Tabu Search のグラフ分割問題への適用と実験的解析

藤沢　克樹 (早稲田大学)

久保　幹雄 (東京商船大学)

森戸　晋 (早稲田大学)

An Application of Tabu Search to the Graph Partitioning Problem and its Experimental Analysis

Katsuki Fujisawa (Waseda University)

Mikio Kubo (Tokyo University of Mercantile Marine)

Susumu Morito (Waseda University)

In this paper, we report on an application of tabu search to the graph partitioning problem which has applications on circuit board wiring and program segmentation. We discuss how to adapt tabu search to the graph partitioning problem and compare the performance with simulated annealing, another variant of local search incorporating randomized technique. Numerical experiments show that our algorithm dominates the simulated annealing algorithm in accuracy of solutions and speed on both uniform and geometric instances. In particular, our tabu search implementation works much better than the simulated annealing algorithm on structured (geometric) instances. We also investigate how to tune up our implementation and to optimize the various parameters via extensive numerical experiments.

キーワード：Tabu Search, Simulated Annealing, グラフ分割問題, 数値実験

1　はじめに

グラフ分割問題は，$\mathcal{NP}-$ 困難な組合せ最適化問題であり，プログラムのセグメント化や VLSI の回路設計に重要な応用を持つため，多くの研究者によって研究されている [5, 7, 10]。

ここでは，次のように定義される問題を考える。

(グラフ分割問題)

無向グラフ $G = (V, E)$ (V は 点，E は 枝 の集合であり，$n = |V|$ は偶数とする) が与えられたとき，$L \cap R = \emptyset$, $L \cup R = V$ を満たす対 (L, R) を V の分割と呼び，L は 左点集合，R は 右点集合と呼ぶ。特に，$|L| = |R| = n/2$ のとき (L, R) を一様分割と言う。グラフ分割問題 の目的は，$c(L, R) = \{$ L と R 間にまたがる枝の本数 $\}$ を最小にする一様分割 (L, R) を求めることである。

グラフ分割問題に対する最近の重要な成果として，Johnson et al. [5] の研究があげられる。Johnson et al. は，グラフ分割問題 に Simulated Annealing [8] を適用して，Kernighan and Lin [7] の近似解法などの従来法と比較実験を行っている。

一方，Simulated Annealing を上回る解法として期待されている近似解法のパラダイムに Tabu Search [1, 2] がある。Tao and Zhao [9] は、拡張したグラフ分割問題において、Tabu Search, Simulated Annealing,

Kernighan and Lin などの比較実験を行っている。グラフ分割問題の一般化した問題に適用した例に [11] などがある。

しかし Tabu Search に関する系統的な数値実験を行なった研究はほとんどなく，その性質や性能については，いまだ明らかにされていない点が多い。そこで，本研究では系統的な数値実験を通して Tabu Search のグラフ分割問題での有効性を示すとともに，パラメータの適正化に関する知見を得る。また，Johnson et al.[5] で使用されたものと同じデータを用いて Simulated Annealing と Tabu Search の比較実験を行う。

2 Tabu Search の概要

まず一般の組合せ最適化問題を用いて，Tabu Search の概要を説明する。

組合せ最適化問題 $\min\{c(x) \mid x \in X\}$ (ここで $c : X \to \Re$ は費用関数を表す写像, X は実行可能解の集合) に対して近傍を以下のように定義する。

$$N : X \to 2^X$$

ある解に対する近傍 N の中で費用関数を減少させる解が存在しないとき, その解を局所最適解と呼ぶ。Tabu Search は Glover [1, 2] によって提案され, 近傍 N の中から最も費用関数値の減少量が大きい解 (減少量が負, すなわち改悪になることもある) に移行していくとともに，局所最適解から脱出するために禁止リスト TL を用いて一度移行した解には, 一定回数以上移行しないように制限するアルゴリズムである。Tabu Search において現在の解 x から次の解に移行するには, 以下に示す関数 $move$ を使用する。

$$move(x) = \begin{cases} x' & \text{if } c(x') \leq c(y) \text{ for all } y \in N(x) \setminus TL \\ \emptyset & \text{if } N(x) \setminus TL = \emptyset \end{cases}$$

procedure Tabu Search

1 $t := 0$

2 $x_0 :=$ some initial solution

3 $TL := \emptyset$

4 $Tabu\_Length :=$ a positive integer

5 **while** stopping-criterion $\neq$ yes **do**

6 $\quad x_{t+1} := move(x_t)$

7 $\quad TL := TL \cup \{x_t\} \setminus \{x_{t-\text{Tabu\_Length}}\}$

8 $\quad t := t + 1$

9 **return** x

3 グラフ分割問題への適用

3.1 近傍構造と計算上の工夫

Tabu Search や Simulated Annealing のような Local Search を基礎とするアルゴリズムを構築する上で，最も重要な要素は近傍構造の設計である。

