最大利得部分木問題に対する近似解法および厳密解法

星崎康広[†] 片岡靖詞[††] 森戸晋[†††]

連結無向グラフにおいて，1 点が根として指定されているとする．また，各枝にはコスト，各点には利得が与えられ，さらに総コスト上限が定められているとする．このとき，最大利得部分木問題とは，根を含む連結部分木で，枝のコストの和が総コスト上限以下で，点の利得の和が最大となるものを求める問題である．本論文の目的は，この問題に対して近似解法および分枝限定法による厳密解法を構築し，計算実験によりそれら有効性を検証することである．特に分枝限定法による厳密解法の構築にあたっては，数理計画による定式化を行って線形緩和を利用したアプローチと，元の問題を部分木の点や枝の数を制限した問題に分割するアプローチの 2 つを試みた．計算機実験の結果，元の問題を分割するアプローチの方が有効であることが分かった．

Approximate and Exact Algorithms for Maximum Profitable Subtree Problem

Yasuhiro Hoshizaki,[†] Seiji Kataoka[††] and Susumu Morito[†††]

Suppose we are given a graph that is undirected and connected. In this graph, a vertex is specified as a root. A cost and a profit are also given to each corresponding edge and vertex, respectively. Then, maximum profitable subtree problem is a problem of finding a subtree that maximizes total profits within a prescribed amount of total cost. Of course, the resultant subtree must be connected and include the root. We propose some approximate and branch-and-bound algorithms to the problem and investigate the efficiency of these procedures with computational experiments. Especially for branch-and-bound exact algorithm, two approaches are proposed: The first approach uses a mathematical programming formulation; the second approach divides the problem into some subproblems that restrict the number of vertices or edges in the subtrees. Our results imply the effectiveness of the latter approach.

1. はじめに

点集合 V，枝集合 E を持つ連結無向グラフ $G = (V, E)$ において，根 r $(\in V)$ が付与されているとする．また，各点には正の値を持つ利得 p_i $(i \in V)$，各枝には非負のコスト c_{ij} $((i, j) \in E)$，さらに総コスト上限 C が与えられているとする．このとき，最大利得部分木問題（Maximum Profitable Subtree Problem: MPSP）とは，根 r を含む連結部分木の中で，枝のコストの総和が C 以下であり，カバーする点の利得の総和が最大になるようなものを求める問題である．

† 富士写真フィルム株式会社生産技術部
Production Engineering & Development Division, Fuji Photo Film Co., Ltd.
†† 防衛大学校情報工学科
Department of Computer Science, National Defense Academy
††† 早稲田大学理工学部経営システム工学科
Department of Industrial & Management Systems Engineering, Waseda University

MPSP は，予算の上限が決められている状況で，できるだけ多くの端末をつなぐ通信ネットワークや，人口の多い都市を結ぶ高速道路網などの構築を行う場合などに応用できる．また，スコア・オリエンテーリング問題など，より複雑な巡回路問題に対する下界値評価などにも利用することができる．**図 1** に最適解の例を示す．

MPSP は，基本的な問題で応用も豊富と考えられるにもかかわらず，同問題に対しての過去の研究は見当たらない．類似した問題として，スコア・オリエンテーリング問題[4] があげられる．スコア・オリエンテーリング問題は，部分木の代わりに，部分巡回路を求める問題である．

仮にすべての点を解に含むことが可能（であるがそれが未知）であるような総コスト上限 C が与えられたとき，スコア・オリエンテーリング問題では巡回セールスマン問題の決定問題と等価になってしまうために $\mathcal{NP}$-完全であるのに対し，MPSP は決定問題であれば最小木問題を解くことにより簡単に解決できる．し

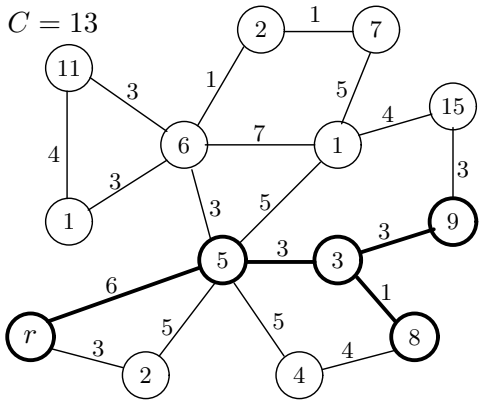

図 1 最大利得部分木問題の最適解
Fig. 1 An example of the optimal solution to MPSP.

