PT-Scotch : Un outil pour la renumérotation parallèle efficace de grands graphes dans un contexte multi-niveaux

Cédric Chevalier, François Pellegrini

RenPar'17 / SympA'2006 / CFSE'5 / JC'2006
Canet en Roussillon, 4 au 6 octobre 2006

PT-SCOTCH : Un outil pour la renumérotation parallèle efficace de grands graphes dans un contexte multi-niveaux

Cédric Chevalier et François Pellegrini

LaBRI et INRIA Futurs
Université Bordeaux I
351, cours de la Libération, 33405 TALENCE, FRANCE
{cchevali | pelegrin}@labri.fr

Résumé
Le renumérotation parallèle de grands graphes est un problème difficile, parce que ni les algorithmes de degré minimum ni les meilleures méthodes de partitionnement de graphes nécessaires à la dissection emboîtée ne se parallélisent bien ni ne sont scalables. Cet article présente un ensemble d'algorithmes, mis en oeuvre au sein de l'outil PT-SCOTCH, qui permettent de calculer de façon parallèle et multi-tâche des renumérotations de très grands graphes, dont la qualité est équivalente à celle des meilleurs algorithmes séquentiels connus.

Mots-clés : Renumérotation, partionnement, graphe, parallélisme, multi-tâche

---

1. Introduction

Le partitionnement de graphe est une technique universelle employée dans de nombreux domaines de l'informatique et de l'ingénierie, tels que l'équilibrage des charges des processeurs en calcul parallèle, l'organisation des bases de données, la conception de circuits électroniques VLSI, ou encore la bio-informatique. Il est la plupart du temps utilisé pour aider à la résolution de problèmes spécifiques, modélisés au moyen de graphes pondérés ou non, pour lesquels trouver de bonnes solutions nécessite de calculer, éventuellement récursivement par une méthode de type « diviser pour résoudre », des partitionnements sommet ou arête de petite taille qui répartissent équitablement les poids des différentes parties du graphe.
À l'heure actuelle, les logiciels généralistes de partitionnement séquentiel de graphes comme SCOTCH [12] ou METIS [7] peuvent manipuler des graphes d'environ dix millions de sommets sur une station de travail de taille moyenne. Cependant, à mesure que la puissance des machines parallèles augmente, la taille des problèmes à traiter augmente aussi, et comme des graphes de plus de cinquante millions de sommets ne peuvent être traités sur un unique ordinateur sans écrouler sa mémoire, il est nécessaire de disposer d'outils parallèles de partitionnement de graphes. Plusieurs de ces outils ont été développés, mais leurs résultats sont mitigés. En particulier, ils ne passent pas bien à l'échelle, et la qualité du partitionnement décroît à mesure que le nombre de processeurs utilisés par le programme augmente.
Le but du logiciel PT-SCOTCH (« *Parallel Threaded* SCOTCH »), une extension du logiciel SCOTCH séquentiel développé au LaBRI dans le cadre du projet SCALAPPLIX de l'INRIA Futurs, est de permettre le partitionnement parallèle efficace de graphes de plus d'un milliard de sommets.

Comme, d'après ce qui précède, il n'est pas souhaitable que les fragments de graphes distribués localisés sur chaque nœud de calcul aient une taille dépassant le million de sommets, PT-SCOTCH doit être conçu pour bien passer à l'échelle, afin de pouvoir calculer des partitions efficaces sur environ mille processeurs.
L'une des applications cibles de PT-SCOTCH au sein du projet SCALAPPLIX est la renumérotation de graphes, qui est un problème crucial pour la factorisation efficace des matrices creuses symétriques, non seulement pour réduire le remplissage et le coût de factorisation, mais aussi pour accroître le parallélisme de l'arbre d'élimination, ce qui est essentiel pour atteindre des performances élevées lorsqu'on résoud ces systèmes linéaires sur des architectures parallèles.
Les deux méthodes séquentielles de renumérotation les plus couramment utilisées dans la communauté scientifique sont les méthodes de degré minimum et des dissections emboîtées. L'algorithme de degré minimum [13] est une heuristique locale extrêmement rapide et souvent efficace, mais elle est intrinsèquement séquentielle, ce qui fait que les tentatives visant à sa parallélisation n'ont pas été concluantes [2], surtout sur les architectures à mémoire distribuée. La méthode des dissections emboîtées [10], en revanche, se prête bien à la parallélisation, puisqu'elle consiste à calculer un séparateur de taille minimale découpant le graphe en deux parties de tailles équivalentes, à renuméroter ce séparateur avec les plus grands indices disponibles, puis à répéter récursivement ce processus sur les deux sous-graphes obtenus jusqu'à ce que la taille des sous-graphes soit inférieure à un seuil limite.
Cet article présente les algorithmes mis en oeuvre dans PT-SCOTCH pour paralléliser la méthode des dissections emboîtées et calculer efficacement en parallèle des renumérotations de grande qualité. Nous décrirons dans la section suivante le schéma algorithmique de notre programme, puis présenterons en section 3 les premiers résultats obtenus. Viendra ensuite la conclusion, où nous donnerons quelques perspectives sur notre travail futur.