ここでは，以下の 2 つの近傍を用いる。分割 (L, R) において，L から R に点を 1 つだけ移動することによって得られる分割を left-to-right 近傍 と呼び，逆に，R から L に 点 を 1 つだけ移動して得られる分割を right-to-left 近傍と呼ぶ。これらの近傍は Johnson et al. と同じである。

left-to-right 近傍の中で，最も費用関数を減少させるものへの移行を left-to-right move と呼び，$\overrightarrow{N}$ と定義し，同様に，right-to-left 近傍をもとにした移行を right-to-left move と呼び，$\overleftarrow{N}$ と定義する。

また，$\overrightarrow{N}$ を k 回行った後に，$\overleftarrow{N}$ を k 回行う操作を k-move と呼ぶことにする。基本的には，1-move を行うが，局所最適解からの脱出力を高めるために $k \geq 2$ の k-move も採用している。§ 4 の実験では，1-move と 2-move を交互に切り替えて行うようにしている。Johnson et al. は、ペナルティー関数方式を採用しており、実行可能性が常に保証されないのに対して、k-move 終了時点では、常に実行可能性が保証されており、これが Johnson et al. と異なる本研究の特徴である。

効率的な Tabu Search を設計するためには，費用関数の変化量を高速に計算しなければならない。そのために，S と D という補助配列を用意する。

$S(i)$ $(D(i))$ には，点 i と同じ (異なる) 点集合に属する点との間にある枝の本数を保持する。すると，点 i を移行させたときの費用の変化量 $\delta(i)$ は，$\delta(i) = S(i) - D(i)$ と計算することができる。また，ある点を移行させたとき，その点に関しては $S(i), D(i)$ の値を交換し、その点との間に枝を持つ全ての点に対して、$S(i), D(i)$ を以下のように更新する。移行前の集合に含まれる点 i に対しては、$S(i) = S(i) - 1$, $D(i) = D(i) + 1$ とし，移行後の集合に含まれる点 i に対しては $S(i) = S(i) + 1$, $D(i) = D(i) - 1$ とする。この工夫により，1 反復あたりの計算量は $O(n)$ となる。

3.2 禁止リストについて

Tabu Search で用いられる禁止リスト TL は，通常，FIFO (First-In-First-Out) の性質を持った Queue 構造である。ここでは，より柔軟な禁止リストの実現方法として，寿命表 (Life Span Table) LS をとる方法を用いる。ここで LS は，点の要素に対応した 1 次元配列であり，配列の添字は点番号に対応している。

ある点 i を別の点集合に移動させたとき，$LS(i)$ に定数を代入する。この数が，Tabu Search における禁止リストの長さ $Tabu\_Length$ に対応する。はじめに，LS には 0 を代入しておき，以後 $LS(i) > 0$ となる全ての $LS(i)$ は 1 反復ごとに 1 だけ減らしていく。そして，$LS(i) > 0$ となっている 点 i は禁止リスト に入っているものとみなして，その選択を禁止する。

寿命表を用いることによって，禁止リストに入っているか否かの判定が $O(1)$ で可能になる。また，$Tabu\_Length$ をランダムにするなどの工夫を導入することも容易になる。

3.3 長期メモリ

上で導入した禁止リストを用いることによって短期的な巡回を阻止することが可能になるが，ここでは，長期的な巡回を防ぐために導入された長期メモリ (long-term memory) [1, 2] について述べる。

長期メモリは，禁止リストと異なり，探索が終了するまでクリアされることなく保持される。また 禁止リストのように特定の点の移動を禁止するのではなく，頻繁に移動を繰り返す点に対して移動しにくくさせるのが目的である。長期メモリも，禁止リスト と同じく n 個の要素を持った配列 $\overrightarrow{M}$ と $\overleftarrow{M}$ を用いる。$\overrightarrow{M}(i)$ には，点 i に関する L から R への移動回数を保持し，同様に $\overleftarrow{M}(i)$ には，R から L への移動回数を保持する。

点 i を移動したときの費用関数の変化量 $\delta(i)$ は，長期メモリを用いて次のように変更される。

$$\overrightarrow{N}\text{のとき} : \delta(i) = S(i) - D(i) + BIAS \times \overrightarrow{M}(i)$$
$$\overleftarrow{N}\text{のとき} : \delta(i) = S(i) - D(i) + BIAS \times \overleftarrow{M}(i)$$

ここで，$BIAS$ は，長期メモリ対するパラメータであり，予備的な実験により適正化するものとする。

3.4 局所最適解からの再出発

Tabu Search においては，禁止リストを用いて，近傍を制限しているため，本来の意味での局所最適解を見つけることができない場合がある。これを避けるための工夫として以下の方法をとった。ある反復回数 ($Clear\_Count$) の間に費用関数の値が更新されないときは，現在までに見つかった最良解に戻り，禁止リストの中身を全てクリアした後で，探索を開始する。アルゴリズムの効率は，パラメータ $Clear\_Count$ の設定に左右されるので，予備的な数値実験によってあらかじめ適正化する。

