ハイパフォーマンス
コンピューティング 52−14
（1994. 7. 23）

あるタスクグラフ分割アルゴリズム

須田 礼仁 (東京大学 理学部 情報科学科)

並列処理におけるもっとも重要かつ困難な問題の一つが高速で効率的な問題分割である。この研究ではデータの相互作用を表わす無向グラフを再帰的に2分割を繰り返す方法の一つとして隣接分割法を提案し、メッシュや、並列回路シミュレーションに現われるグラフを分割させてその性能を他の代表的な方法と比較検討した。この方法は計算量がO(N^2)と比較的大きいが、幅広い問題に対して良好な結果をあたえた。また、同様に計算量がO(N^2)のランダム分割も良好な性能を示したが、大規模な問題に対しO(N)サンプルでは性能が低下することを示唆するデータが得られた。計算量O(N)の諸解法は性能が悪かった。O(N^2)の方法はサンプリングを行うため並列化が容易であり、実装によってその高い並列化効率を立証することができた。回路解析のようにランダム構造を持ち並列化効率の出しにくい問題についてはこのような高精度の方法が必要であると思われる。

A Split Algorithm for Task Graphs

Reiji Suda (Dpt. Info. Sci., Fac. Sci., Univ. Tokyo)

One of the most important and most difficult problem for parallel processing is fast and high quality problem mapping. This paper proposes a split algorithm, "neighbor-split method," for undirected graph which represents the dependency of the data, and compare it with some other methods applying them to mesh graphs and graphs which appear in parallel circuit simulation. The proposed method has a computational complexity as high as O(N^2), but provides excellent splits for a wide range of problems. Similar results were obtained by random split method with also O(N^2) of complexity, but some data imply that O(N) sampling for large scale problems may degrade quality. Other methods of O(N) complexity were not satisfiable. The proposed method has a high and trivial parallelism because it chooses the best split from a large number of samples, and the high parallelism is confirmed by evaluation through implementation. High-quality methods will be necessary for randomly structured, uneasily parallelizable problems such as circuit analysis.

1. はじめに

並列処理にとって不可避かつ困難な問題のひとつにタスクの分割がある。プロセッサの遊びを作らず効率的に並列処理を行うには各プロセッサの仕事量が平均していることと通信・同期のオーバーヘッドが小さいことが必要であるが、この両者は必ずしも両立しない。そして最適な分割を求める問題はNP完全であるため、実用上はなんらかの近似アルゴリズムを用いてタスクを分割せざるをえない。

タスク分割はタスクを表わすグラフをプロセッサを表わすグラフにマッピングするというかたちに定式化するのが一般的である。以下ではPをプロセッサ数、Nをタスクグラフの頂点と辺の数の合計とする。タスクグラフは仕事の依存関係を表わす有向グラフの場合とデータの相互作用を表わす無向グラフの場合とがあるが、ここでは無向グラフを取り扱うことにする。また、一度にP台のプロセッサに仕事を割り付ける場合と2分割を繰り返して割り付ける方法とがあるが、ここでは後者を採り上げる。これはP台のプロセッサに一度に割り付ける方法は計算量がPに強く依存することが多く、大規模並列処理には使えないためである。これに対し繰り返し2分割を行う場合は2分割の計算量がO(N)またはO(N^2)であればP台への割り付けにはO(N log P)またはO(N^2)の時間でよい。これは十分実用的な範囲と思われる。

2. これまでに提案されてきた方法

これまでにも多くのグラフ分割アルゴリズムが提案されてきた。大きく分けると、グラフを準備なしに分割する分割アルゴリズムと、ある分割を初期値としてそれを改良する改良アルゴリズムとがある。さらに、以下のような種類に細分できる [7]。

順序づけによる分割

この方法は連立一次方程式の解法であるLU分解のためのグラフの頂点の順序づけを利用する。単純に順序の途中で2つに分割する方法 (線形分割) と、依存関係と順序から木を構成してこれを分割する方法 (木分割 [4]) とがある。順序づけの方法としては有名なものはCuthill-McKee, reverse CM [1], GPS [2] など帯行列を期待するもののほか nested disection [3] などがある。

グラフ的距離による分割

適当な頂点を選び、そこからの距離に従ってグラフを分割する (距離分割)。2頂点をとって双方からの距離の差を用いる場合もある (ボロノイ分割)。頂点の選び方が重要で、pseudo-peripheral [2] がひろく用いられるが、ランダムに選んで最適なものを採用することもできる (ランダム分割)。

