2003－AL－91　(8)
2003／9／19

最小重み点被覆問題に対する並列分枝限定法の性能評価

下田 善隆*　高藤 大介*　田岡 智志*　渡邉 敏正*

[概要] 分枝限定法は，さまざまな組合わせ最適化問題を解くための最も一般的な解法であるが，問題サイズの増加にともない計算時間が指数関数的に増加する．そのため，計算時間短縮を意図した並列化が考えられる．分枝限定法においては，分枝の早い段階においてより最適解に近い暫定解が得られれば，枝刈りが早く起こる可能性が高くなり計算回数を減らすこと，すなわち，解法の高速化につながる．そこで，上界値計算に高性能な既存近似解法を用いることによって計算回数が減少され，計算時間の短縮が期待できる．また，分枝限定法には，問題中のどの変数を固定するか（分枝変数選択）の選択がある．本稿では，最小重み点被覆問題に対する PC クラスタ上での並列分枝限定法について，上界値計算における近似解法使用の効果，分枝変数選択の効果を計算機実験により検証する．

Performance Evaluation of Parallel Branch-and-Bound Algorithms for the Weighted Vetex Cover Problem

Yoshitaka Shimoda*,　Daisuke Takafuji*,　Satoshi Taoka*　and　Toshimasa Watanabe*

[abstract] Even though a branch-and-bound algorithm (BB for short) is the most general technique to deal with various combinatorial optimization problems, the computation time is likely to be exponential as the size of input increases. Parallelization is one way of improvement. We can expect that using a better upper bound makes bounding operations appear at early stages of computation, possibly resulting in short computation time. Here we consider using an approximation algorithm in computing an upper bound. In this paper, we experimentally evaluate how using an upper bound given by an existing approximation algorithm and variable selection strategies affect the computation time of a parallel BB for solving the minimun cost vertex cover problem on a PC cluster.

1　はじめに

整数計画法等の組合わせ最適化問題は一般に NP-困難であることが知られており，その最適解を求めるために最悪の場合は網羅的な探索を必要とする．そのため組合わせ最適化問題の一般的解法として分枝限定法が古くから研究されている．分枝限定法の基本構造（分枝操作，限定操作，上下界値計算，探索法等）については [4] に集大成されており，本研究で使用する概念や基本性質はこれに基づく．分枝限定法は問題の規模が大きくなると膨大な計算時間を必要とすることがあるため，計算時間短縮の一方法として並列化が考えられる．このとき，より早い時期の暫定値更新によって，

$$\text{加速度} = \frac{1\text{台のプロセッサによる処理時間}}{p\text{台のプロセッサによる処理時間}}$$

で定義される加速度が p 以上となる現象，異常加速，も期待できる．

分枝限定法には，分枝木のどの節点 (部分問題) で分枝を行うか（分枝節点選択），問題中のどの変数を固定するか（分枝変数選択）の二つの選択があり，[7] においてこれらの選択方法による並列分枝限定法が PC クラスタネットワーク上で比較実験されている．

また，分枝限定法においては，分枝の段階においてより最適解に近い暫定解を用いれば，枝刈りが早く起こるようになり計算回数を減らすことが期待でき，結果的に，この解法の高速化が予想される．そこで，上界値計算に高性能な既存近似解法を用いることによる計算回数の減少と，計算時間の短縮を試みる．

本稿では，グラフの最小重み点被覆問題を PC クラスタ上での並列分枝限定法により解く場合について，上界値計算に既存近似解法を適用することや分枝変数選択法の効果を計算機実験により検証する．

2　準備

2.1　諸定義

グラフ $G = (V, E)$ は，有限な空でない点集合 V と辺集合 E とから成る．V，E をそれぞれ $V(G)$，$E(G)$ と表すこともある．辺に向きが付いていないもの（無

*広島大学大学院工学研究科／ Graduate school of Engineering, Hiroshima University

向辺）と，付いているもの（有向辺）があり，2 つの点 u, v（$u, v \in V$，u と v は同一であってもよい）を結ぶ場合，無向辺を (u, v)，有向辺を $\langle u, v \rangle$ と表す．u, v をその辺の端点と呼ぶ．有向辺 $\langle u, v \rangle$ は，u から v への向きであるとする．すべての辺が無向辺（それぞれ有向辺）のグラフを，無向グラフ（有向グラフ）という．本稿では，特に断りがない限り無向辺および無向グラフを，それぞれ辺およびグラフと呼ぶ．

無向グラフ G において，辺 $e = (u, v)$ が G の辺であれば，点 u と点 v は**隣接**しているといい，点 u と辺 e，または点 v と辺 e は**接続**しているという．また，2 つの異なる辺 e_1, e_2 が 1 つの端点を共有しているとき，辺 e_1, e_2 は隣接しているという．2 つの異なる辺 e_1, e_2 がともに点 u, v と接続しているとき，すなわち，$e_1 = (u, v), e_2 = (u, v)$ であるとき，これらの辺を**多重辺**という．辺 e がただ 1 つの点 u を両端点としているとき，すなわち，$e = (u, u)$ のとき，辺 e を**自己閉路**という．自己閉路も多重辺も含まないグラフを**単純グラフ**という．

グラフ $G' = (V', E')$ が，$V' \subseteq V$ かつ $E' = \{(u, v) \in E \mid u \in V' \text{ 且つ } v \in V'\}$ であるならば，G' はグラフ G の V' による**誘導部分グラフ**といい，$G[V']$ と記す．