3.5 終了判定基準

終了判定基準に、パラメータ $Stop\_Time$ を用いる。Tabu Search においては，アルゴリズムの途中までに発見された最小費用を保持しておく。この値が更新されなくなってからの経過時間が $Stop\_Time$ を越えたときに，アルゴリズムは終了する。

3.6 Tabu Search アルゴリズムの概要

まず，アルゴリズムで用いるサブルーチンの概要を示す。

READ_INSTANCE(): データを読み込み，データ構造を組立てる。また $Tabu\_Lengh$ (禁止リストの長さ), $Stop\_Time$ (終了条件), $Clear\_Count$ (再出発までの反復回数) などのパラメータも読み込む。

INIT_CALC(): 全ての点 i について $S(i)$ と $D(i)$ を初期計算する。

UPDATE_SD() ある点が別の点集合に移動したときに，$S(i)$ と $D(i)$ を更新する。

MOVE_LR(): 点集合 L のなかで (禁止リストに入っているものは除く) R に移動したときに最も費用関数を減少させる点を選択して R に移動する。

MOVE_RL(): MOVE_LR() と同様に，R から L に点を移動する。

BACKUP_DATA(): 最良解が更新されたとき，現在の解，目的関数値，$S(i)$, $D(i)$ を保存する。

RESTORE_DATA(): 保存しておいた最適解，$S(i)$, $D(i)$ を復元する。

CLEAR_LIFESPAN(): 禁止リスト LS の中身をクリアする。

TIME_PROCESS(): 変数 $time$ が 0 に設定されてからの経過時間を測定して、$time$ に代入する。

次に，グラフ分割問題に対する Tabu Search の概要を示す。

procedure グラフ分割問題に対する Tabu Search

```
 1 READ_INSTANCE()
 2 INIT_CALC()
 3 swapmode := 2  (2-move と 1-move を交互に行う)
 4 time := bestcount := 0
 5 bestcut := ∞  (現在までに見つかった最良解の値)
 6 while  time < Stop_Time do
 7     movecount := 0
 8     while  movecount < swapmode do
 9         MOVE_LR()
10         UPDATE_SD()
11         movecount := movecount + 1
12     movecount := 0
13     while  movecount < swapmode do
```

```
14          MOVE_RL()
15          UPDATE_SD()
16          movecount := movecount + 1
17      if  c(L, R) ≤ bestcut then
18          bestcut := c(L, R)
19          bestcount := 0
20          time := 0
21          BACKUP_DATA()
22      else
23          bestcount := bestcount + 1
24      if  bestcount > Clear_Count then
25          bestcount := 0
26          RESTORE_DATA()
27          CLEAR_LIFESPAN()
28          if  swapmode := 2 then
29              swapmode := 1
30          else
31              swapmode := 2
32      TIME_PROCESS()
33  return bestcut
```

4 数値実験

ここでは，数値実験によって，上で構築した Tabu Search の性能評価および挙動分析を行う。

また，従来の研究における Johnson et al. [5] の実験との比較をするために，全く同じ問題を用いる。Johnson et al. が数値実験に使用したグラフは，次の 2 種類である。

ランダムグラフ: 点の数 n, 枝の存在確率 p $(0 < p < 1)$ を入力とし，任意の 2 点間に確率 p で枝を引くことによって無向グラフを構成する。1 つの 点 から出ている 枝の平均本数 (平均次数) は，約 $p(n-1)$ である。

幾何学的グラフ: 点の数 n, パラメータ d を入力とし，単位正方形 $[0,1]^2$ 内に一様かつ独立に n 個の 点を分布させ，2 点間の距離が d 以下のときに枝を引くことによって無向グラフを構成する。平均次数は，約 $n\pi d^2$ になる。

Johnson et al. が実験に用いたデータは，ランダムグラフ が 16 個，幾何学的グラフが 8 個であり，主に，Simulated Annealing，Kernighan and Lin，Local Search の挙動解析と比較を行っている。要約すると，Johnson et al. は次のような結果を得ている。ランダムグラフにおいては，Simulated Annealing が良く，構造を持った幾何学的グラフにおいては，Kernighan and Lin またはその変形法が，Simulated Annealing を大