グラフの縮約による分割

グラフの密に結合している頂点同士を合併することを繰り返し、最終的に頂点の数を2までする方法である [6]。この解法は一度にP台のプロセッサに仕事を割り付けても時間が余計にかからないのが特徴である。

局所的な改良

ある程度よい分割を基にして、評価を向上させる様に頂点をひとつ、あるいはいくつかずつ移動させるもの。高速であるためCADの分野で広く用いられてきた。局所的な最適解に到達する。

確率的改良

Simulated annealing, neural network, genetic algorithm など、乱数を用いて確率的に改良する方法である。適切な設定のもとで十分時間をかければ最適解が理論的にはもとまるが、全体的改良は時間がかかりすぎる。むしろ局所的な改良に用いるほうが期待がある。

3. 提案するアルゴリズム

ここではひとつの距離に基づく分割アルゴリズムを提案する。このアルゴリズムは辺を一つ選び、その辺を除いたグラフにおいて辺の両頂点から各頂点への距離を求め、その距離の差によって頂点をレベルづけする。ここで、グラフから当該の辺を除いてから距離を計ることが重要である。そう

しないとレベルはいつも3つになってしまう。最後にこのレベルにしたがって仕事のバランスの最も良いところでグラフを2つに切る。この方法を以下では「隣接分割」とよぶ。

このアルゴリズムにおいて重要なのは辺の選び方である。アルゴリズムの基本形ではすべての辺を選んで最適なものを最終的に選ぶ。計算量を減らしたい場合は辺を選ぶ必要があるが、ランダムに選んでもよい。また、ここではある辺を除いたグラフにおけるその辺の両頂点の距離をその辺の「弱さ」と定義して、ある程度よりも弱い辺をすべて選ぶことも有効である。

この方法では一辺で決まる隣接分割を計算するのにO(N)の時間がかかる。よって全部の辺を選ぶ方法では全体でO(N^2)の計算量が必要になる。これは、これまでに提案されてきた多くの解法の計算量O(N)に比べ大きい。しかしこの後見るように、分割性能はO(N)の方法に比べはるかによい。しかも同じく計算量O(N^2)のランダム分割がほぼ同程度の性能を示す。Simulated annealingはきわめて小さい問題しか実用的な時間で解くことができない。

4. アルゴリズムの比較

ここでは3種類の問題グラフを用いてアルゴリズムを評価比較する。対比する方法はO(N)の方法4つ: (1) GPS順序による線形分割 (GPS-L)、(2) GPS順序による木分割 (GPS-T)、(3) pseudo-peripheralを8つ選び、それらによるボロノイ分割の最適なものを選ぶ (BS8)、(4) グラフ縮約による2分割を16回行い、最適なものを採用する (GC16)、O(N^2)の方法3つ: (5) ランダムにN組の点を選び、それらによるボロノイ分割の最適なものを選ぶ (RS1)、(6) 全部の辺をチェックする隣接分割 (NS0)、(7) 弱さが4以上の辺のみすべて選ぶ隣接分割 (NS4)、それにsimulated annealingをもちいる方法2つ: (8) 弱さが4以上の辺をすべて除いたグラフの連結部分をクラスタとし、それらをsimulated annealingで2分割する (ES4)、(9) グラフ縮約を24クラスタまでとし、それらをsimulated annealingで2分割する (GC24)、の9つである。

問題は3種類用意した。第1のものはメッシュで、これは理論的に考察するにとどめる。第2に現実の回路シミュレーションにあらわれた回路の結合網そのもので、ランダムで非常に疎なグラフである。第3はこれらの回路を前処理付Jacobi法 [6] で解く場合の計算をグラフにしたもので、これはensparsed decomposed inverseが用いられているため回路網に比べてグラフがはるかに密である。