グラフ G において，点と辺の交替列 $P = v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n$ $(v_i \neq v_j,\ 0 \leq i \neq j \leq n)$ が存在するとき，P を点 v_0 から点 v_n への長さ n の**パス**という．但し，$e_i = (v_{i-1}, v_i)$ $(1 \leq i \leq n)$ である．点 v_0 から v_n へのパスに v_0, v_n を両端点とする辺を追加したものを**サイクル**という．サイクルの長さを $n + 1$ とし，その長さが奇数となるサイクルを**奇サイクル**と呼ぶ．

グラフ G において，どの 2 点 u, v に対しても，u と v を端点とするパスが存在するならば，G は**連結**であるという．グラフ $G = (V, E)$ から点集合 $V' \subseteq V$ とそれに接続するすべての辺を取り去ることを，点集合 V' の**除去**といい，取り去った後のグラフを $G - V'$ と表す．

任意の点集合 $V' \subseteq V$ に対し，$Adj_G(V') = \{w \in V - V' \mid v \in V' \text{ and } (v, w) \in E\}$ とおき，$\Delta_G(V') = |Adj_G(V')|$ と記す．特に $V' = \{v\}$ ならば，それぞれ $Adj_G(v)$，$d_G(v)$ と記す．$d_G(v)$ を点 v の点次数と呼ぶ．

分枝限定法とはある問題をいくつかの小規模な問題 (部分問題) に分解し，かつ生成された部分問題に対しても同様に分解を進め (分枝操作)，その全てを解くことで等価的に元の問題を解くような，組み合わせ最適化問題の一般的解法である．3 節で詳細を述べる．

2.2 最小重み点被覆問題

ここでは本稿で扱うグラフの最小重み点被覆問題とそれに対する近似解法について概説する．

グラフの点被覆とは，頂点集合であって，任意の辺の両端点のうち少なくとも一方を含むものである (図 1，2 参照)．点被覆の重みは，点被覆に含まれる各頂点の重み総和である．グラフの最小重み点被覆問題は，自然数の点重みを持つ無向グラフが与えられたとき，重み最小となる点被覆を求める問題である．この問題は，次のように定式化される：

P_0: 目的関数 $|I| \rightarrow$ 最小
制約条件 I は点被覆である

グラフの最小重み点被覆問題は，各頂点の重みが等しい場合であっても，NP-完全であることが知られており [9]，様々な研究が行われている [10-41]. その中の多項式時間の近似解法を表 1, 2 にまとめる．近似解法の「近似比が r である」とは，その近似解法で得られる解が最適解の r 倍以下であることが証明されていることを意味する．

[8] において，点重みが不均一でもよい場合は表 1 のすべての近似解法に対し，点重みが均一の場合には表 1，2 の CLA，BAR，NT，BE，NI に対し，それぞれ計算機実験による性能評価が報告されており，解精度と計算時間から CLA が最も効率的な近似解法であると報告されている．

表 1: 多項式時間の近似解法

近似比	計算量	文献	表記法
2	$O(\|E\| \log \|V\|)$	[17]	CLA
2	$O(\|E\|)$	[11]	BAR
2	$O(\|E\|\|V\|^2)$	[35]	NT
$2 - \frac{\log\log\|V\|}{2\log\|V\|}$	$O(\|E\|\|V\| \log \|V\|)$	[12]	BE
2（期待値）	$O(\|E\|)$	[38]	PIT

表 2: 点重みが均一の場合の多項式時間近似解法

近似比	計算量	文献	表記法
$2 - \frac{8\|E\|}{13\|V\|^2 + 8\|E\|}$	$O(\|V\|\|E\|)$	[34]	NI
$2 - \frac{1}{k+1}$ $(*)$	$O(\|V\|\|E\|)$	[32]	MS

$(*)$ $k : |V| \leq (2k+3)^k(2k+2)$ を満たす最小の整数

3 分枝限定法

分枝限定法は取り扱う問題により手法の細部が異なるため，手法の概要を述べた後で，最小重み点被覆

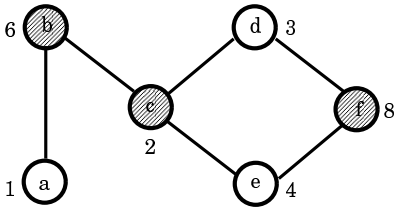

図 1: 点被覆 (黒丸の点) の例．重みは 16.

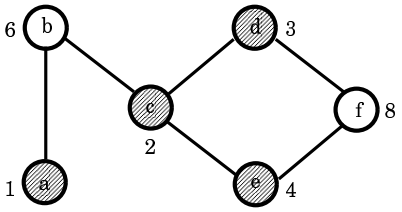

図 2: 最小重み点被覆 (黒丸の点) の例 (最適解)．重みは 10.

問題についての分枝限定法を説明する．

3.1 概要

以下のように定義される問題 P(これを組合わせ最適化問題と考える) を対象とする：

目的関数　$f(\mathbf{x}) \rightarrow$ 最小

制約条件　$\mathbf{x} \in S$，ただし $S \subset X$

$f(\mathbf{x})$ の最大化も考えられるが，以下では断らない限り最小化問題を扱うことにする．

ここで，X は基礎空間と呼ばれ，有限集合または可算無限集合である．具体例としては n 次元整数ベクトル集合やその有限部分集合，有限文字集合 Σ から選んだ文字により生成される有限長文字列の集合 Σ^* やその有限部分集合などである．S は制約条件を集合として表現したもので，$\mathbf{x} \in S$ は x が制約条件を満たすことを表している．$f(\mathbf{x})$ は $x \in S$ に対して整数値や実数値などを与える関数で，$f : S \rightarrow Z$(整数集合) あるいは $f : S \rightarrow R$(実数集合) である．