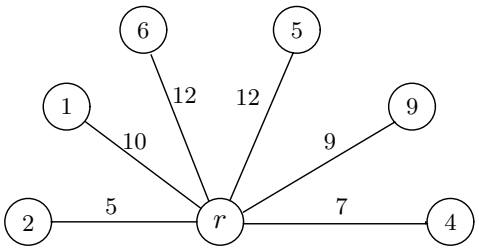

図 2 ナップサック問題への帰着
Fig. 2 Transformation to knapsack problem.

かしながら，C がすべての点を解に含めることができないような値のときは，MPSP の最適化問題は，**図 2** ようにグラフのすべての枝が根に接続している場合を考えれば，ナップサック問題に帰着できるため，同問題は $\mathcal{NP}$-困難であることが証明される．この場合，総コスト上限がナップサックの容量に相当し，各枝のコストがナップサックに入れる品物の容積，各点の利得がナップサックに入れる品物の利得に相当する．

MPSP に対する解法を構築することは，実問題への適用や問題の性質を知るうえで重要である．そこで本論文では，ヒューリスティック解法を 2 種類（2 章）と，分枝限定法を用いた厳密解法も 2 種類提案する．それぞれの分枝限定法では，問題の数理計画による定式化を行い，線形緩和を利用した分枝限定法（3 章）と，問題の性質を利用した分枝限定法（4 章）を構築する．5 章の計算機実験においてヒューリスティック解法および 2 種類の分枝限定法の有効性の比較を行う．

2. ヒューリスティックアルゴリズム

ヒューリスティックアルゴリズムは，一般に構築型と逐次改善型の 2 種類に大別できるが，本論文においても以下の 2 種類のヒューリスティックアルゴリズムを提案する．

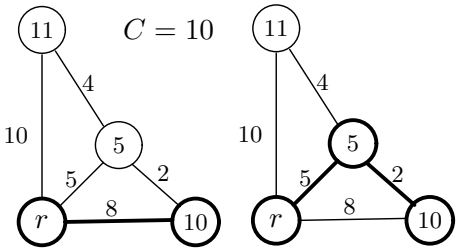

図 3 点を加えることによりコストの総和が少なくなる場合
Fig. 3 The reduction of total cost by including a vertex.

2.1 ヒューリスティック 1（構築型）

構築型のヒューリスティックでは，根のみの自明な木を初期解として，総コスト上限制約を破らないように点を 1 つずつ取り込んで木を拡張していく戦略をとる．このとき，MPSP では，解に含まれる点集合が決められた場合，それらの点から誘導されるグラフが連結であれば，枝集合はそのグラフでの最小木を構成するときが最も効率が良い．このことに注意して，ここでのヒューリスティックアルゴリズムでは，各繰返しにおいて連結性が保たれるように取り込む候補となる点を定め，それらに対し最小木を求め，その値をもとに 1 つの取り込むべき点を決定している．手順 Heuristic1() で，MST() とあるのは，点集合を入力としたときの最小木の解（枝集合）を返す関数である．もし，次のヒューリスティック 2 で起こりうるように，最小木が構成できない場合，MST() はすべての枝集合 E を返すことにする．また，現在の木を構成している点および枝集合を V_T，E_T とするとき，$p(V_T) = \sum_{i \in V_T} p_i$，$c(E_T) = \sum_{e \in E_T} c_e$ である．

