アルゴリズム 69－1
（1999．9．2）

グラフ分割問題に対するメタ戦略の並列化

丸田寛之* 加地太一†
小樽商科大学

概要

グラフ分割問題は頂点と辺で構成されるグラフをカットエッジが最小となるように分割する，NP-complete な組合せ最適化問題の一種である．本研究ではグラフ分割問題に Tabu Search, Simulated Annealing の各種メタ戦略を適用し，実験と考察を行う．さらに各戦略を並列化し，速度および解の質の改善を試みた．

Meta Strategy Parallelization for Graph Partitioning Problem

Hiroyuki Maruta Taichi Kaji
Otaru University of Commerce

abstract

The graph partitioning problem, that is NP-complete problem, is to find the minimum cost partition of the nodes of a graph into subset of a given size. We implement parallel simulated annealing and parallel tabu search in workstation cluster system to be improve cost quality and computation time and assesed the performance parallel meta heuristics algorithm.

1 はじめに

近年，組合せ最適化問題に対して各種のメタ戦略の有効性が示されている．メタ戦略とは，組合せ最適化問題を解くためのヒューリスティックを有機的に結合させた近似解法の一種であり，代表的なものに Simulated Annealing[1], Tabu Search などがある．

*Email: g9832@edu.otaru-uc.ac.jp
†Email: kaji@res.otaru-uc.ac.jp

組合せ最適化問題の一種である, グラフ分割問題は頂点と辺で構成されるグラフをカットエッジが最小となるように分割する NP-complete な問題である. その中でもグラフを 3 つ以上の複数に分割を行う多分割な問題は汎用性が高まると同時に複雑度も高い.

本研究では, まずシーケンシャルアルゴリズムにおける Tabu Search, Simulated Annealing の各アルゴリズムの比較を行う. Tabu Search は全ての近傍を探索するため計算時間が膨大であるため, 近傍を分割することで並列化し, 計算時間の短縮を試みる. さらに, 問題を部分問題に分割して各プロセッサに割り当て, 一定回数ローカルな解を生成した後にプロセッサ間で整合をとるアルゴリズムの考察を行う. Simulated Annealing に関しては, 問題を部分問題に分割して各プロセッサに割り当て, 対象となる部分問題に対して一定回数近傍を生成した後プロセッサ間で整合をとるアルゴリズムと, さらにより良い解を求めるために, 上記の近傍生成の過程においていくつかの候補のなかから最良のものを選択する処理を加えたアルゴリズムを提案する.

2 グラフ分割問題の定義

今回対象とするグラフ分割問題は, 頂点集合 V と無向辺集合 E からなるグラフ $D(V,E)$ において, そのカットエッジのコストが最小となるような多分割を求める問題であり, 制約条件として, 分割した各部分集合における頂点のウェイト総和がある値 P_i をこえないものとする. 本問題は多分割のため, 従来研究されている 2 分割に比べて, 汎用的であるが複雑度が高くなる. この問題は以下のように定式化される.

$$\min \sum_{i \in V_i, j \in V_j, i \neq j} e_{ij}$$

$$\text{subject to } \sum_{v \in V_i} w(v) \leq P_i,\ i = 1, ..., k$$

ここで, $w(v)$ は頂点 v に与えられるウェイトとする.

3 シーケンシャルアルゴリズム

3.1 Tabu Search

Tabu Search は Local Search の変形で, 一度交換した頂点を Tabu List に一定期間記憶し, 同じ解の探索を禁止することで解のサイクリングを防ぐ. またコストが改悪される交換も受理することで, 局所最適解から脱出する可能性を持つ.

近傍の生成は, 全ての部分集合の組み合わせの中から頂点を交換して, 最も目的関数値を改善するものを選択し, その頂点を交換することによって解を生成するものとした. Fig.1 にそのアルゴリズムを示す.