例題 3.1 与えられたグラフ $G = (V, E)$ の点集合，辺集合をそれぞれ $V = \{1, \ldots, n\}$，$E \subseteq V \times V = \{(i, j) \mid i, j \in V,\ i \neq j\}$ とする．また，$c : V \rightarrow Z^+$(非負整数集合) は各点 $v \in V$ に重み $c(v)$ を与える関数で，$\mathbf{c}^T = (c(1), \ldots, c(n))$ とし，X をすべての n 次元整数ベクトルの集合とする．ここで，

$$S = \{\mathbf{x}^T = (x_1, \ldots, x_n) \in X^T \mid x_i \in \{0, 1\}$$
$$(1 \leq i \leq n) \text{ かつすべての } (j, k) \in E$$
$$\text{に対して } x_j + x_k \geq 1\}$$

とおく．このとき最小重み点被覆問題は

目的関数　$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \rightarrow$ 最小

制約条件　$\mathbf{x} \in S$，ただし $S \subset X$

と記述される．なお，$\mathbf{c}$ がすべての要素を *1* とする n 次元ベクトルの場合が，いわゆる最小点被覆問題である．□

以下で使用するいくつかの用語を説明しておく．**節点**は，分枝木における点のことであり，分枝限定法においては部分問題に相当する．**先端ノード**は，まだ解かれておらず，かつその解操作が打ち切られてもいない部分問題を表す節点である．**未固定変数**は，問題の変数 x_i のうちで分枝操作によって値が固定されていない (あるいは条件判定が終了していない) 変数である．**固定変数**は，分枝操作の際にその値が固定された (あるいは条件判定が終了した) 変数である．**上界値**は，現在解いている部分問題において，目的関数が取り得る値の最大値である．**下界値**は，現在解いている部分問題において，目的関数が取り得る値の最小値である．**暫定値**は，これまでに得られた上界値のうちで最小の値であり，**暫定解**は，その暫定値を与える解である．**枝刈り**は，現在解いている部分問題からこれ以上の分枝操作を行っても暫定解よりも良い解が得られないことがわかった場合，そこからの分枝操作を打ち切ることである．**計算回数**は，分枝操作もしくは枝刈り操作を行った回数である．

3.2 解法の枠組み

分枝限定法は次のようにして実行していく．以下，断らない限り最小化問題として説明する．

基本操作は，原問題から分枝操作を開始し，解が得られるまで分枝操作を続けることである．ここで分枝操作とは，親問題の 1 つの変数固定操作と，それにより部分問題 (子問題) を生成する操作を合わせた操作である．

さらに，分枝限定法の大きな特徴である，ある部分問題に対してそれ以上の分枝操作を打ち切る「限定操作」が下界値を用いて行われる．すなわち，下界値が暫定値より大きいとき，あるいは下界値の計算で部分問題の最適解が求められたときは，この部分問題からのそれ以上の分枝操作は打ち切る．このような操作を「枝刈り」といい，分枝操作を行っても暫定解より良い解が得られないと判断できる場合，あるいはそこで 1 つの許容解が得られた場合に分枝操作を打ち切ることで，計算時間を減少させる．

分枝限定法の挙動は分枝木で表わされ，原問題は木の根，子問題は親問題を表す節点に接続される子節点，分枝過程は木の枝を子節点方向にたどること，枝刈りが生じた問題は木の葉となる．

以下に，最適解を 1 つ求める場合の分枝限定法の手順概略を示す．ただし，部分問題内の優越関係による枝刈りの詳細は省略している．また，計算終了後に，$z < \infty$ ならばこのときの暫定解が最適解であり，z がその最適値である．$z = \infty$ ならば許容解は存在しない．

1. 先端ノード集合 $A \leftarrow \{P_0\}$，暫定値 $z \leftarrow \infty$，最適解 $\leftarrow \phi$ と初期化する．
2. $A = \emptyset$(空集合) ならば計算終了．$A \neq \phi$ なら分

枝節点選択ルール *NSR* に従い A の中から部分問題 P_i を選ぶ.

3. P_i の上界値 g_U^i を計算し $g_U^i < z$ ならば，$z \leftarrow g_U^i$ とし，かつ暫定解も更新する．g_U^i の計算により P_i の最適解が得られたならば Step 7 へ行く．
4. P_i の下界値 g_L^i を計算する．$g_L^i \geq z$ ならば，Step 7 へ行く．
5. 優越関係により，P_i からの分枝操作が不要となれば Step 7 へ.
6. 分枝変数選択ルール *VSR* により P_i から変数 x_j を選び，P_i から k 個の子問題 $P_{i_1}, \ldots, P_{i_k}$ を生成し，$A \leftarrow A \cup \{P_{i_1}, \ldots, P_{i_k}\}$ とする．ただし，新たに加える子問題の添字 i_x ($1 \leq x \leq k$) は，$i_x \neq i_y$ ($1 \leq y \leq k$ かつ $x \neq y$) を満し，かつ追加前の A 内に含まれる子問題の添字とは異なるものとする．
7. $A \leftarrow A - \{P_i\}$ と更新して，Step 2 へ戻る． □

3.3 最小重み点被覆問題と分枝限定法

最小重み点被覆問題は次のようにして分枝限定法により解くことができる．

G の点集合を点次数の大きい点から小さい点へ順に $v_1, v_2, \ldots, v_n$ と表し，各 v_i に対し変数 $x_i = 1$ は v_i を点被覆の集合に加え $x_i = 0$ は点被覆の集合に加えないことを表す．最小重み点被覆問題の上界値，下界値は次のように計算する．