幅に上回っている。

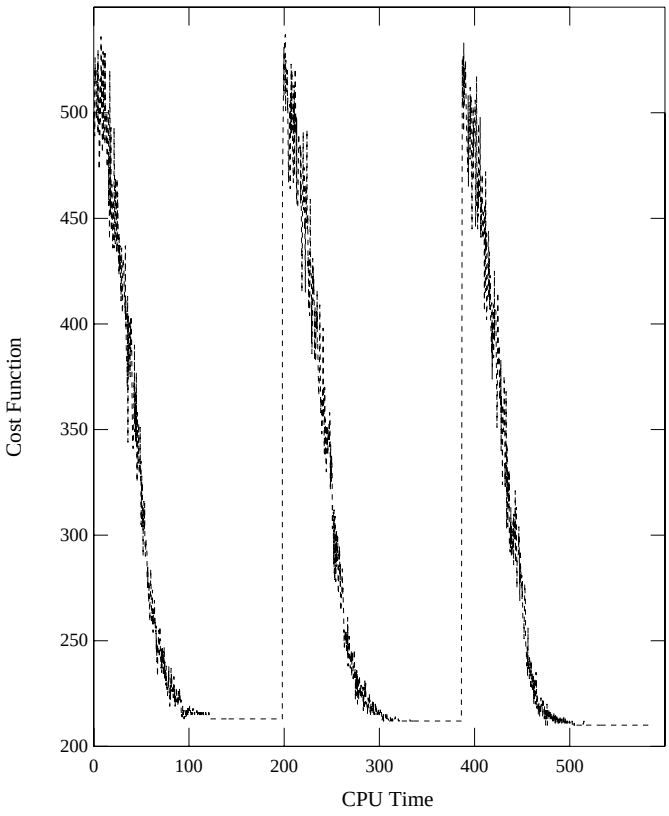

図 1: Annealing2 の実行時における費用関数の変化

Figure 1: The evolution of the cost function during Annealing2.

ここでは，提案した Tabu Search をもとにしたアルゴリズムが，ランダムグラフと幾何学的グラフの両方のグラフ上で効率的に働くことを示す。なお，全ての実験は SUN SPARC station IPX 上で行い，プログラム言語は GNU-C である。

同じ計算機環境下での比較を行うために，[5, 6] をもとにして Johnson et al. の Simulated Annealing と同等かそれ以上の性能を示すプログラムを作成した。本研究で作成した Simulated Annealing (以下，Annealing2 と呼ぶ) は，アルゴリズムに関しては，ほぼ Johnson et al. と同じものであり，パラメータ設定に関しても [5] で述べられているパラメータの値の範囲を参考にして設定を行っている。しかし，Johnson et al. の Simulated Annealing では温度がある程度下がると局所最適解に達したとして探索を終了してしまうが，Annealing2 では，現在の解を初期解として、温度を再加熱して再出発する方法を採用する。なお Ingber [3, 4] は、パラメータの再設定も含めた再加熱の研究を行なっている。

はじめに，図 1,2 に Annealing2 と Tabu Search における計算時間と費用関数値の変化の様子を示す。なお，以下の挙動解析に関する実験では，Johnson et al. がパラメータ設定に用いた $n = 500, p(n-1) = 5.0$ のランダムグラフを用いる。

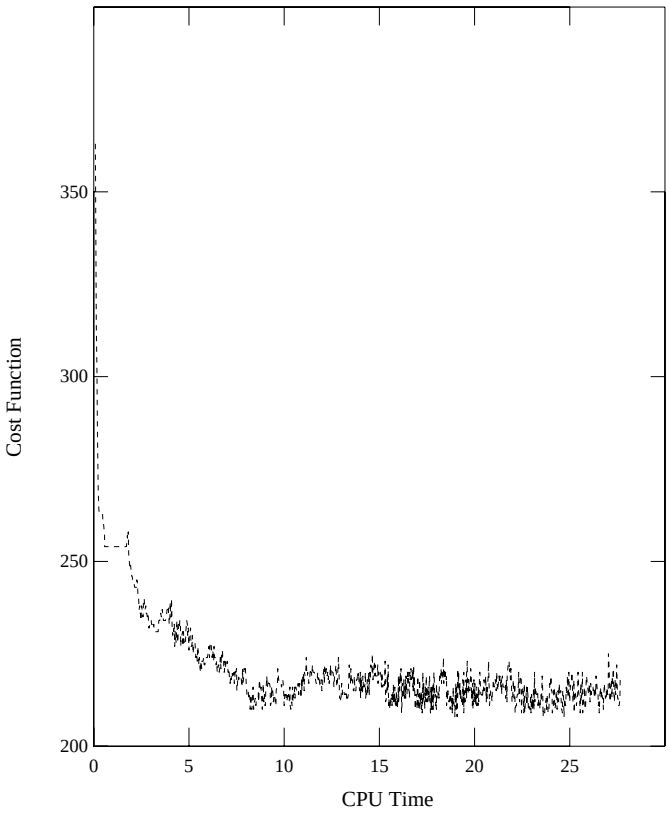

図 2: Tabu Search の実行時における費用関数の変化

Figure 2: The evolution of the cost function during Tabu Search.