3.2 Simulated Annealing

Simulated Annealing は, Kirkpatrick, Gelatt & Vecchi らによって紹介された, 固体の Annealing(焼きなまし) のアナロジーを利用して, 最適化問題を解くものである. Simulated Annealing は, 局所探索をランダムに行い, 解が改悪された場合でも新しい解にある確率で移ることで, 局所最適解に補足されることを防ぐアルゴリズムである.

```
Tabu := φ;
i_0 := Initialize;
TL := positive integer;
repeat
    min_generate(i_{t+1}
      from S_{i_t} \ Tabu);
    Tabu := Tabu ∪ i_t \ {i_{t-TL}};
    t := t + 1;
until stop-criterion
```

Fig. 1: Sequencial Tabu Search

```
i := Initialize;
repeat
    for l := 1 to L_k do begin
        rand_generate(j from S_i);
        if f(j) ≤ f(i) then i := j;
        else
            if (exp((f(i)-f(j))/c_k)) > random[0,1)
              then i := j;
    end;
    k := k + 1;
    L_k := L_{k-1} × LI;
    c_k := c_{k+1} × ci;
until stop-criterion
```

Fig. 2: Sequencial Simulated Annealing

Algorithm	100	300	500
SA	115.9	324.5	546.3
TS	116.0	328.0	650.0

Table. 1: コスト

Algorithm	100	300	500
SA	9.87	10.26	10.40
TS	42.69	450.51	1296.33

Table. 2: 計算時間 (秒)

近傍の生成は, 確率的に2つの頂点を選択し, 採択基準に沿って交換を実行するものとした. Fig.2 にそのアルゴリズムを示す.

3.3 実験結果

シーケンシャルアルゴリズムのコスト比較を Table.1, 計算時間の比較を Table.2 に示す.

今回の実験では, 各アルゴリズムのパラメータは数通りのテストを行い調整したものを採用した.

コストは Tabu Search, Simulated Annealing ともに同等であったが, 時間においては Tabu Search は全近傍を探索するため, 頂点数の増加とともに指数関数的に計算時間が増加する.

4 並列アルゴリズム

4.1 並列実験環境

並列プログラミングには MPI(Message Passing Interface) を利用した. MPI はメッセージ通信のためのライブラリの標準であり, 様々な実装が存在し, 多くのプラットフォームで動作する. [4]

今回は UNIX マシンを 100base-TX と Switching HUB によるネットワークに接続したワークステーションクラスタを構成し, 互いにメッセージ通信を行う分散メモリ型の MIMD(Multiple-Instruction/Multiple-Data) マシンとした.

今回利用した MPI ライブラリは mpich であり, 各ワークステーション間のメッセージの通信路

には100base-TX の Ethernet を利用する.

4.2 実装方法

並列アルゴリズムの実装についてはいくつかの手法が研究されている.

今回の実験環境でMPIの通信パフォーマンステストを行ったところ, 1024バイトブロックの実効転送速度は1.3Mbps程度であった. ブロックサイズを大きくしたピーク時でも23Mbps程度であった. この通信路のバンド幅の狭さはEthernetを利用した並列処理における不利な点であり, ローカルメモリはもちろん, 専用の並列サーバ等のバンド幅と比較しても大きく劣る点である.

ワークステーションクラスタにおいては, 通信の間隔をできるだけ長くし, 通信の粒度を粗くできるようなアルゴリズムでなければならない.

4.3 メタ戦略の並列化へのアプローチ

並列環境で処理を分散させる場合, 大きく分けると機能分割とデータ分割の2つのアプローチがある.

機能分割はパイプライン処理等に代表されるような, 機能そのものを各プロセッサに分散させるものである. データ分割はデータを分割することで, 各プロセッサの計算量を軽減するものである.