ここで，現在の木に新たに点を加えるとき，$p_i > 0$ $(i \in V)$ より，利得の変化量は必ず増加するが，コストの変化量は**図 3** のように点を追加することによって減少する場合もあることに注意する．したがって，利得の増加量が同じであれば，Heuristic1() に示すようにコスト和が小さくなる方の点を優先的に取り込む方がよい．

$V_T := \{r\}$; $E_T := \emptyset$; $C_T := 0$; /*初期解*/
Heuristic1(V_T, E_T, C_T)
while (1) {
  max $:= -\infty$;
  $V_{cut} := \{j \mid i \in V_T, j \in V \setminus V_T, (i,j) \in E\}$;
  for $(i \in V_{cut})$ {
    $T_i :=$ MST$(V_T \cup \{i\})$;
    if $(c(T_i) \leq C$ and max $< (p(V_T) + p_i)/c(T_i))$ {

```
      v := i; /*最良解の保存*/
      T := T_i;
      max := (p(V_T) + p_i)/c(T_i);
    }
  }
  if (max > −∞) { /*追加可能な点が存在する*/
    V_T := V_T ∪ {v}; /*解の更新*/
    E_T := T;
    C_T := c(T);
  }
  else /*追加可能な点がない*/
    return V_T, E_T, C_T;
}
```

構築型の Heuristic1() だけで求めた解では，図 3 のような例題に対し，右側の図のような利得 15 の解で停止してしまう．しかしながら，この例題には利得 16 という最適解が存在するのは明らかであろう．このような場合を避けるために，Heuristic1() で求めた解をもとに逐次改善を行うヒューリスティック 2 を提案する．

2.2 ヒューリスティック 2（逐次改善型）

ヒューリスティック 2 の手順を Heuristic2() に示す．基本的には点を交換するアルゴリズムであり，その操作を ExchangeNode() の中で行っている．ExchangeNode() で得られる最良解というのは，入力である現在解の近傍に，現在解よりも良い解がない場合には，改悪された解になってしまう場合もある．そのため Heuristic2() では，解が改善されたときには，V_{best}，E_{best}，C_{best} に解を保存するが，改悪された場合にも現在解を更新し，ベストの解を更新しない繰返しが連続して L_{lim} 回に達した場合に終了することにしている．本研究では，予備実験の結果 $L_{lim} = 50$ に設定している．

Heuristic2() では，ベストの解を更新できる/できないにかかわらず，V_T に何らかの更新があった場合には，再び Heuristic1() に入り，さらに追加できる点があるかどうかを調べている．また，ExchangeNode() あるいは Heuristic2() の前後で解に変化があるか否かで，若干無駄を省き，効率をあげることもできる．

```
Heuristic1() で得た V_t, E_t, C_t /*初期解*/
Heuristic2(V_T, E_T, C_T)
V_best := V_T; E_best := E_T; C_best := C_T; L := 0;
while (L < L_lim){
  ExchangeNode(V_T, E_T, C_T);
  if (p(V_best) < p(V_T)){ /*ベストの更新*/
    V_best := V_T;
    E_best := E_T:
    C_best := C_T;
    L := 0 /*L の初期化*/
  }
  else /*ベストの更新なし*/
    L := L + 1;
  Heuristic1(V_T, E_T, C_T); /*点を追加*/
}
return V_best, E_best, C_best;

ExchangeNode(V_T, E_T, C_T)
max := −∞;
V_cut := {j | i ∈ V_T \ {r}, j ∈ V \ V_T, (i, j) ∈ E};
for (i ∈ V_T, j ∈ V_cut){ /*点 i, j の交換*/
  V_ij := (V_T \ {i}) ∪ {j};
  T_ij := MST(V_ij);
  if (c(T_ij) ≤ C and max < p(V_ij)/c(T_ij)){
    V := V_ij; /*最良解の保存*/
    T := T_ij;
    max := p(V_ij)/c(T_ij);
  }
}
V_T := V;
E_T := T;
C_T := c(T);
return V_T, E_T, C_T;
```