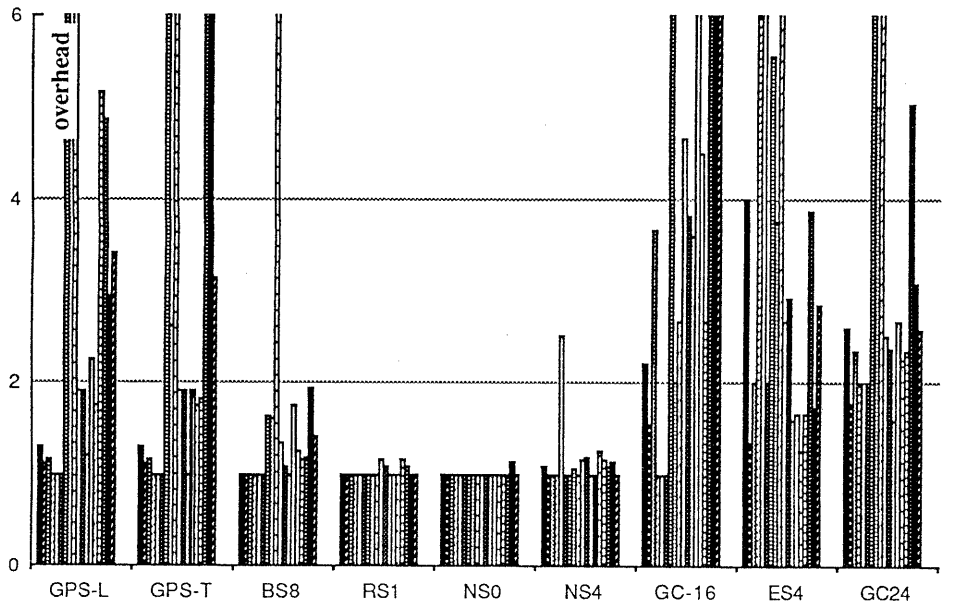

図1. 疎なグラフに対する各解法の性能

メッシュ問題にたいして隣接分割は、全部の辺を選べば、最も長い辺の真ん中で2分する。これは多くの場合最適である。またランダムに辺を選ぶ方法でも3次元の場合 O(N^1/3) の辺を選べば最適分割が求まる。このような最適分割はボロノイ分割[1] あるいは O(N^2/3) のサンプルをもちいたランダム分割でも可能である。GPS 法はメッシュを斜めに切ってしまうため最適な分割は与えない。グラフ縮約法が最適な分割を与える確率はきわめて低い。このように、メッシュ問題に対してはボロノイ分割、隣接分割、ランダム分割が最適解を与えるが、計算量はこの順に増加する。

回路の結合グラフに対しては図1のような結果が得られた。縦軸は各方法による2分割の結果の、重いほうのプロセッサの仕事量に通信量を加えたもので、9種類の方法のうち最も良いもので正規化してある。これで見ると、全辺を試す隣接分割 NS0 がほぼ完全で最も良く、ランダム分割 RS1 と弱い辺のみによる隣接分割 NS4 が続く。他のものはかなり悪い。

次に、前処理つき Jacobi 法のタスクグラフの分割の結果を図2に示す。ここではすべての辺の弱さが2であるため、辺の弱さを用いる NS4 と ES4 は適用できない。ここでは隣接分割はわずかに性能が落ち、ランダム分割が最適である。他の解法はあまりよくない。

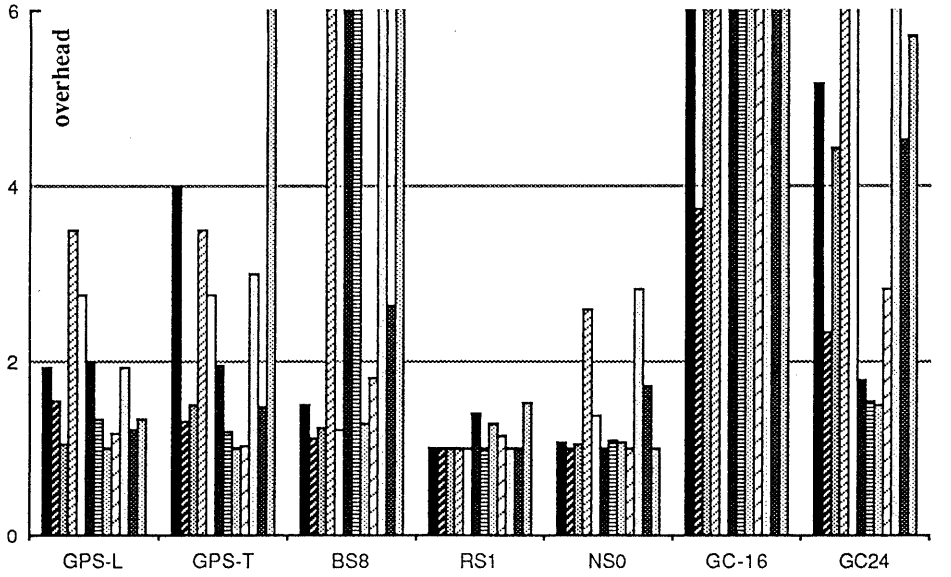

図2．密な行列に対する各解法の性能

このように、O(N^2) の計算量をもつ隣接分割、ランダム分割が他の方法に比べて絶対的に良好であった。隣接分割は特にメッシュ、疎なグラフに対して、また、ランダム分割はランダム構造のグラフに対して良好であった。これらの結果をまとめると、次のようになる。