図 1 は，Annealing2 の 計算時間と費用関数値の関係を示したもので，費用関数値は，図 2 と比較した場合、横軸のスケーリングが異なるために実際は Annealing2 のほうが緩やかに減少をしている。再加熱を行なうと，ここでは 1 回目の初期設定温度にまで温度を上げているので費用関数値は急激に上昇し，また減少をはじめる。この例では，再加熱を 2 回おこなっているので，似たような動きが全部で 3 回みられる。収束までの時間は，段々と短くなっており，収束値も $216, 213, 210$ と順々に減少している。

ここで再加熱の有効性を確認するために、さらに追加実験を行なう。ここでは、再加熱を 4 回行なうが、実行時間的には再加熱を行なわないで、ランダムな初期解から 5 回連続実験を行なうのとほぼ等しくなるので、再加熱を 4 回行なうのと、再加熱を行なわずに 5 回連続実験を行なうのをそれぞれ 1 組にして、100 組の数値実験を行ない、費用関数値の平均、最大、最小を比較することにした。また再加熱時の温度設定は、予備実験の結果より低めに設定すると、あまり再加熱の効果がみられないので、初期設定温度と同じに設定した。実験で用いるグラフは、ランダムグラフと幾何学的グラフから 2 つずつ点数 500 と 1000 のグラフを選択した。

表 1: Annealing2 における再加熱の効果 (表内の数字は目的関数値.5–SA はランダムな初期解から 5 回 SA を行ったもの. 再加熱 SA は 1 つの初期解から 4 回再加熱を行ったもの. 試行回数は 100 回.)

Table 1: Effect of Re-annealing on the average, maximum, minimum cost function.

	5−SA			再加熱 SA		
グラフ-点数	平均	最大	最小	平均	最大	最小
ランダム-500	213.6	227	206	210.4	215	206
ランダム-1000	3397.9	3447	3382	3386.9	3402	3382
幾何学的-500	252.8	456	178	226.3	432	178
幾何学的-1000	40.1	66	22	30.7	42	16

表 1 では、試行回数が 100 回と多いため最小値は再加熱の有無であまり差が出ていないが、平均値、最大値では、再加熱を行なったほうが良い結果が出ている。

図 2 に Tabu Search における計算時間と費用関数値の関係を示す。収束するまでの実行時間は，Tabu Search は約 25 秒, Annealing2 は約 200 秒と Tabu Search 方がかなり短い。Annealing2 は，2 回再出発を行って，最良解 210 を得たが，Tabu Search では 1 回の試行で 最良解 206 を得ている。他のグラフに対しても同じような挙動が観察できるが，全てのグラフに対する比較については，後で詳しく触れる。

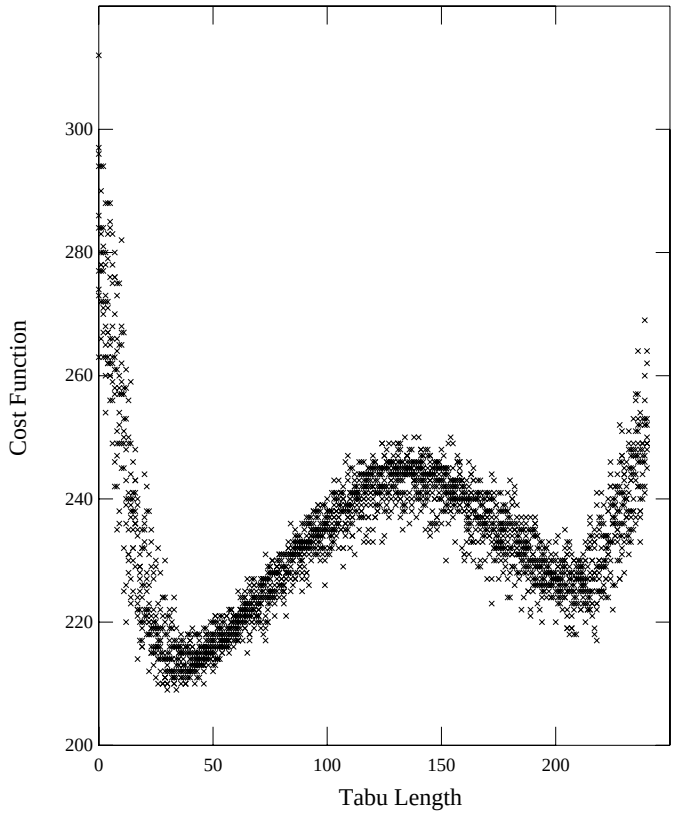

図 3: 禁止リストの長さによる解の値の変化 (他の工夫は導入しないとき)

Figure 3: Effect of *Tabu_Length* on the cost function (without using other parameters).