2. Schéma algorithmique de la renumérotation parallèle

Le calcul parallèle des renumérotations dans PT-SCOTCH fait appel à différents niveaux de parallélisme. Le premier niveau concerne la parallélisation de la méthode des dissections emboîtées proprement dite. Partant du graphe initial, distribué sur p processeurs de façon arbitraire, l'algorithme procède comme illustré en figure 1 : une fois un séparateur obtenu en parallèle par une méthode décrite plus bas, chacun des p processeurs participe à la construction du sous-graphe induit distribué correspondant à la première partie (cependant, certains processeurs peuvent n'en posséder aucun sommet), puis effectue le repliement de ce graphe sur les $\lfloor \frac{p}{2} \rfloor$ premiers processeurs, afin que le nombre moyen de sommets par processeur, garant de l'efficacité du fait des possibilités de recouvrement des communications par le calcul, reste constant. Une procédure identique est utilisée pour construire sur les $\lceil \frac{p}{2} \rceil$ autres processeurs le sous-graphe induit replié correspondant à l'autre partie. Ces deux constructions étant indépendantes, chaque calcul d'un sous-graphe induit et de son repliement peut s'effectuer en parallèle, par la création temporaire d'une tâche supplémentaire par processeur. À la fin du processus de repliement, chaque processeur dispose d'un unique fragment de l'une des deux parties, et le processus de dissections emboîtées peut se poursuivre récursivement, de façon totalement indépendante, sur chacun des groupes de $\frac{p}{2}$ (puis $\frac{p}{4}$, $\frac{p}{8}$, etc.) processeurs résultants, jusqu'à ce que chaque groupe soit réduit à un unique processeur. Dès lors, le processus de dissections emboîtées se poursuivra séquentiellement sur chaque processeur, au moyen des routines de la bibliothèque SCOTCH, avec couplage éventuel avec des méthodes de degré minimum [11] (mais donc seulement de façon séquentielle).

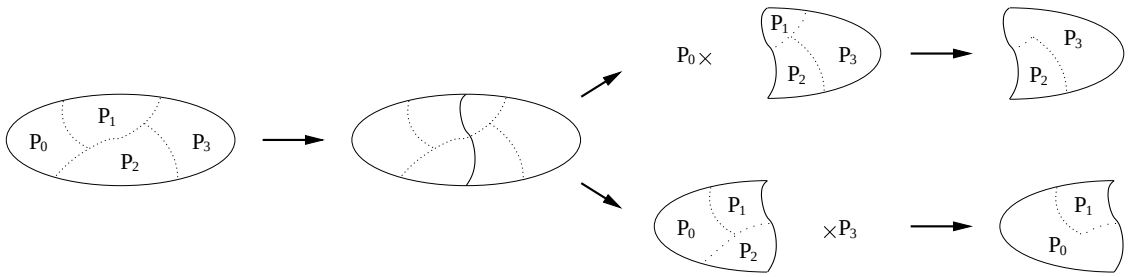

FIG. 1 – Schéma d'une étape de dissection pour un (sous-)graphe distribué sur 4 processeurs. Une fois le séparateur connu, les deux sous-graphes induits sont construits (éventuellement en parallèle), puis repliés, formant deux sous-graphes distribués sur 2 processeurs chacun.