- 下界値計算：分枝操作における各部分問題 P_j でのグラフを H_i と表す．P_j の中で点被覆 ($x_j = 1$) に固定，もしくは点被覆の集合に加えない ($x_j = 0$) と固定した点を H_i から除去して定まる連結成分を $G_i = (V_i, E_i)$, $i = 1, 2, ..., k_j$ とする．また，$|V_i| = p_i$，各 G_i での最大次数を $dmax_i$，最小次数を $dmin_i$ とする．さらに $p_i \leq 6$ の場合 p_i，$dmax_i$，$dmin_i$ の全ての組合せに対し各 G_i 中の点被覆数の最小値 n_i を求めてこれらを表にまとめておき，$p_i \leq 6$ の場合はこれを利用して n_i を求める．また，$p_i > 7$ の場合は，

$$n_i = \begin{cases} 1 & dmax_i = p_i - 1 \text{ のとき} \\ |E_i|/dmax_i & dmax_i \neq p_i - 1 \text{ のとき} \end{cases}$$

で定める．これに G_i の点の最小重みを掛けたものを m_i と表す.

下界値は，このようにして求めた m_i の総和と部分問題中の点被覆に固定した点の重みの和である．

- 上界値計算：上界値は，部分問題の未固定変数集合に対し，4 節の貪欲解法または近似解法を用いて得る．

3.4 節点選択ルールと変数選択ルール

先端ノード集合 A の中から分枝節点（部分問題）を選択ルール (*NSR*) によって選択し，未固定変数の中から分枝変数選択ルール (*VSR*) によって，新たに 0 または 1 に値を固定する変数を選択する．これまでに多数のルールが提案されているが特性と性能の基本的な部分はほとんど固まっているようであり，[4] に集大成されている．

分枝限定法においては，これらの選択ルールの組合わせに依存して，同じ問題を解いても得られる分枝木の形が変わってしまう．従って計算回数，及び計算時間にも大きな影響を与えるため，選択ルールの組合わせは分枝限定法において重要な意味を持つと言える．

3.4.1 節点選択ルール (*NSR*)

NSR は，次の (a)，(b) が主流である．

(a) Depth-First Node Selection ルール (DFN)[4]：先端ノード集合 A の中で深さの大きい節点 (つまり，サイズの小さい部分問題) を選択する．生成される部分問題をスタックに維持する．

(b) Best Upper-Bound Node Selection ルール (BUN)[4]：先端ノード集合 A の中で上界値が最大である節点を選択する．生成される部分問題をヒープに維持する．

本研究では [1-3, 6] の結果を受け，節点選択ルールは DFN を用いる．

3.4.2 変数選択ルール (*VSR*)

分枝節点が選択されると，次は未固定変数中でどの変数を固定するかを決定しなければならない．*VSR* については取り扱う問題によって扱いが変わるため，各問題に対して用意する．

最小重み点被覆問題に対しては次の (i)，(ii) を用いる．

(i) 最大次数の点を選択する (但し，次数が同じ場合には重みが最小の点を選択する).
(ii) 重み／次数なる比が最小の点を選択する.

4 貪欲解法および既存近似解法

最小重み点被覆問題に対する貪欲解法 $Greedy$，近似比が 2 である 2 つの近似解法 CLA と BAR，および発見的解法 PIT を概説する．なお，本節で扱う貪欲解法および近似解法の入出力は次の通りである．

入力: 連結単純無向グラフ $G = (V, E)$ と点重み関数 $w : V \rightarrow Z^+$（非負整数）.
出力: 点被覆 $C \subseteq V$.

4.1 貪欲解法 $Greedy$

(1) $\overline{V} \leftarrow V$, $G' \leftarrow G[\overline{V}]$, $C \leftarrow \emptyset$.

(2) $\overline{V} = \emptyset$ まで以下を繰り返す:
$d_{G'}(v)$ が最小の点 $v \in \overline{V}$ を選んで次の操作を行う．但し，次数が等しい点が複数存在する場合は，v_j の中で点重みが最大の点とする．

$\overline{V} \leftarrow \overline{V} - (\{v\} \cup Adj_{G'}(v))$, $C \leftarrow C \cup Adj_{G'}(v)$, $G' \leftarrow G[\overline{V}]$.

4.2 近似解法 CLA [17]

(1) 任意の $v \in V$ に対し，$W(v) \leftarrow w(v)$, $D(v) \leftarrow d_G(v)$.

(2) $C \leftarrow \emptyset, V' \leftarrow V, E' \leftarrow E$.

(3) $E' = \emptyset$ となるまで次の操作を繰り返す：
“$\frac{W(v)}{D(v)}$ が最小となる $v \in V$ を選ぶ．$C \leftarrow C \cup \{v\}$, $W(v) \leftarrow 0$, $V' \leftarrow V' - \{v\}$, $E' \leftarrow E' - \{(v, v') \mid v' \in Adj_{G'}(v)\}$. 但し，$G' = (V, E')$. 各 $u \in Adj_{G'}(v)$ において，$W(u) \leftarrow W(u) - \frac{W(v)}{D(v)}$, $D(u) \leftarrow D(u) - 1$ とする．もし $D(u) = 0$ ならば，$V' \leftarrow V' - \{u\}$ とする．”

4.3 近似解法 BAR [11]

(1) 任意の $v \in V$ に対し，$W(v) \leftarrow w(v)$.

(2) $C \leftarrow \emptyset$, $E' \leftarrow E$.