データ分割による並列化の試みは C.-N.Fiechter が巡回セールスマン問題で可能解の集合を分割して各プロセッサに割り当て, もっとも良いものを探索する方法や, 独立した部分問題に分割して各プロセッサで移動を行い, グルーピングする方法などを提案している. [5]

また, 機能分割の試みは SAUL A.KRAVITZ, ROB A.RUTENBAR によって, Simulated Annealing の選択や評価のプロセスを分割して並列化する方法などを提案している. [6]

今回扱うグラフ分割問題では, ノードの集合であるブロック単位に各プロセッサに分散させることが容易であり, 機能分割を用いるには通信路のバンド幅が狭いことから, データ分割的手法を適用することとした.

5 並列 Tabu Search

5.1 近傍分割型 Parallel Tabu Search

Tabu Search はシーケンシャルアルゴリズムにおいて優れた性能を示すが, 全ての近傍を探索するため, その探索時間が非常に膨大になる. そこで, 近傍の探索を並列化することによる探索時間の短縮を試みた.

近傍の探索は分割する各ブロックのペアを各プロセッサに割り当て, プロセッサは割り当てられたブロックに含まれるノードの計算のみを行う. プロセッサが担当する探索領域を限定することで, プロセッサの計算量を分散した.

探索はTabuを考慮してプロセッサ *proc* に割り当てられた限定構成された近傍の中から最良な解 j を生成し, それを MPI_Allgather 関数により全プロセッサ間で相互に送信して解を交換し, その中で最良のものを採用する.

Fig.3 に各プロセッサにおけるアルゴリズムと全体的な流れを示す. ただし, p はプロセッサ数とする.

$Tabu := \phi$;

i_0 := Initialize;

TL := positive integer;

repeat

assign_block($S_{i_{proc}}$ from S_{i_t});

{ 各プロセッサに探索領域を割り当てる }

min_generate(j_{proc} from $S_{i_{proc}} \setminus$ Tabu);

{ 割り当てられた探索領域から解を求める }

MPI_Allgather(j, $j_{proc}; proc = 1...p$);

{ 全プロセッサ間で求めた解を交換する }

$i_{t+1} := \min\{j_{proc};\ proc = 1, ..., p\}$;

Tabu := Tabu $\cup\, i_t \setminus \{i_{t-TL}\}$;

$t := t + 1$;

until stop-criterion

end;

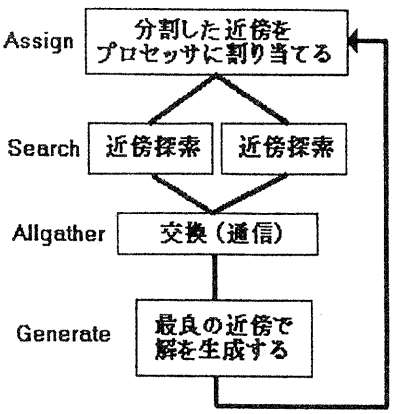

Fig. 3: 近傍分割型 Parallel Tabu Search

5.2 部分問題型 Parallel Tabu Search

上記の方法では, 探索毎に各プロセッサで通信を行う必要がある. そのため通信コストが膨大になり, 効率的な並列化とは言えない. そこで, 一定回数の交換を各プロセッサが割り当てられたブロックに関してローカルで行い, その後に各プロセッサ間で通信を行い整合を取るものとした.

このアルゴリズムでは, まず各ブロックのペアを各プロセッサに割り当て部分問題を生成する. プロセッサは割り当てられた部分問題において交換を一定回数行い, その後に交換ノードを他のプロセッサへ通信して解の整合をとるものとした. Fig.4 に各プロセッサにおけるアルゴリズムと全体的な流れを示す.

5.3 実験結果

各並列 Tabu Search アルゴリズムのコスト比較を Table.3, 計算時間の比較を Table.4 に示す.

シーケンシャルの Tabu Search と比較して, 近傍分割型 Parallel Tabu Search は計算時間が約半分に改善された. また, 部分問題型 Parallel Tabu Search に関しては, 近傍分割型 Parallel Tabu Search と比較して計算時間はほぼ同等であるが, コストは改善されていることが観察された.