Le second niveau de parallélisme concerne le calcul des séparateurs. L'approche choisie est de type multi-niveaux [1, 5, 6] : cette méthode consiste à réduire de façon répétitive la taille du graphe à partitionner en trouvant des appariements qui fusionnent les sommets et les arêtes tout en conservant la structure topologique du graphe initial, puis à calculer une partition initiale sur le graphe le plus grossier, dont la taille n'excède pas quelques centaines de sommets, et enfin à projeter la partition obtenue, de proche en proche sur les graphes intermédiaires, jusqu'à obtenir une partition du graphe original. Le plus souvent, on utilise un algorithme d'optimisation locale de type Kernighan-Lin [9] ou Fiduccia-Mattheyses [4] (FM) pour raffiner la partition projetée à chaque niveau, afin que la granularité de la solution soit celle du graphe original et non celle du graphe le plus grossier.

Notre implémentation est schématisée en figure 2 : l'appariement des sommets s'effectue en parallèle au moyen d'un algorithme probabiliste asynchrone mettant en œuvre plusieurs tâches par processeur. La phase de construction du sous-graphe contracté peut être paramétrée, permettant de choisir entre la conservation de l'ensemble des sommets contractés sur leur processeur original (illustré dans les premières phases de la figure 2), qui réduit le nombre de sommets possédés par chaque processeur et accélère les calculs ultérieurs, ou bien le repliement avec duplication du graphe contracté (phases suivantes de la contraction en figure 2), qui multiplie le nombre de copies de travail du graphe, et peut ainsi améliorer la qualité de la partition obtenue. En effet, les algorithmes de calcul de partitions, qui sont des heuristiques locales, dépendent fortement de la qualité des graphes contractés, et nous avons déjà observé avec la version séquentielle de SCOTCH que prendre à chaque fois la meilleure partition parmi celles renvoyées par deux cycles multi-niveaux complets améliorait la qualité finale de la renumérotation. En optant pour le repliement avec duplication dans les premiers niveaux de contraction, on met ainsi en œuvre ce mécanisme de façon parallèle, chaque sous-groupe de processeurs possédant une copie du graphe se chargeant alors d'un calcul multi-niveaux indépendant presque complet, à part les premières étapes qui sont communes.

Dès le moment où les graphes contractés sont repliés sur un unique processeur, on passe dans une phase multi-séquentielle, illustrée en bas de la figure 2 : on appelle les routines de la bibliothèque SCOTCH, qui poursuivent séquentiellement la contraction, calculent la partition initiale, et projettent celle-ci jusqu'au plus gros graphe séquentiel possédé. Ensuite, c'est de façon parallèle que la partition est projetée de proche en proche sur les graphes distribués de taille de plus en plus grande, avec sélection de la meilleure partition entre deux si l'on remonte d'un niveau avec repliement et duplication, jusqu'à obtenir une partition du graphe initial.

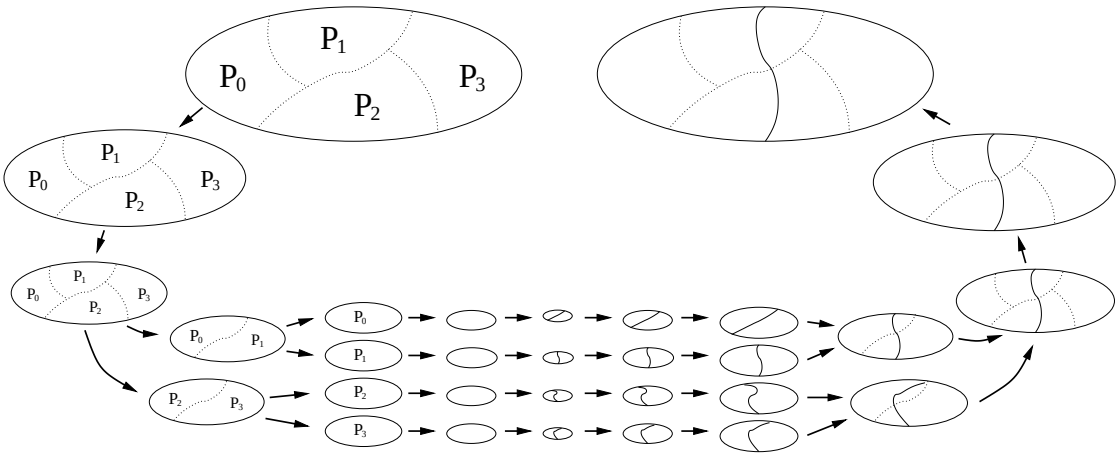

FIG. 2 – Schéma du calcul parallèle du séparateur d'un graphe distribué sur 4 processeurs, par contraction parallèle avec repliement et duplication éventuels, calcul multi-séquentiel de partitions initiales raffinées localement sur chaque processeur, puis expansion parallèle de la meilleure partition trouvée.