(3) $E' = \emptyset$ となるまで次の操作を繰り返す：
“任意に $(u_1, u_2) \in E'$ を選ぶ．$W(u_1) \leq W(u_2)$ ならば $j = 1$，そうでなければ $j = 2$ として，$C \leftarrow C \cup \{u_j\}$, $E' \leftarrow E' - \{(u_j, v') \mid v' \in Adj_{G'}(u_j)\}$, $W' \leftarrow W(u_j)$ とする．但し，$G' = (V, E')$ である．更に，$W(u_i) \leftarrow W(u_i) - W'$ $(i = 1, 2)$ とする．”

4.4 発見的解法 PIT [38]

(1) $C \leftarrow \emptyset$. $E' \leftarrow E$.

(2) $E' = \emptyset$ となるまで次の操作を繰り返す：
“任意に $(u, v) \in E'$ を選ぶ．$0 \leq n \leq w(u) + w(v) - 1$ なる乱数 n を発生させる．一般性を失わず，$w(u) \leq w(v)$ とする．$n < w(u)$ ならば $C \leftarrow C \cup \{v\}$ とし，そうでなければ $C \leftarrow C \cup \{u\}$ とする．更に，$E' \leftarrow E' - \{e\}$ とする．”

例題 4.1 (最小重み点被覆問題の実行例) 図 *1* で与えられたグラフに対する最小重み点被覆問題を考える．

図 *3*, *4* はそれぞれ上界値計算に $Greedy$, CLA を採用した場合の実行例である．それぞれ，部分問題を *8* 個または *4* 個生成すると実行が終了しており，上界値計算手法により計算回数が異なることが分かる．

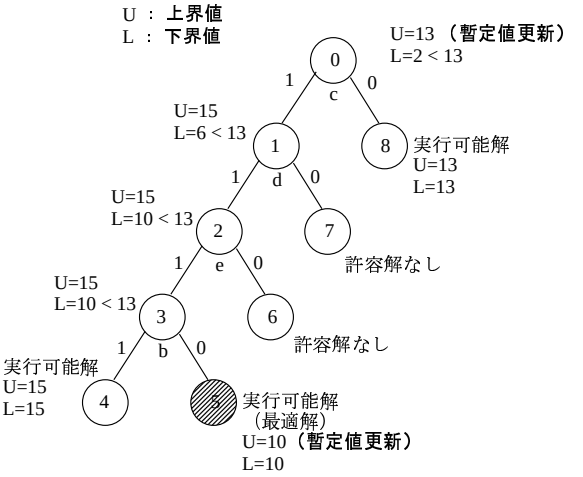

図 3: 上界値計算に $Greedy$ を用いた場合の分枝限定法の実行例．ただし，頂点内の番号は部分問題の処理順序に対応する

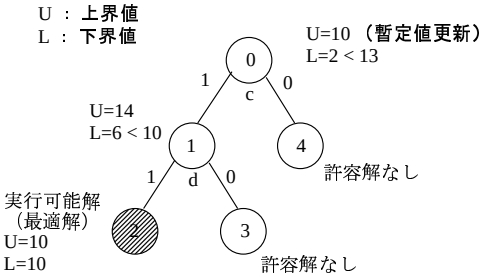

図 4: 上界値計算に CLA を用いた場合の分枝限定法の実行例．ただし，頂点内の番号は部分問題の処理順序に対応する

5 実験概要と結果

5.1 概要

最小重み点被覆問題に対する並列分枝限定法を 100Base-TX のイーサネットで結合した (図 6 参照)12 個のプロセッサ（CPU:Pentium II 450MHz から Pentium III 500MHz; メモリ:800MB から 1GB）からなる計算機ネットワーク上 (各計算機の OS は FreeBSD 4.3-RELEASE) に C 言語を用いて実装した．並列化には MPI[5] を用いた．

5.2 入力データ生成

最小重み点被覆問題の入力グラフ $G = (V, E)$ は，$|V| = 20, 40, 60, 80, 100$ に対して，次の方法によりそれぞれ 10 個 (総数 50 個) のグラフを生成した: 点数が $|V|$ の木をランダムに生成し，$0.2|V| + 1$ 本の辺を多重辺や自己ループを作らないようにランダムに付加する．(つまり，$|E| = 1.2|V| + 1$ である．)

5.3 実験結果

計算機の台数を p とおく．

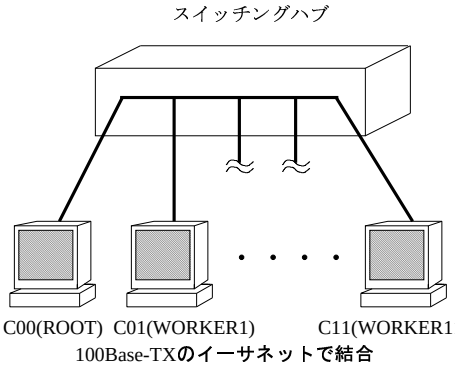

図 5: システム構成の概要

[変数選択ルールの結果] 表 3，4，5，図 6 に示す．表中の “–″ はメモリ不足のため解が出力されず，計測できなかったことを表す．表 3 は $|V| = 100$ の場合における PC クラスタ内の各 PC の計算回数の総和，表 4 は $|V| = 100$ の場合における PC クラスタ内の各 PC の計算回数の最大値，表 5(1)〜(5) はそれぞれ $p = 1, 2, 4, 8, 12$ に対する計算時間を示している．なお上界値計算手法は貪欲解法 $Greedy$ を用いている．