以上の 2 種類のヒューリスティック解法に関して，計算機実験結果は 5 章にまとめて報告する．また，分枝限定法による厳密解法の中での暫定解を利用する部分においても，これらのヒューリスティック解法を利用している．

3. 問題の定式化と分枝限定法

一般に，無向グラフ上で定義される問題は，有向グラフ上の問題としても等価に変換できる．ここでは非対称型巡回セールスマン問題で研究されている分枝ルールなどを効果的に用いるためにも，いったん問題を有向グラフ $G^D = (V, E^D)$ 上で再定義する．ここに，点集合 V および各点の利得は無向グラフの場合と同じとし，有向枝集合 E^D は無向枝集合 E の各枝に対し，両方向に向きをつけたものとする．さらに有向枝のコストを $c_{ij} = c_{ji} = c_{(i,j)}$ $((i, j) \in E,\ i, j \in V)$ とし，問題は根 r からの有向出力部分木を求める問題

とする．この章では，まず MPSP を有向グラフ上での数理計画問題として定式化を行い，分枝限定法による厳密解法について説明するが，煩雑をさけるため，特に混乱のない限り有向枝も枝，有向出力部分木も有向木と呼ぶことにする．

3.1 定 式 化

根を点 1 とし，以下のように決定変数を定める．

$$x_{ij} = \begin{cases} 1: & \text{枝 } (i,j) \text{ が部分木に含まれる}, \\ 0: & \text{枝 } (i,j) \text{ が部分木に含まれない}. \end{cases}$$

$$y_i = \begin{cases} 1: & \text{点 } i \text{ が部分木に含まれる}, \\ 0: & \text{点 } i \text{ が部分木に含まれない}. \end{cases}$$

このとき，MPSP は次のように 0-1 整数計画問題として定式化できる．

$$\max \sum_{i \in V} p_i y_i, \tag{1}$$

$$\text{s.t.} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \le C, \tag{2}$$

$$\sum_{i \in V} x_{ij} = y_j, \qquad \forall j \in V \setminus \{1\}, \tag{3}$$

$$2x_{ij} \le y_i + y_j, \qquad \forall (i,j) \in E^D, \tag{4}$$

$$y_1 = 1, \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \quad \forall S \subseteq V, S \neq \emptyset, \tag{6}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E^D, \tag{7}$$

$$y_i \in \{0,1\}, \qquad \forall i \in V. \tag{8}$$

3.2 線形緩和による分枝限定法

3.1 節で示した定式化において，式 (6) で示される部分巡回路除去制約は，その数が膨大なものになるため，汎用ソフトウェアをそのまま利用することは困難である．そこで，本研究では式 (6) の代わりとして，任意の 2 点間の部分巡回路のみを除去する制約

$$x_{ij} + x_{ji} \le 1, \quad \forall (i,j),(j,i) \in E^D \tag{9}$$

に置き換え，さらに 0-1 整数制約 (7)，(8) を連続緩和した問題を緩和問題とした．この線形計画問題は，汎用ソフトウェア XPRESS-MP を用いて解き，各子問題における上界値とした．また，分枝限定法の本体部分および XMPRESS-MP とのインタフェース部分は C 言語でプログラムを作成した．この線形緩和問題を解いて実行可能解が得られればそれが最適解であるが，一般に分数解が得られる場合と，3 つ以上の点で式 (6) を満足しない解が得られる場合とがある．それぞれの場合において，分枝変数および分枝ルールを次のように定める．

分枝変数の選択としては，どちらの場合でも，点に対応する変数に着目することとする．これは，本来この問題は最適な枝集合を求める問題ではあるが，一般に点の数の方が枝の数よりも少なく，さらに，部分木に含まれる点集合が一意に決まった場合，それらを張る枝も最小木として一意に定まることから，点に対応する変数を分枝変数とした方が効果的であると予想されるからである．また，ある点を含まないように固定された問題では，その点に接続する枝はすべて除去でき問題を縮小することもできる．このことは，4 章の問題の性質を利用した分枝限定法でも同様とする．