問題	順序分割	ボロノイ	ランダム	隣接分割	グラフ縮約
メッシュ	△	○	△	○	△
回路結合	△	△	○	○	×
PJ 法	△	△	○	○	×

このように、計算量の大きい方法がよい結果を出すことは、ある意味で当然である。しかし、特に構造がランダムで並列化効率が出しにくい回路関係のシミュレーションなどには高精度の問題分割が不可欠である。ここでは第1分割のみを取り扱ったが、実際にはこの分割をもとにしてさらに分割を続けなければならない。このときわずかな分割性能の差が拡大して大きな差になることは容易に理解できる。そのためこの程度の計算量をかけてよい分割を得ることは必要不可欠である。

[1] ここではメッシュに対して最適解を与えるようにボロノイ分割のアルゴリズムを改良して用いた。

ランダム分割と隣接分割とは両方ともサンプリングを行っている。つぎに、どの程度のサンプリングでどの程度の性能が得られるかを考察する。図3はサンプル数をN/rとしたときのrと性能をプロットしたものである。回路は比較的大きなもの3つである。右の図から隣接分割は問題の大きさに余りよらない分解性能を示し、常に全数サンプルすることが望ましいことがわかる。これに対し左のランダム分割は回路が大きくなるほど性能の低下が激しくなる。このことからランダム分割は小さい係数ではあるもののO(N^2)などのサンプリングが適切であることが予想され、数万ノード以上の問題にたいしてO(N)サンプルのランダム分割の性能が低下する恐れがある。しかし今回のサイズの問題ではそれを確認することはできなかった。

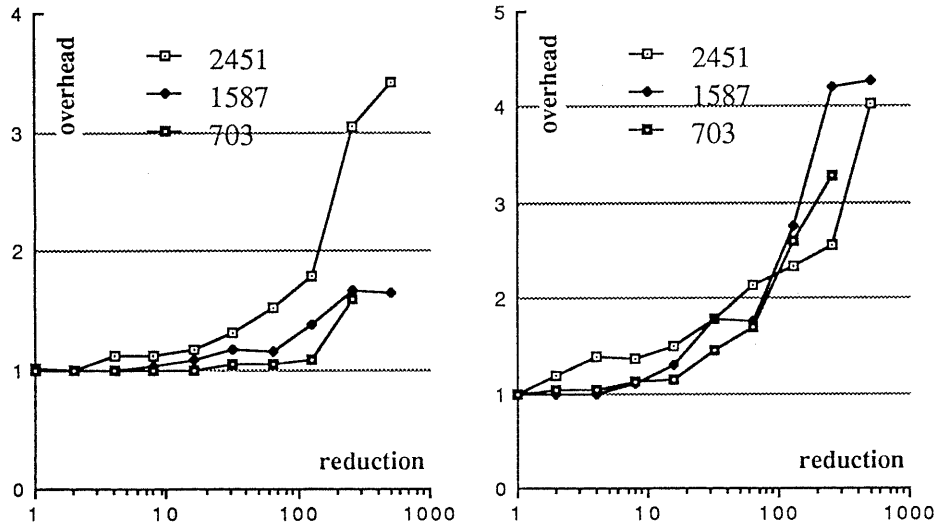

図3.サンプル数と分解性能（左：ランダム分割、右：隣接分割）

最後に、隣接分割以外の方法の特徴について触れておく。まず、順序による分割では、CM, RCM, NDにくらべて圧倒的にGPSが良かった。しかし、線形分割と木分割ではさほどの違いはなかった。厳密にいえば順序による分割は基本的に頂点分割法であるのでこのような辺分割には必ずしも向いていない。とくにnested dissectionはそうである。グラフ縮約法の系統は概して性能が悪い。また、それ自体が近くだけを見ている方法であるのである意味で局所最適解に落ちており、simulated annealingなどをおこなってもほとんど改良されない。これに対し順序づけや距離による方法はいずれもたいていの場合、分割辺を含む頂点のみに対して部分的なsimulated annealingをほどこすことによって実用的な時間でさらに改良することができる。

5. 並列タスク分割

本来の問題のほうは並列化され、より大規模な問題がより短い時間で解かれるようになるであろう。しかしこのスピードアップに問題分割の方が追い付かなければ決して実用的な並列処理は実現しない。そのためには問題分割自体の並列処理が必要である。しかし、効率的な問題分割の並列処理のために問題分割が必要なのでは話にならない。これに対し、ここでとりあげた2つのO(N^2)アルゴリズムにおいては、かなりの数の分割方法をためしてそのうち最適のものを採り上げる方法が使われている。このようなアルゴリズムは自明な並列性を持つため並列化が容易にできる。