結果は表 5 のように (i) 最大次数の点を選ぶ (但し次数が同じ場合には重みが最小の点を選ぶ) が良い結果を示している．点被覆問題においては，ある点を点被覆集合に加えるとその隣接点は次数を一つ下げることができそれを繰り返すことによりグラフの各連結グラフのサイズが小さくなる．次数が大きい点を選んだ方がより早くグラフの各連結グラフのサイズが小さくなり，早期に実行可能解を得て枝刈りが起るため，(i) が良い結果を示していると考えられる．

[上界値計算手法の結果] 表 6，7，8，9，10，図 7 に示す．表中の “–″ はメモリ不足のため解が出力されず，計測できなかったことを表す．表 6，8 はそれぞれ $|V| = 80$，$|V| = 100$ の場合における PC クラスタ内の各 PC の計算回数の総和，表 7，9 はそれぞれ $|V| = 80$，$|V| = 100$ の場合における PC クラスタ内の各 PC の計算回数の最大値，表 10(1)〜(5) はそれぞれ $p = 1, 2, 4, 8, 12$ に対する計算時間を示している．なお，上記の結果から変数選択ルールには (i) 最大次数の点を選択する (但し，次数が同じ場合には重みが最小の点を選択する) を用いている．

表 8，9 より $|V| = 100$，$p = 12$ の場合においては，上界値計算に既存近似解法 CLA を適用することにより，総計算回数も各 PC の計算回数の最大値も減少させることができた．

しかし，BAR，PIT を適用した場合には，総計算回数も各 PC の計算回数の最大値も増加している．

[8] によると，CLA は BAR，PIT よりも高速に良い解を出力しており，近似解法の性能の優劣が総計算回数の優劣に対応していることがわかる．表 10 より，$p = 12$ の場合において CLA を適用した場合総計算回数が減少したにもかかわらず，計算時間は増大している．このことから，CLA を適用した場合は $Greedy$ に比べ上界値計算 1 回当たりの計算時間が非常に長いことがわかる．近似解法による上界値計算の時間増大が並列分枝限定法の計算時間を増大させる結果となった．

この結果から，分枝限定法においての上界値計算には精度が多少悪くても，高速に値を出力するものを適用する方がより高速に解が得られることがわかった．

6 まとめと今後の課題

本稿では，組合わせ最適化問題である最小重み点被覆問題に対する，並列分枝限定法を PC クラスタネットワーク上で実装し，変数選択ルール，上界値計算手法が計算時間にいかなる影響を与えるかを調べた．今回の実験結果からは，変数選択ルールにおいては「最大次数の点の選択 (但し，次数が同じ場合には重みが最小の点を選択)」が最も計算時間が短くなる，という結果が得られた．

また，上界値計算手法では，精度は悪くてもより高速に暫定解が求められる貪欲解法を適用した方が計算時間が短くなるという結果が得られた．

今後の課題は，次の通りである．

(1) PC クラスタの計算機台数が多い場合，あるいは点数が多い場合の比較実験．

(2) 最小重み点被覆問題の近似解法により得られた近似解の最適解との絶対比較．

参考文献

[1] T. Watanabe, S. Kajita and K. Onaga, “Experimental evaluation of node/variable-selection and load-balancing strategies in parallel branch-and-bound algorithms for solving 0-1 knapsack problems on a transputer network”, Transputer/Occam Japan 4(S. Noguchi and H. Umeo (Eds.)), IOS PRESS, Amsterdam, pp. 102–119, 1992-06.

[2] 梶田, 渡邉, 翁長, “トランスピュータネットワークにおける 0-1 ナップサック問題の分枝限定解法”, 信学技報, NA37-9, pp. 67–74, 1991.

[3] 末広, 渡邉, 翁長, “並列 0-1 ナップサック分枝限定法における節点，変数選択効果”, 信学技報, CPSY89-49, pp. 25–31, 1989.

[4] 茨木俊秀, “組合わせ最適化ー分枝限定法を中心として（講座，数理計画法 8）”, 産業図書, 1980.

[5] P. パチェコ 著, 秋葉 訳, “MPI 並列プログラミング”, 培風館，2001.

[6] T. Kajita, “Evaluations of Parallel Branch-and-Bound Algorithms for Some Combinatorial Optimization Problems on a Transputer Network“, 広島大学大学院工学研究科博士課程前期情報工学専攻修士論文，1992.

[7] 田岡, 渡邉, “PC クラスタ並列分枝限定解法における接点、変数選択規則の実験評価”, IEICE Technical Report COMP2003-7, pp. 47–54, 2003.

[8] 高藤, 田岡, 渡邉, “最小重み点被覆問題に対する近似解法の実験的性能評価”, 第 16 回回路とシステム軽井沢ワークショップ, pp. 507–512, 2003.

[9] M. R. Garey and D. S. Johnson, "Computers and Intractability: a Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1979.

[10] E. M. Arkin, M. M. Halldrsson, and R. Hassin, "Approximating the tree and tour covers of a graph," *IPL*, pp. 275–282, 1993.

[11] R. Bar-Yehud and S. Even, "A linear time approximation algorithm for the weighted vertex cover problem," *J. Algorithms*, Vol. 2, pp. 198–203, 1981.

[12] R. Bar-Yehuda and S. Even, "A local-ratio theorem for approximating the weighted vertex cover problem," *Analysis and Design of Algorithms for Combinatorial Problems*, pp. 27–46, 1985.

[13] N. Bshouty and L. Burroughs, "Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem," in *Proc. 15nth Ann. Symp. on Theoretical Aspects of Comput. Sci.* LNCS, pp. 298–308, Springer-Verlag, 1998.