Le troisième niveau de parallélisme concerne les heuristiques de raffinement qui sont utilisées pour améliorer chaque projection. Tant que l'on reste dans le cadre multi-séquentiel, on peut utiliser l'algorithme séquentiel FM de SCOTCH, mais ce type d'algorithmes se parallélise mal. Ainsi, PARMETIS [8] possède une version parallèle de FM mais, afin de relâcher la contrainte séquentielle forte qui nécessiterait une communication chaque fois que l'on souhaite migrer un sommet ayant des voisins sur d'autres processeurs, seuls les mouvements qui améliorent strictement la qualité de la partition sont autorisés, ce qui limite les capacités de l'algorithme FM à sortir des minima locaux de sa fonction de coût, et conduit à dégrader fortement la qualité des solutions fournies à mesure que le nombre de processeurs (et donc de voisins potentiels distants) augmente.
Nous avons proposé et testé avec succès dans [3] une solution à ce problème : comme chaque raffinement est réalisé au moyen d'un algorithme local, qui ne perturbe que peu la position du séparateur projeté, seuls sont nécessaires à l'algorithme d'optimisation les quelques rangs de sommets situés à une distance au plus 3 du séparateur projeté. Ce sous-graphe de bande, graphe distribué induit obtenu par les mêmes algorithmes que ceux utilisés pour construire les sous-graphes de la dissection emboîtée, peut alors être multi-centralisé sur chacun des processeurs du graphe, pour qu'un algorithme d'optimisation séquentiel de type FM lui soit appliqué, et que la meilleure partition obtenue soit ensuite reprojetée sur le graphe distribué, comme illustré en figure 3. Cette solution est acceptable car, pour les graphes de maillages 2D et 3D, la taille des séparateurs est de plusieurs ordres de grandeur inférieure à la taille des graphes séparés : en $O(n^{\frac{1}{2}})$ pour les maillages 2D et en $O(n^{\frac{2}{3}})$ pour les maillages 3D [10]. La petite taille du graphe bande nous a également permis d'utiliser avec succès un algorithme génétique, travaillant en parallèle sur l'ensemble des processeurs recevant une copie du graphe bande, pour optimiser le séparateur de façon scalable ; voir [3] pour plus d'informations à ce sujet.

Au cours de son exécution, PT-SCOTCH construit une structure d'arbre distribué s'étendant sur chacun des processeurs sur lesquels il a été lancé, et dont les feuilles représentent des fragments

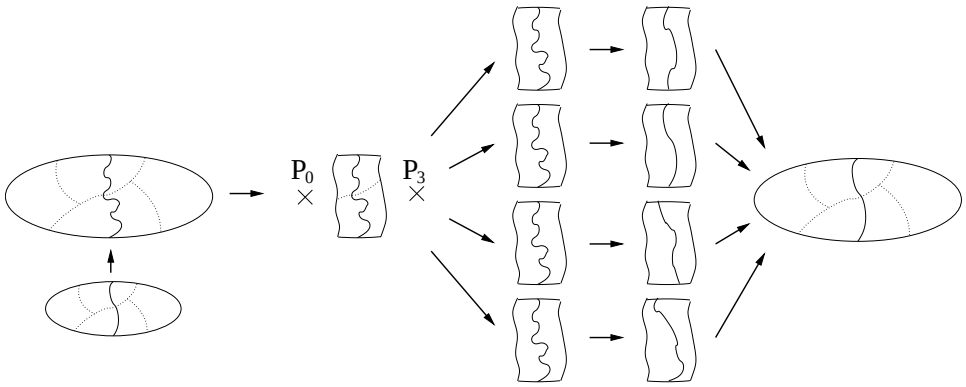

FIG. 3 – Schéma du raffinement multi-séquentiel d'un séparateur projeté à partir d'un graphe distribué grossier à 4 processeurs vers son graphe distribué père. Une fois le graphe bande distribué construit, une version centralisée en est construite sur chacun des processeurs, pour y exécuter un algorithme séquentiel d'optimisation locale de type FM. La meilleure partition raffinée est alors redistribuée sur le graphe initial.