分枝ルールは，それぞれの場合において次のように定める．仮に 2 つの場合が同時に起こった場合は，分数解が得られる場合を優先させる．

分数解が得られる場合 分数解となった変数に注目し，それを 0 または 1 に固定する 2 分木戦略をとる．分枝変数の候補が複数ある場合には，最も利得の大きな点を優先的に選ぶ．

3 つ以上の点で式 (6) を満足しない解が得られる場合 非対称型巡回セールスマン問題の標準的な分枝ルール[2] に準じ，それを点に対応する変数に変換したものを用いる．つまり，式 (6) を満たさない点集合を $\{v_1, v_2, \ldots, v_m\}$ とし，分枝頂点 ℓ において，I_ℓ を解に必ず含める点集合，O_ℓ を解に絶対含めない点集合とすると，分枝頂点 ℓ の t $(t = 1, 2, \ldots, m)$ 番目の子問題を次のように定める．

$$I_t = I_\ell \cup \{v_1, \ldots, v_{t-1}\}$$
$$O_t = O_\ell \cup \{v_t\}$$

候補が複数ある場合には，最も m の値の小さなものに着目する．

未分枝頂点の選択ルールは，奥行き優先とする．また，分枝限定法に入る前に，下界値となる暫定解を求める方法としては，2 章で説明したヒューリスティックアルゴリズム（Heuristic1()，Heuristic2()）を用いている．

4. 問題の性質を利用した分枝限定法

ここでは，MPSP の最適解が k 個の点を含んでいるものと仮定して，k の値を固定した問題 k-MPSP を解く分枝限定法を考える．一般に k の値を前もって知ることはできないが，次に示す k-MPSP の上下界値などの求め方を利用すれば，k の上下限の値も同時に知ることができる．その範囲内の各々の k の値において k-MPSP を最適に解き，それらの最適値の中で最大のものをとればよい．

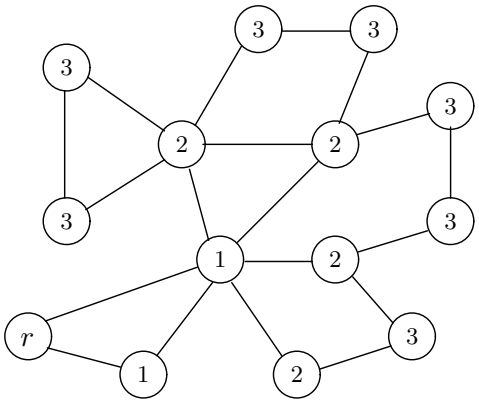

図 4 点の距離
Fig. 4 The distance of each vertex.

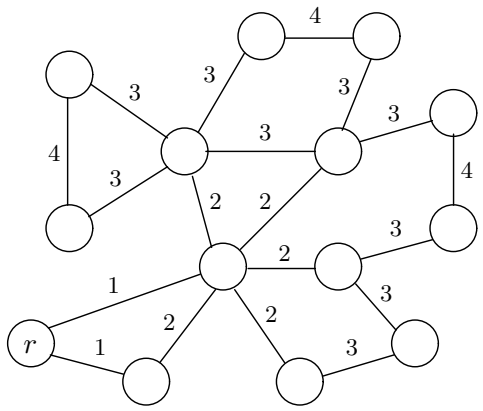

図 5 枝の距離
Fig. 5 The distance of each edge.