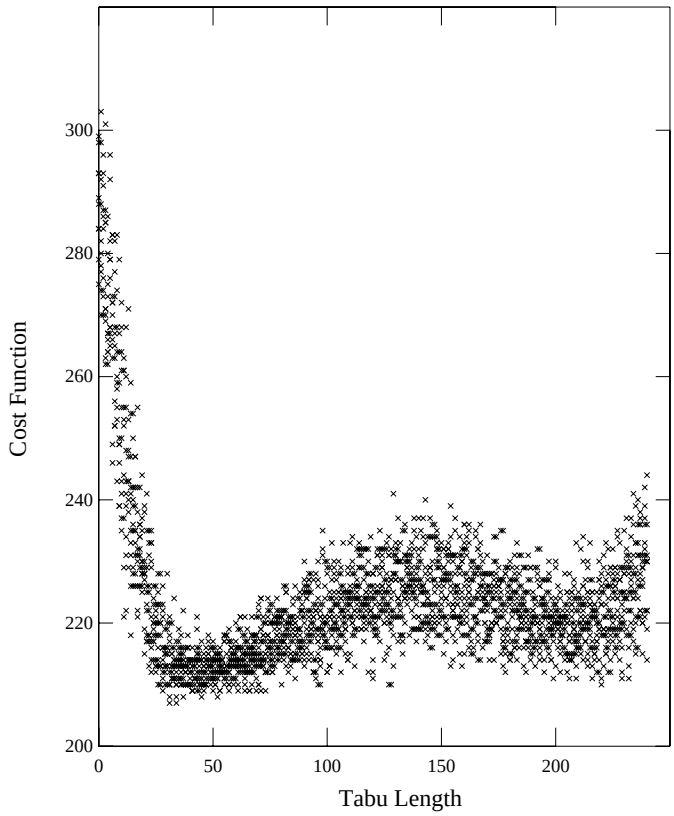

図 4: 禁止リストの長さによる解の値の変化 ($Clear\_Count = 2000$)

Figure 4: Effect of $Tabu\_Length$ on the cost function ($Clear\_Count = 2000$).

次に，Tabu Search におけるパラメータの適正化と挙動分析を目的とした実験の結果を示す。

Tabu Search を効率的に動かすために最も重要なパラメータは，禁止リストの長さ ($Tabu\_Length$) である。禁止リストのみを用いたときの，$Tabu\_Length$ の変化に対する費用関数値の変化を図 3 に示す。図 3 では，禁止リストの長さを 0 から増やしていくと，費用関数値は減少し，25 ～ 30 あたりで費用関数値が最小になり，増加をはじめるが 150 を越えたあたりから再び減少をはじめている。この例では，禁止リストの適正値は，約 30 であり，他のグラフでも同様の実験をすることによって，適正値を求めることができる。また $BIAS$ や $Clear\_Count$ などのパラメータも同様に適正値を求めることができる。

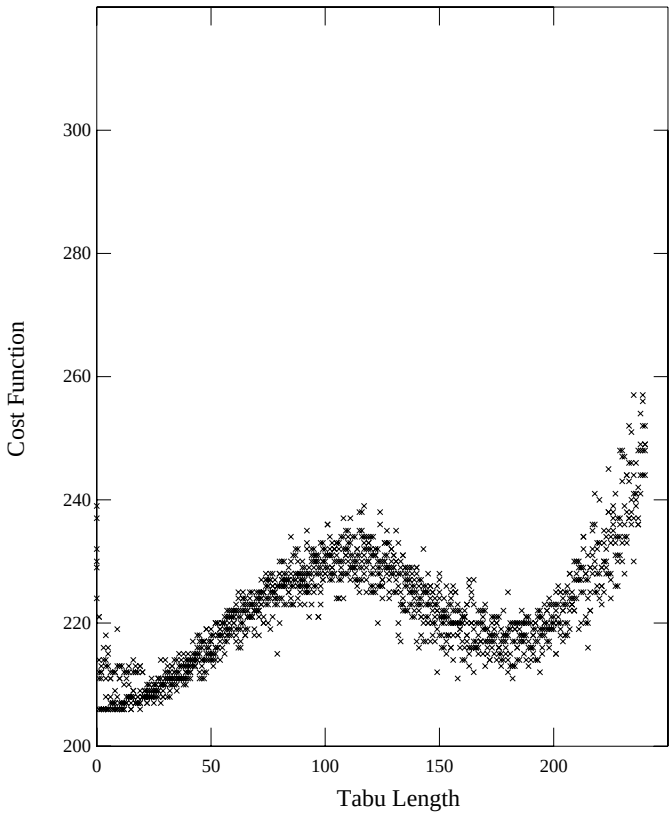

図 5: 禁止リストの長さによる解の値の変化 ($BIAS = 2$)

Figure 5: Effect of $Tabu\_Length$ on the cost function ($BIAS = 2$).