Graphe	**Taille** ($\times 10^3$)		**Degré moyen**
	V	E	
altr4	26	163	12.50
channel1m	81	527	13.07
conesphere1m	1055	8023	15.21

TAB. 1 – Les graphes utilisés pour nos premiers tests.

de la permutation inverse décrivant la renumérotation calculée. Nous utilisons la permutation inverse plutôt que la permutation directe parce que la permutation inverse peut se construire et se maintenir de façon totalement distribuée : tout sous-graphe à renuméroter n'est décrit, du point de vue de la permutation, que par le nombre de sommets à renuméroter, ainsi que par le plus petit indice de la nouvelle numérotation à affecter aux sommets du sous-graphe. Lorsqu'un sous-graphe (séparateur ou feuille de l'arbre de séparation) doit être numéroté, on crée un nouveau fragment de permutation sur chacun des processeurs possédant des sommets du sous-graphe, chaque fragment ayant pour taille le nombre de sommets du sous-graphe possédés localement par le processeur, et comme numéro d'indice initial le plus petit numéro d'indice attribué au sous-graphe, auquel s'ajoute la somme des tailles des fragments hébergés par les processeurs de rang inférieur. L'ordre dans lequel les indices globaux des sommets du sous-graphe sont agencés dans chaque fragment définit la façon dont sont numérotés les sommets du sous-graphe.

3. Résultats expérimentaux

Les résultats ci-dessous ont été obtenus sur un cluster IBM de processeurs Power5 *dual-core* à 1,5 GHz. Les graphes que nous avons utilisés, fournis par le CEA, sont listés dans le tableau 1. Le tableau 2 nous présente les conséquences de l'utilisation du repliement et de la duplication

Cas test	Nombre de processeurs		
	2	4	8
SCOTCH **avec repliement et duplication**			
OPC	**3.60e+08**	**3.60e+08**	**3.56e+08**
Temps (s)	0.97	0.87	0.98
SCOTCH **sans repliement**			
OPC	3.70e+08	3.67e+08	3.86e+08
Temps (s)	1.56	2.09	4.40
PARMETIS			
OPC	4.19e+08	4.49e+08	4.46e+08
Temps (s)	0.31	0.20	0.13

TAB. 2 – Comparaison de la qualité, en termes de nombre d'opérations nécessaires à la factorisation et de temps nécessaire au calcul de la renumérotation, des renumérotations calculées par PARMETIS et SCOTCH avec et sans le repliement du graphe lors du partitionnement multi-niveau parallèle, pour le graphe **altr4**.

lors des phases de bipartitionnement multi-niveaux. Comme on pouvait s'y attendre, cette méthode nous apporte un gain de qualité, dû aux essais multiples effectués en parallèle lorsque c'est possible. Plus surprenant est le fait que l'utilisation du repliement et de la duplication nous permet de gagner du temps par rapport à la version qui en est privée.
Ceci peut s'expliquer par le fait que notre algorithme d'appariement asynchrone génère énormément de communications entre sommets voisins appartenant à des processeurs différents. Sans repliement, les graphes manipulés lors de la phase de contraction sont plus répartis, et celle-ci est donc bien plus coûteuse. Des optimisations de cet algorithme sont en cours de développement, pour grouper les communications afin de soulager le système. Pour le moment aussi, le repliement nécessite que le nombre de processeurs soit une puissance de deux, mais une amélioration permettant une redistribution des données sur un nombre quelconque de processeurs est également en cours de développement. Il est à noter que PARMETIS utilise lui aussi la technique du repliement, avec les mêmes contraintes sur le nombre de processeurs, mais sans duplication.

OPC	Nombre de processeurs					
	2	4	8	16	32	64
conesphere1m (OPC séquentiel : 1.81e+12)						
SCOTCH	**1.81e+12**	**1.79e+12**	**1.82e+12**	**1.83e+12**	**1.87e+12**	**1.88e+12**
PARMETIS	2.20e+12	2.46e+12	2.78e+12	2.96e+12	2.99e+12	3.29e+12
channel1m (OPC séquentiel : 2.42e+09)						
SCOTCH	**2.44e+09**	**2.45e+09**	**2.43e+09**	**2.47e+09**	**2.53e+09**	**2.50e+09**
PARMETIS	2.83e+09	2.85e+09	3.11e+09	3.28e+09	3.62e+09	3.67e+09

TAB. 3 – Comparaison de la qualité, en terme de nombre d'opérations dans la factorisation, de la renumérotation obtenue entre PARMETIS et SCOTCH lors d'exécutions parallèles.