4.1 k-MPSP の上界値

ここでは，k-MPSP の上界値を与える貪欲的アルゴリズム（Greedy algorithm to obtain Upper Bound: GUB）を提案する．GUB は根から各点への距離という概念を利用したアルゴリズムである．

点の距離 点の距離とは各枝の長さを 1 と見なして，根から各点への最短距離のことをいう．**図 4** に点の距離の例を示す．

各点に与えられた距離を利用し，点集合 V_1 を距離 1 の点，V_2 を距離 2 の点，$\cdots$，V_i を距離 i の点となるよう V を分割する．i が点の距離の最大値よりも大きいときは，$V_i = \emptyset$ とする．そして，1 つめの点は V_1 から，2 つめの点は $V_1 \cup V_2$ から，k 個めの点は $V_1 \cup V_2 \cup \ldots \cup V_k$ からそれぞれ利得が最大のものを選ぶ．

以下に GUB() のアルゴリズムを示す．この中で $d(v)$ は点 v の距離を示している．

GUB(k)
$V_{UB} := \emptyset$; $i := 1$;
while ($i \leq k$) {
  $v_i := \arg\max\{p(v) \mid v \in \bigcup_{j=1}^{i} V_j\}$;
  $V_{d(v_i)} := V_{d(v_i)} \setminus \{v_i\}$;
  $V_{UB} := V_{UB} \cup \{v_i\}$;
  $i := i + 1$;
}
return $p(V_{UB})$;

GUB() では，選ぶべき点集合に制約がついているので，単純に利得の最大のものから k 個の点を選ぶよりも効果的な上界値を得ることができる．

定理 1 GUB() は，k-MPSP の上界値を与える．

証明 点の距離を用いて定義される点集合 V_i と，最適な k-MPSP の解を構成する部分木において同様に点の距離を用いて定義される点集合 V_i' を考えると，明らかに $(\bigcup_{j=1}^{i} V_j') \subseteq (\bigcup_{j=1}^{i} V_j)$ $(i = 1, \ldots, k)$ である．つまり，最適な k-MPSP で得られるような点集合は，GUB() の各繰返しで得られる点集合に含まれる．したがって，緩和の原理より，GUB() は k-MPSP の上界を与える． □

また，GUB() を用いて k の値の下限を求めることができる．GUB() の返す値が暫定解の値未満になれば，そのときの k からは MPSP の最適解が得られることはない．また，GUB() の返す値は k によって単調増加となるので，最初に暫定値以上になる k が下限値となる．

4.2 k-MPSP のコスト総和の下界値

次に枝を $k-1$ 本使った場合に k-MPSP のコスト総和の下界値を与える貪欲的アルゴリズム（Greedy algorithm to obtain Lower Bound: GLB）を考える．GUB() と同様に今度は根から各枝への距離という概念を利用する．

枝の距離 枝の距離とは，根からグラフの枝の幅優先探索を行った際に生成される探索木のレベルのことである．**図 5** に枝の距離の例を示す．

各枝に与えられた距離を利用して，枝集合 E_1 を距離 1 の枝，E_2 を距離 2 の枝，$\cdots$，E_i を距離 i の枝となるように枝集合 E を分割する．この場合も，i が枝の距離の最大値よりも大きいときは，$E_i = \emptyset$ とする．そして，1 本目の枝は E_1 から，2 本目の枝は $E_1 \cup E_2$ から，$k-1$ 本目の枝は $E_1 \cup E_2 \cup \cdots \cup E_{k-1}$ からそれぞれコストが最小のものを閉路が生成しないように選ぶ．もし，途中で閉路ができる場合には，その枝を考慮の対象から外した後，次にコストの小さいものを

選ぶ．

GLB() のアルゴリズムを示す．この中で $\delta(e)$ とあるのは，枝 e の距離を示している．

GLB(k)
$E_{LB} := \emptyset$; $i := 1$;
while ($i < k$) {
  $e_i := \arg\min\{c(e) \mid e \in \bigcup_{j=1}^{i} E_j\}$;
  $E_{\delta(e_i)} := E_{\delta(e_i)} \setminus \{e_i\}$;
  if ($E_{LB} \cup \{e_i\}$ has no cycle) {
    $E_{LB} := E_{LB} \cup \{e_i\}$;
    $i := i + 1$;
  }
}
return $c(E_{LB})$;

この場合も選ぶべき枝集合に制約がついているので，単純にコストの小さい枝から $k-1$ 本の枝を選ぶよりも効果的な k-MPSP の下界値を得ることができる．また，最小木を解く Kruskal のアルゴリズム[1] を $k-1$ 回目で止めても同様に下界値となるが，GLB() はさらに良い下界値を得ることができる．