[14] J. Chen and I. Kanj, "On approximating minimum vertex cover for graphs with perfect matching," in *International Symposium on Algorithms and Computation*, pp. 132–143, 2000.

[15] J. Chen, I. Kanj, and W. Jia, "Vertex cover: Further observations and further improvements," in *Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 313–324, 1999.

[16] J. Chen and I. A. Kanj, "On constrained minimum vertex covers of bipartite graphs: Improved algorithms," in *Graph-Theoretic Concepts in Computer Science WG'01*, A. Brandstädt and V. B. Le, Eds. Vol. 2204 of *LNCS*, pp. 55–65, Springer, 2001.

[17] K. L. Clarkson, "A modification of the greedy algorithm for vertex cover," *IPL*, Vol. 16, pp. 23–25, 1983.

[18] T. Fujito, "A note on approximation of the vertex cover and feedback vertex set problems - unified approach," *IPL*, Vol. 59, No. 2, pp. 59–63, 1996.

[19] T. Fujito, "Approximation algorithms for submodular set cover with applications," *IEICE Trans. INF. & SYST.*, Vol. E83-D, No. 3, pp. 480–487, Mar. 2000.

[20] M. K. Goldberg, T. H. Spencer, and D. A. Berque, "A low-exponential algorithm for counting vertex covers," *Graph Theory, Combinatorics, Algorithms, and Applications*, Vol. 1, pp. 431–444, 1995.

[21] O. Goldreich, "Using the FGLSS-reduction to prove inapproximability results for minimum vertex cover in hypergraphs," ECCC Reports TR01-102, Electronic Colloquium on Computational Complexity (ECCC), 2001.

[22] T. F. Gonzales, "A simple LP-free approximation algorithm for the minimum weight vertex cover problem," *IPL*, Vol. 54, No. 3, pp. 129–131, 1995.

[23] E. Halperin, "Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs," in *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 2000.

[24] J. Håstad, "Some optimal inapproximability results," in *Proc. 29th ACM Symp. on Theory of Computing*, El Paso, pp. 1–10, 1997.

[25] D. S. Hochbaum, "Approximation algorithms for the set covering and vertex cover problems," *SIAM J. Computing*, Vol. 11, pp. 555–556, 1982.

[26] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns, "Nc-approximation schemes for NP- and PSPACE-hard problems for geometric graphs," *J. Algorithms*, pp. 238–274, 1998.

表 3: 変数選択ルール ($|V| = 100$) 別の総計算回数

台数	1	2	4	8	12
(i)	353134	359328	365848	434580	413128
(ii)	11970695	12609898	10913486	9982364	7135929

表 4: 変数選択ルール ($|V| = 100$) 別の PC クラスタ内の各 PC の計算回数の最大値．括弧内は $p = 1$ の場合との比

台数	1	4	12
(i)	353134	146048(2.4179)	62114(5.6852)
(ii)	11970695	3700799(3.2346)	968768(12.3566)

[27] M. Karpinski and A. Zelikovsky, "Approximating dense cases of covering problems," Tech. Rep. TR97-004, ECCC, 1997.

[28] M. Karpinski and A. Zelikovsky, "Approximating dense cases of covering problems," *Electronic Colloquium on Computational Complexity (ECCC)*, Vol. 4, No. 004, 1997.

[29] S. Khuller, U. Vishkin, and N. E. Young, "A primal-dual parallel approximation technique applied to weighted set and vertex covers," *J. Algorithms*, Vol. 17, No. 2, pp. 280–289, 1994.

[30] S. Khuri and T. Bäck, "An evolutionary heuristic for the minimum vertex cover problem," in *Genetic Algorithms within the Framework of Evolutionary Computation – Proc. of the KI-94 Workshop*, J. Hopf, Ed., Saarbrücken, Germany, pp. 86–90, 1994.

[31] J. Kleinberg and M. X. Goemans, "The lováz theta function and a semidefinite programming relaxation of vertex cover," *SIAM Journal on Discrete Mathematics*, Vol. 11, No. 2, pp. 196–204, 1998.

[32] B. Monien and E. Speckenmeyer, "Ramsey numbers and an approximation algorithm for the vertex cover problem," *Acta Inf.*, pp. 115–123, 1985.

[33] R. Motwani, "Lecture notes on approximation algorithms: Volume I," Tech. Rep. CS-TR-92-1435, Stanford University, Department of Computer Science, 1992.

[34] H. Nagamochi and T. Ibaraki, "An approximation of the minimum vertex cover in a graph," *Japan J. Indust. Appl. Math.*, pp. 369–375, 1999.

[35] G. L. Nemhauser and L. E. Trotter Jr., "Vertex packing: Structural properties and algorithms," *Mathematical Programming*, Vol. 16, pp. 232–248, 1975.

[36] R. Niedermeier and P. Rossmanith, "Upper bounds for vertex cover further improved," *LNCS*, pp. 561–570, 1999.

[37] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Comput. System Sci.*, Vol. 43, pp. 425–440, 1991.

[38] L. Pitt, "A simple probabilistic approximation algorithm for vertex cover," Tech. Rep. YaleU/DCS/TR-404, Department of Computer Science, Yale University, 1985.

[39] Y. Saab, "Iterative improvement of vertex covers," *IPL*, Vol. 55, No. 2, pp. 95–98, 1995.

[40] C. Savage, "Depth-first search and the vertex cover problem," *IPL*, Vol. 14, pp. 233–237, 1982.