$Tabu := \phi$;

i_0 := Initialize;

TL := positive integer;

repeat

assign_block($S_{i_{proc}}$ from S_{i_t});

{ 各プロセッサに探索領域を割り当てる }

repeat

min_generate(j_{proc} from $S_{i_{proc}} \setminus$ Tabu);

{ 割り当てられた探索領域で解を生成する }

Tabu := Tabu $\cup\, i_t \setminus \{i_{t-TL}\}$;

until loop-criterion

MPI_Allgather(j, j_{proc}; $proc = 1...p$);

{ 全プロセッサ間で求めた解を交換して整合を取る }

$t := t + 1$;

until stop-criterion

end;

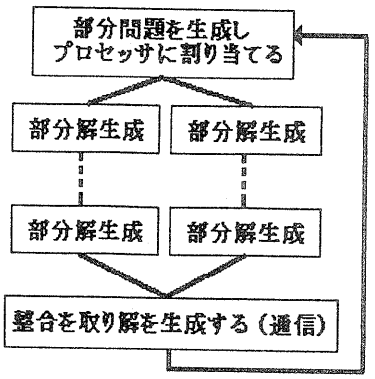

Fig. 4: 問題分割型 Parallel Tabu Search

Algorithm	100	300	500
Sequencial TS	116.0	328.0	550.0
近傍分割 TS	118.0	331.0	552.0
部分問題 TS	119.0	317.0	524.0

Table. 3: 並列 TS コスト

Algorithm	100	300	500
Sequencial TS	42.69	450.51	1296.33
近傍分割型 PTS	21.29	220.90	633.35
部分問題型 PTS	26.16	236.77	665.14

Table. 4: 並列 TS 計算時間 (秒)

6 並列 Simulated Annealing

6.1 Parallel Simulated Annealing

Simulated Annealing は計算時間は非常に優れており, 有効な結果を得た. そこで, Simulated Annealing に対して複数の探索を行うことによってさらに有効な解を探索するデータ分割型の並列アルゴリズムを構成する.

Simulated Annealing は 1 回の近傍生成に要する時間が非常に短く, Tabu Search のように単純な近傍分割のみでは通信コストのほうが計算時間を上回り, 効果的な結果を得ることは難しい.

そこで, 部分問題を生成して各プロセッサに割り当て, それぞれが一定回数ローカルで交換を行った後, 通信を行って整合をとるものとした. Fig.5 に各プロセッサにおけるアルゴリズムと全体的な流れを示す.

$i :=$ Initialize;

repeat

assign_block($S_{i_{proc}}$ from S_{i_t});

for $l := 1$ to L_k do begin

repeat

rand_generate(j_{proc} from $S_{i_{proc}}$);

if $f(j_{proc}) \leq f(i_{proc})$

then $i_{proc} := j_{proc}$;

else

if $(\exp\left(\frac{f(i_{proc})-f(j_{proc})}{c_k}\right) > random[0,1)$

then $i_{proc} := j_{proc}$;

MPI_Allgather(i, $i_{proc}; proc = 1...p$);

until loop-criterion

end;

$k := k + 1$;

$L_k := L_{k-1} \times LI$;

$c_k := c_{k+1} \times ci$;

until stop-criterion

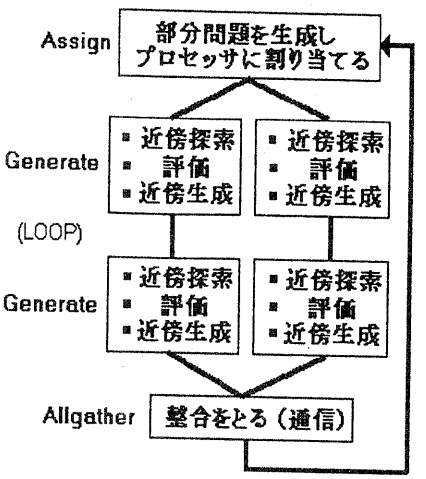

Fig. 5: Parallel Simulated Annealing

6.2 Parallel Greedy Simulated Annealing

上記の Parallel Simulated Annealing に加えて, 計算時間を抑えつつ更なるコストの改善を得るために, 解生成のメカニズムに改良を加えた Parallel Greedy Simulated Annealing を提案する.