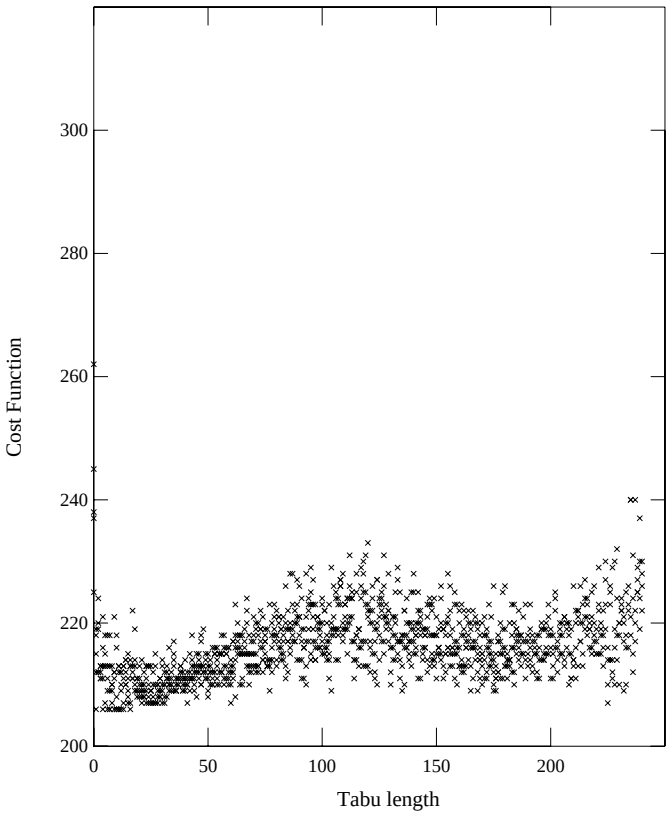

図 6: 禁止リストの長さによる解の値の変化 ($BIAS = 2$, $Clear\_Count = 2000$)

Figure 6: Effect of $Tabu\_Length$ on the cost function ($BIAS = 2$, $Clear\_Count = 2000$).

次に、再出発の有効性を確認するために $Clear\_Count = 2000$ に設定して上と同じ実験を試みた。結果を図 4 に示す。図 3 と 図 4 を比較してみると全体的に平均値、最良値ともに改良されているのが分かる。

次に、長期メモリの有効性を確認するために、$BIAS = 2$ に設定して上と同じ実験を試みた。結果を図 5 に示す。

図 3 と 図 5 を比較してみると再出発と同様に、全体的に平均値、最良値ともに改良されているが、特に禁止リストの値が小さい範囲において効果が著しいのが分かる。最後に、長期メモリと再出発の両方の採用して有効性を確認するために，$BIAS = 2$, $Clear\_Count = 2000$ に設定して，上と同じ実験を試みた。結果を図 6 に示す。図 3,4,5,6 より長期メモリと再出発の両方を組み込んだ場合には、全体的に安定して、良い結果を示しているが、特に、長期メモリの効果によって図 6 では、禁止リストの適正値が $10 \sim 20$ 付近に移動しているのがわかる。

以上の結果より、禁止リスト、長期メモリ、再出発の工夫は、パラメータを適正値付近に設定すれば、かなりの効果をあげているといえる。

表 2: 16 個のランダムグラフ上での費用関数値の平均

Table 2: Average Cost Function for 16 Random Graphs

	$p(n-1)$				
n	2.5	5.0	10.0	20.0	Algorithm
124	13.2	63.6	178.5	454.4	Annealing2
	13.0	63.0	178.0	449.0	Tabu Search
250	30.9	115.1	358.6	829.9	Annealing2
	29.5	114.2	357.7	828.2	Tabu Search
500	55.0	222.5	630.4	1749.9	Annealing2
	51.2	219.4	628.9	1745.2	Tabu Search
1000	105.1	456.8	1373.0	3389.3	Annealing2
	102.6	451.6	1366.3	3387.8	Tabu Search

表 3: 16 個のランダムグラフ上での計算時間 (秒) の平均

Table 3: Average Running Times in Seconds for 16 Random Graphs

	$p(n-1)$				
n	2.5	5.0	10.0	20.0	Algorithm
124	23.3	17.8	11.4	15.7	Annealing2
	0.9	1.9	1.8	1.5	Tabu Search
250	57.3	70.2	81.8	67.2	Annealing2
	6.3	9.6	8.6	19.7	Tabu Search
500	253.5	239.0	230.1	181.0	Annealing2
	180.0	117.1	68.7	54.3	Tabu Search
1000	1288.7	1206.5	681.5	905.0	Annealing2
	458.1	265.8	158.6	98.3	Tabu Search

表 4: 8 個の幾何学的グラフ上での費用関数値の平均

Table 4: Average Cost Function for 8 Geometric Graphs

	$n\pi d^2$				
n	5	10	20	40	Algorithm
500	18.2	50.2	220.6	693.6	Annealing2
	2.2	26.0	178.0	412.0	Tabu Search
1000	34.0	92.4	350.5	1081.2	Annealing2
	2.7	39.0	222.0	737.0	Tabu Search

表 5: 8 個の幾何学的グラフ上での計算時間 (秒) の平均

Table 5: Average Running Times in Seconds for 8 Geometric Graphs

	$n\pi d^2$				
n	5	10	20	40	Algorithm
500	184.2	181.0	101.7	133.2	Annealing2
	89.2	40.4	25.3	7.7	Tabu Search
1000	987.4	1163.4	662.7	337.2	Annealing2
	265.3	156.5	49.7	10.2	Tabu Search