[41] A. Srinivasan, "Improved approximation guarantees for packing and covering integer programs," *SIAM Journal on Computing*, Vol. 29, No. 2, pp. 648–670, 1999.

表 5: 変数選択ルール別の計算時間 (単位：秒)

(1) $p = 1$

n	20	40	60	80	100
(i)	0.007	0.017	0.275	2.406	23.422
(ii)	0.007	0.043	9.039	117.217	803.466

(2) $p = 2$

n	20	40	60	80	100
(i)	0.008	0.032	0.353	2.376	23.215
(ii)	0.010	0.058	10.042	102.564	691.703

(3) $p = 4$

n	20	40	60	80	100
(i)	0.017	0.023	0.323	1.962	17.708
(ii)	0.010	0.049	6.567	61.400	446.277

(4) $p = 8$

n	20	40	60	80	100
(i)	0.021	0.042	0.182	0.862	13.522
(ii)	0.028	0.069	3.266	39.062	253.731

(5) $p = 12$

n	20	40	60	80	100
(i)	0.061	0.079	0.216	0.656	10.231
(ii)	0.082	0.105	1.953	31.506	167.862

表 6: 上界値計算手法 ($|V| = 80$) 別の総計算回数

台数	1	2	4	8	12
Greedy	43580	39423	40116	31603	28699
CLA	40103	40984	40550	32013	30718
BAR	55637	51093	51362	41727	36828
PIT	51925	51764	47016	35087	31087

表 7: 上界値計算手法 ($|V| = 80$) 別の PC クラスタ内の各 PC の計算回数の最大値．括弧内は $p = 1$ の場合との比

台数	1	4	12
Greedy	43580	17869(2.4388)	4245(10.2661)
CLA	40103	19531(2.0532)	4448(9.0159)
BAR	55637	23222(2.3958)	5220(10.6584)
PIT	51925	20934(2.4804)	4619(11.2416)

表 8: 上界値計算手法 ($|V| = 100$) 別の総計算回数

台数	1	2	4	8	12
Greedy	353134	359328	365848	434580	413128
CLA	-	-	-	-	392433
BAR	719509	673788	679658	782872	658357
PIT	605062	441850	568256	633961	532472

表 9: 上界値計算手法 ($|V| = 100$) 別の PC クラスタ内の各 PC の計算回数の最大値．括弧内は $p = 1$ の場合との比

台数	1	4	12
Greedy	353134	146048(2.4179)	62114(5.6852)
CLA	-	-	59258(-)
BAR	719509	250908(2.8676)	93041(7.7332)
PIT	605062	204590(2.9574)	77995(7.7577)

表 10: 上界値計算手法別の計算時間 (単位：秒)

(1) $p = 1$

n	20	40	60	80	100
Greedy	0.007	0.017	0.275	2.406	23.422
CLA	0.007	0.103	1.692	12.040	-
BAR	0.007	0.042	0.607	5.477	81.378
PIT	0.007	0.027	0.427	3.698	52.647

(2) $p = 2$

n	20	40	60	80	100
Greedy	0.008	0.032	0.353	2.376	23.215
CLA	0.009	0.100	1.578	11.360	-
BAR	0.014	0.050	0.635	4.764	66.568
PIT	0.011	0.038	0.474	3.421	42.084

(3) $p = 4$

n	20	40	60	80	100
Greedy	0.017	0.023	0.323	1.962	17.708
CLA	0.018	0.081	1.148	7.573	-
BAR	0.008	0.050	0.500	3.423	42.368
PIT	0.010	0.028	0.382	2.510	28.869

(4) $p = 8$

n	20	40	60	80	100
Greedy	0.021	0.042	0.182	0.862	13.522
CLA	0.060	0.079	0.578	2.939	-
BAR	0.034	0.049	0.262	1.305	28.184
PIT	0.042	0.049	0.191	0.973	19.867

(5) $p = 12$

n	20	40	60	80	100
Greedy	0.061	0.079	0.216	0.656	10.231
CLA	0.092	0.112	0.437	2.120	33.067
BAR	0.084	0.110	0.231	0.987	19.205
PIT	0.089	0.103	0.179	0.764	13.635

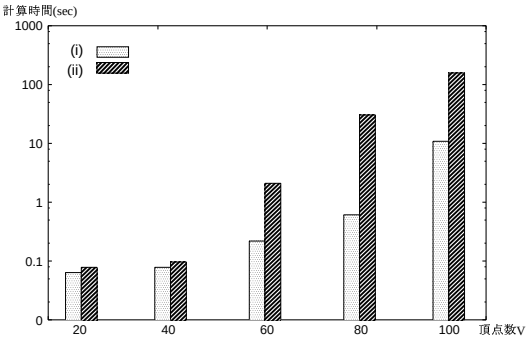

図 6: 変数選択ルール別の計算時間 ($p = 12$ 固定，単位：秒)

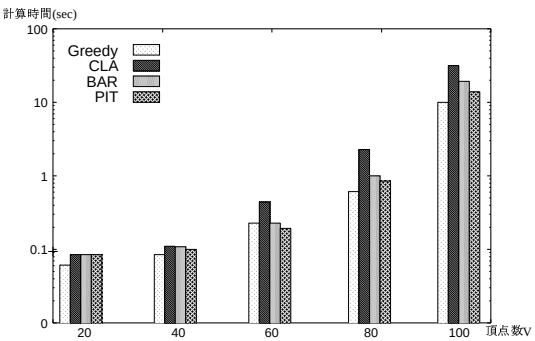

図 7: 上界値計算手法別の計算時間 ($p = 12$ 固定，単位：秒)