最初の2分割はそれでよい。しかし第2段階以降の分割は少々問題を生ずる。第1分割で生じた2つのクラスタの次の分割は、逐次的に処理しなければ最適解が得られない。第2段階の2つの分割をそれぞれP台全部のプロセッサを用いて行う方法も考えられるが、ここではプロセッサの遊びを許してP/2台ずつ、しかし逐次的に処理することにした。この場合、並列化効率は3/2になるものと期待される。図4はこの方法で隣接分割法をAP-1000に実装し回路網グラフを分割した結果で、予想に近い並列化効率が得られている。

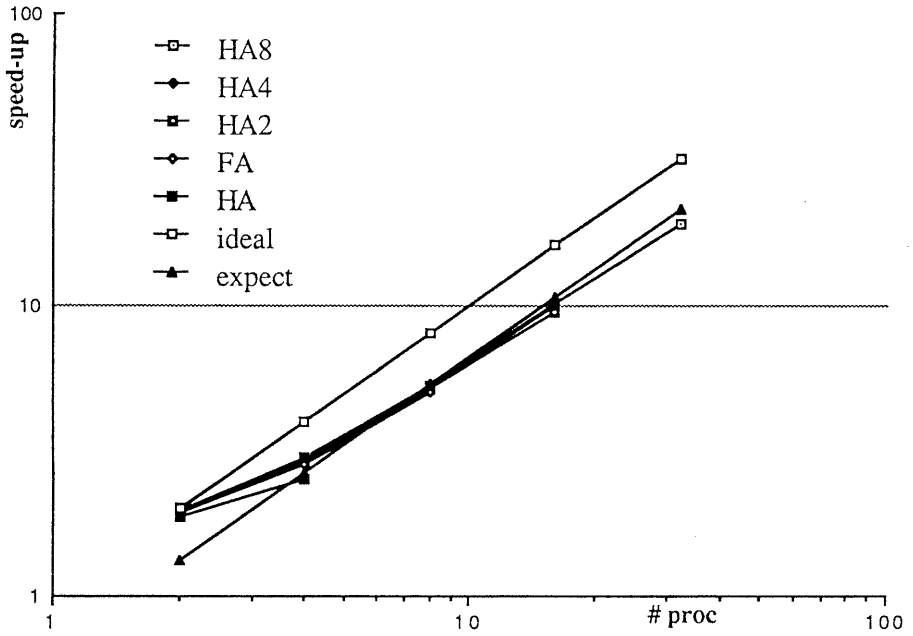

図4．並列隣接分割の並列化効果

6. まとめ

本論文では並列処理のための仕事分割の方法としてO(N^2)の計算量をもつグラフの隣接分割法を提案し、メッシュおよび並列回路解析に出てくるグラフの分割性能を他の方法と比較した。この方法はより計算量の少ない他の方法に比べ広い範囲のグラフに対して安定な分割性能をあたえることがわかった。また、同様にO(N^2)の計算量をもつランダム分割も良好であった。このことから、特に高精度の分割が必要な場合にはこのような計算量の大きい方法を用いるほうがよいことが予想される。また、これらの方法は効率的に並列化することができ、このことを実装によって確認した。

7. 参考文献

[1] W-H. Liu and A. H. Sherman, "Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algoriothms for Sparse Matrices," SIAM J. Num. Anal., Vol. 13, No. 2, Apr. 1976, pp. 198--213.

[2] N. E. Gibbs, W. G. Poole Jr, and P. K. Stockmeyer, "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," SIAM J. Num. Anal., Vol. 13, No. 2, Apr. 1976, pp. 236--256.

[3] A. George and J. W. H. Liu, "An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems," SIAM J. Num. Anal., Vol. 15, No. 5, Oct. 1978, pp. 1053--1069.

[4] J. W. H. Liu, "A Graph Partitioning Algorithm by Node Separators," ACM Trans. Math. Soft., Vol. 15, No. 13, Sep. 1989, pp. 198--219.

[5] 須田礼仁，「前処理付き Jacobi 法による並列回路 Simulation,」情報処理学会研究報告 Vol. 93, No. 72 (93-HPC-48), Aug. 1993, pp. 73--79.

[6] R. Ponnusamy, N. Mansour, A. Choudhary, and G. C. Fox, "Mapping Realistic Data Sets on Parallel Computers," pp. 123--128.

[7] N. Chrisochoides, E. Houstis, and J. Rice, "Mapping Algorithms and Software Environment for Data Parallel PDE Iterative Solvers," J. Para. Dist. Comp., No. 21, 1994, pp. 75--95.