Le tableau 3 montre les résultats obtenus en terme de qualité de la renumérotation calculéeproduite entre SCOTCH, dans sa version sans repliement, et PARMETIS. La stratégie de renumérotation utilisée par SCOTCH est de faire une méthode multi-niveau avec comme optimisation lors de la remontée l'utilisation de FM sur une bande séquentialisée. On constate que l'intérêt de cette méthode est évident en terme de qualité, puisque celle ci reste constante, quel que soit le nombre de processeurs.

4. Conclusion et perspectives

Nous avons présenté dans cet article les différents algorithmes parallèles mis en œuvre par le renuméroteur parallèle de matrices creuses PT-SCOTCH, ainsi que les premiers résultats obtenus, qui sont très satisfaisants en terme de qualité mais demandent à être améliorés en terme de scalabilité.

Bien qu'elle corresponde à une préoccupation actuelle au sein du projet SCALAPPLIX, pour obtenir rapidement des numérotations de qualité de graphes de l'ordre de la dizaine de millions de sommets, la renumérotation de matrices n'est pas le domaine d'application dans lequel se trouveront les plus gros cas à traiter, car les programmes parallèles actuels de factorisation de matrices creuses ne peuvent traiter des matrices de maillages 3D supérieurs à la trentaine de millions d'inconnues.

Nous comptons donc, en nous basant sur les briques logicielles déjà écrites, étendre l'interface de PT-SCOTCH pour réaliser des partitionnements k-aires de grands maillages destinés aux méthodes itératives, ainsi que des placements statiques de graphes de processus, à l'image de ce que propose la bibliothèque SCOTCH séquentielle. La conservation de la scalabilité de notre programme pour de grands graphes nécessitera plusieurs optimisations, comme l'amélioration de l'équilibrage de la charge lors du processus de repliement (avec redistributions afin de gérer des nombres de processeurs qui ne soient pas uniquement des puissances de deux), le groupage des communications asynchrones de l'algorithme d'appariement, ainsi que la mise en place de graphes de bande distribués, lorsqu'ils seront trop gros pour tenir dans la mémoire d'un unique processeur, sur lesquels opèreront des versions adaptées de notre algorithme génétique de raffinement des séparateurs.

Bibliographie

1. S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency : Practice and Experience*, 6(2) :101–117, 1994.
2. T.-Y. Chen, J. R. Gilbert, and S. Toledo. Toward an efficient column minimum degree code for symmetric multiprocessors. In *Proc. 9th SIAM Conf. on Parallel Processing for Scientific Computing, San-Antonio*, 1999.
3. C. Chevalier and F. Pellegrini. Improvement of the efficiency of genetic algorithms for scalable parallel graph partitioning in a multi-level framework. `http ://www.labri.fr/~pelegrin/papers/scotch_efficientga.pdf`. Soumis à EuroPar 2006.
4. C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181. IEEE, 1982.
5. B. Hendrickson and R. Leland. The CHACO user's guide – version 2.0. Technical Report SAND94–2692, Sandia National Laboratories, 1994.
6. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Tech. Rep. 95-035, University of Minnesota, June 1995.

7. METIS : Programs for partitioning unstructured graphs and hypergraphs and computing fill-reducing orderings of sparse matrices. http ://www-users.cs.umn.edu/~karypis/metis/.
8. PARMETIS : Programs for parallel graph partitioning and sparse matrix ordering. http ://www-users.cs.umn.edu/~karypis/metis/parmetis.
9. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49 :291–307, February 1970.
10. R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16(2) :346–358, April 1979.
11. F. Pellegrini, J. Roman, and P. Amestoy. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. *Concurrency : Practice and Experience*, 12 :69–84, 2000.
12. SCOTCH : Static mapping, graph partitioning, and sparse matrix block ordering package. http ://www.labri.fr/~pelegrin/scotch/.
13. W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *J. Proc. IEEE*, 55 :1801–1809, 1967.