他のデータにも同様な予備実験を行い，各パラメータを設定して本実験を行う。各パラメータの適正値は，点数 n が同じならば平均次数にはあまり左右されず，ほぼ同じ値であり，$n = 500$ の場合は，禁止リストの長さ $Tabu\_Length$ は $20 \sim 30$，長期メモリの強さ $BIAS$ は $2 \sim 5$，再出発までの反復回数 $Clear\_Count$ は 2000 程度が推奨できる。n が大きくなると，$Tabu\_Length$, $Clear\_Count$ は，多少大きめに設定する必要があるが，$BIAS$ は，n によらず一定値でもかまわない。

また，Annealing2 における再出発は，その回数が多いほど費用関数値に改善がみられるが，実行時間とのトレードオフから，本実験では 1 回とする。本実験は，各データに対して初期解をランダムに変えて Annealing2 と Tabu Search をそれぞれ 20 回ずつ実験を行い，その費用関数値と実行時間で比較する。

表 2,3,4,5 に，Tabu Search および Annealing2 を初期解を 20 回変えて試行したときの平均値と，平均実行時間をあげる。結果を総合すると Tabu Search は，ランダムグラフと幾何学的グラフともに Annealing2 よりも優れた，そして安定した結果を残している。特に，幾何学的グラフに対しては，Tabu Search は，Annealing2 を大幅に上回る結果を出していることが分かる。

表 6,7 に Tabu Search および Annealing2 を初期解を 20 回変えて実行したときの最良値を示す。表中の best known は，Johnson et al. が，Simulated Annealing, Kernighan and Lin の近似解法, およびその改良法を用いて得た最良解の値を示している。

表 6,7 の結果をまとめると以下のようになる。Tabu Search は，ランダムグラフでは 16 問中 7 問，幾何学的グラフでは 8 問中 2 問，従来の研究における最良解を更新しており，他のグラフについても，最良解と同じ値を算出している。Annealing2 も，ランダムグラフに対しては 16 問中 7 問，最良解を更新しているが，Tabu Search で得た最良解には及ばない。また，幾何学的グラフに対しては，Tabu Search または従来の研究における最良値 (その多くは，Kernighan and Lin のアルゴリズムによって得られた) に及ばない。これは，本研究で作成した Annealing2 は，Johnson et al. の Simulated Annealing より多少良い解を算出するが，同様の欠点 (幾何学的グラフに対して弱い) を持つことを示す。

表 6: 16 個のランダムグラフ上での費用関数値の最良値

Table 6: Best Cost Function for 16 Random Graphs

	$p(n-1)$				
n	2.5	5.0	10.0	20.0	Algorithm
124	13	63	178	449	Annealing2
	13	63	178	449	Tabu Search
	13	63	178	449	best known
250	29	114	357	828	Annealing2
	29	114	357	828	Tabu Search
	29	114	357	828	best known
500	51	218	626	1744	Annealing2
	49	218	626	1744	Tabu Search
	52	219	628	1744	best known
1000	100	450	1364	3382	Annealing2
	96	449	1362	3382	Tabu Search
	102	451	1367	3389	best known

表 7: 8 個の幾何学的グラフ上での費用関数値の最良値

Table 7: Best Cost Function for 8 Geometric Graphs

	$n\pi d^2$				
n	5	10	20	40	Algorithm
500	10	26	178	449	Annealing2
	2	26	178	412	Tabu Search
	4	26	178	412	best known
1000	18	55	222	742	Annealing2
	1	39	222	737	Tabu Search
	3	39	222	737	best known

以上の分析により，本研究で構築した Tabu Search は，グラフ分割問題に対しては従来のどの解法よりも優れ，最良解の探索に使用することのできる解法であると結論できる。

5 おわりに

本研究では，グラフ分割問題に対する Tabu Search の適用を論じた。提案したアルゴリズムは，グラフ分割問題の特性を生かしたデータ構造の工夫，パラメータの設定，探索の工夫が入れられており，従来の方法と比べて優れた近似解法を構築できた。また，系統的な数値実験により，グラフ分割問題に対しては、Tabu Search のパラメータ適正化に対する多くの知見を得た。これらの知見は，他の組合せ最適化問題に対して，本研究のような効率的な Tabu Search を構築するときに役にたつと思われる。

謝辞

最後に，実験データの提供および多くの貴重な助言を戴いた AT & T の David Johnson 博士に感謝致します。

参考文献

[1] F. Glover. Tabu search I. *ORSA Journal on Computing*, 1:190–206, 1989.

[2] F. Glover. Tabu search II. *ORSA Journal on Computing*, 2:4–32, 1989.

[3] L. Ingber. Very fast simulated re-annealing. *Math. Computer Modeling*, 12:967–973, 1989.

[4] L. Ingber. Simulated annealing: practice vs. theory. *Math. Computer Modeling*, 18:29–57, 1993.

[5] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation, part I, graph partitioning. *Operations Research*, 37:865–892, 1989.

[6] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation, part II, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.

[7] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[9] L.Tao. and Y.C.Zhao. Effective heuristics for multi-way graph partition. *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, 1992.

[10] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[11] S.Areibi and A.Vannelli. Circuit partitioning using a tabu search approach. *Proc. IEEE International Symposium on Circuit and Systems*, 1993.