Parallel Greedy Simulated Annealing では, 解を生成するときに複数の候補を選び出す. この候補の中で最もコストを改善するものを採用し, それを Simulated Annealing の採用基準をもとに採用する. これを一定回数繰り返した後, プロセッサ間で整合をとる.

基本的なアルゴリズムは上記の Parallel Simulated Annealing と同じであるが, Parallel Greedy Simulated Annealing の特徴である近傍生成の流れを Fig.6 に示す.

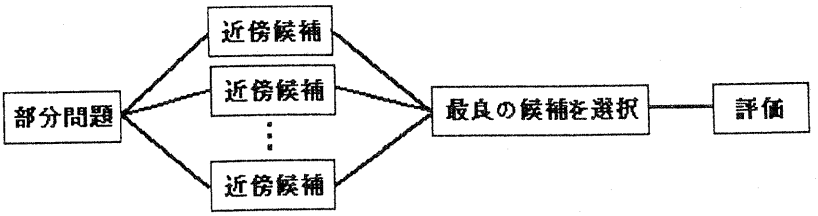

Fig. 6: Parallel Greedy Simulated Annealing

Algorithm	100	300	500
Sequencial SA	2552.5	7545.4	12526.3
P-SA	2524.0	7390.2	12363.5
PG-SA	2479.6	7152.0	11817.4

Table. 5: 並列SA コスト

Algorithm	100	300	500
Sequencial SA	9.87	10.26	10.40
P-SA	18.08	23.08	30.18
PG-SA	88.37	113.11	153.23

Table. 6: 並列SA 計算時間 (秒)

6.3 実験結果

各並列 Simulated Annealing アルゴリズムのコスト比較を Table.5, 計算時間の比較を Table.6 に示す.

シーケンシャルの Simulated Annealing と比較して, Parallel Simulated Annealing は時間は増加したものの, コストが改善された. また, Parallel Greedy Simulated Annealing においては大幅なコストの改善が観察された.

7 おわりに

本研究では Tabu Search, Simulated Annealing のシーケンシャルアルゴリズムの評価を行い, Tabu Search, Simulated Annealing の各並列アルゴリズムを提案した. その結果, Tabu Search ではコスト改善と大幅な時間短縮に成功し, Simulated Annealing においてもコストの改善が得られた.

今回の実験では2台のワークステーションクラスタという小規模な並列環境での実験であったが, 今後更に大きなクラスタでの実験, また異なる特性のグラフに関する実験や, アルゴリズムの更なる改善を予定している.

参考文献

[1] Emile Arts,Jan Korst: Simulated Annealing and Boltzmann Machines, JOHN WILEY & SONS, 1989.

[2] 丸田, 加地: "グラフ分割問題に対するメタ戦略の有効性評価", 平成10年度電気関係学会北海道支部連合大会, pp.315, 1998.

[3] 丸田, 加地: "メタ戦略の評価と並列アルゴリズムへのアプローチ", 情報処理北海道シンポジウム '99, pp.14-15, 1999.

[4] Message Passing Interface Forum: メッセージ通信インターフェイス標準 (日本語訳ドラフト), http://www.ppc.nec.co.jp/mpi-j/index.html, 1996.

[5] C.-N.Fiechter: A parallel tabu search algorithm for large traveling salesman problems, Discrete Applied Mathematics 51, pp243-267, 1994.

[6] SAUL A.KRAVITZ and ROB A.RUTENBAR: Placement by Simulated Annealing on a Multiprocessor, IEEE TRANSACTIONS ON COMUTER-AIDED DESIGN, VOL. CAD-6, NO.4, July 1987, pp.534-549, 1987.