la minimisation des échanges entre les sous-ensembles. Une stratégie intéressante concernant le premier point pourrait être l'application des travaux de [Liu89]. Pour le second point, une approche qui intègre la duplication des sommets des interfaces dans tous les sous-ensembles avoisinants pourrait permettre une mise en œuvre parallèle efficace du calcul sous-jacent.

La deuxième direction est de considérer les problèmes relatifs au placement de graphes. C'est en effet la suite logique des problèmes de partitionnement de graphes. Des travaux théoriques ont déjà été réalisés, par un des auteurs, dans ce domaine (voir [Lamo91]).

Remerciement

Merci à Laurent Crouzet pour nous avoir fourni notre 4^{eme} exemple qu'il a construit avec MODULEF.

Bibliographie

[BaSi92] S. T. Barnard and H. D. Simon, *A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems.* Technical Report, NASA Ames Research Center, RNR–92–033, 1992.

[Berg87] C. Berge, *Graphes.* Gauthier-Villars 1987.

[CiaJ92] P. Ciarlet, Jr, *Etude de préconditionnements parallèles pour la résolution d'équations aux dérivées partielles elliptiques. Une décomposition de l'espace* $L^2(\Omega)^3$. Thèse, Université Pierre et Marie Curie, 1992.

[Erhe90] J. Erhel, *Sparse matrix multiplication on vector computers.* International Journal of High Speed Computing, Vol. 2, No. 2, pp 101–116, 1990.

[Farh88] C. Farhat, *A simple and efficient automatic FEM domain decomposer.* Computers and Structures, Vol. 28, No. 5, 1988, pp 579–602.

[Farh89] C. Farhat, *On the mapping of massively parallel processors onto finite element graphs.* Computers and Structures, Vol. 32, No. 2, 1989, pp 347–353.

[FaLe93] C. Farhat and M. Lesoinne, *Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics.* International Journal for Numerical Methods in Engineering, Vol. 36, No. 5, Mar. 1993, pp 745–764.

[GoMi85] M. Gondran et M. Minoux, *Graphes et algorithmes.* Collection de la direction des études et recherches d'électricité de France, Eyrolles, 1885.

[HeLe92] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations.* Technical Report, Sandia National Laboratories, SAND 92–1460, 1992.

[HyRi] L. Hyafil, R. L. Rivest, *Graph partitionning and constructing optimal decisions trees are polynomial complete problems.* Communication privée.

[Laff93] T. Lafforgue, *Algorithme du gradient conjugué : vectorisation et parallélisation.* Rapport de stage CEL-V / DAM, septembre 1993.

[Lamo91] F. Lamour, *Contribution aux problèmes de placement sans routage sur un réseau de Transputers.* Thèse, Université Pierre et Marie Curie, 1991.

[Liu89] J. W. H. Liu, *A graph partitionning algorithm by node separators.* ACM Transactions on Mathematical Software, Vol. 15, No. 3, 1989, pp 198-219.

[Meur91] G. Meurant, *Numerical experiments with a domain decomposition method for parabolic problems on parallel computers.* Dans Domain decomposition methods for partial differential equations, R. Glowinski, Y. A. Kuznetsov, G. Meurant, J. Périaux and O. Widlund, eds, Siam 1991, pp 394–408.

[PoSL90] A. Pothen, H. D. Simon and K.–P. Liou, *Partitioning sparse matrices with eigenvectors of graphs.* SIAM J. MATRIX ANAL. APPL., Vol. 11, No. 3, Jul. 1990, pp 430–452.

[Simo91] H. D. Simon, *Partitioning of unstructured problems for parallel processing.* Computing Systems in Engineering, Vol. 2, No. 2/3, pp 135–148, 1991.

[Smit91] B. F. Smith, *A domain decomposition algorithm for elliptic problems in three dimensions.* NUMER. MATH., Vol. 60, 1991, pp 219–234.