定理 2 GLB() は，k-MPSP の枝コストの総和の下界値を与える．

証明 これは，アルゴリズムの中に閉路を生成するかどうかのチェックが入っているので，証明は定理 1 ほど明かではないが，Kataoka ら[3] はマトロイド理論を利用し，GLB() が k-MPSP の枝コスト総和の下界値を与えることを示している． □

GUB() と同様，GLB() を用いて k の値の上限を求めることができる．GLB() の返す値が総コスト上限制約 C を超えた場合，MPSP の最適解では，そのときの k の値をとることはない．また，GLB() の返す値は k によって単調非減少となるので k の上限値を求めることができる．

4.3 k-MPSP の分枝限定法

GUB() と GLB() を用いて k の値の上下限が定められるので，その間の各 k に対して，k-MPSP を分枝限定法により厳密に解くことを考える．

k-MPSP における分枝限定法では，緩和問題として GUB() やコスト総和の下界を求める GLB() を各分枝頂点において適用する．そのとき，選択すべき点で誘導されるグラフが連結していない場合は，$\mathcal{NP}$-困難なスタイナー木問題になってしまうことに注意する．このため，選択しなければならない点は連結性を保っていなければならない．このことは，それらの点集合を 1 つの根として縮約できることを意味している．

また，連結性を保ちながら，新たに解に含める点の候補が複数ある場合には，最もコストの小さい枝に接続する点を優先的に選択する．このような点の選択ルールにより，縮約された根に含まれる点集合では，つねに最小木を保つことができる．

定理 3 縮約された根の中における最小木は，この分枝頂点の問題における k-MPSP の最適解の部分グラフになっている．

証明 点 R をある分枝頂点における縮約された根，枝 (R, s) が最もコストの小さい枝としたとき，点 s が選択しなければならない点となる．この問題の最適解を考えたとき，点 s が枝 (R, s) 以外の枝によって点 R と連結している場合，点 s と点 R の間に枝 (R, s) 以外の路が存在することになる．この路に含まれる枝のうち，R に接続する枝を (R, a) とすると，分枝規則により枝 (R, a) のコストは，枝 (R, s) のコスト以上になる．よって枝 (R, a) を枝 (R, s) と交換した方がよくなり，定理が証明された． □

定理 3 より，いくつかの点が固定された各分枝頂点の問題においても GUB()，GLB() が使え，それぞれが上界値，コストの総和の下界値を返すことができる．したがって，分枝限定法の枠組みとして，ある分枝頂点で GUB() で返してくる値が暫定解の値以下であれば，その分枝頂点から先の分枝を停止することができる．

さらに，GUB()，GLB() の利用として，ある分枝頂点において GUB() で算出された解の点集合で最小木を解いた場合，枝コストの総和が総コスト上限 C 以下であれば，実行可能解が出たものとしてその分枝頂点から先の分枝を停止することができる．それが暫定解より良いものであれば，暫定解の更新もできる．GLB() の活用としては，縮約された根の中の枝と合わせて $k-1$ 本の枝を選んだとき，それらのコストの総和が，総コスト上限を超えれば，実行不可能性が保証され，その場合もその分枝頂点から先の分枝を停止することができる．

分枝方法は選択した分枝変数に対応する点による 2 分木探索を行い，奥行優先とする．ここでも，暫定解としては，Heuristic1()，Heuristic2() で求めた解を用いる．

5. 計算機実験および結果

計算機実験では，2 章で議論した構築型と逐次改善型の 2 種類のヒューリスティック解法の比較，および 3 章，4 章で議論した線形緩和を利用した分枝限定法

と問題の性質を利用した分割型の分枝限定法との比較を中心に行う．いずれの場合も，C 言語を用いてプログラムを記述し，Sun Spark Station 20 上で実行させる．

実験に用いるグラフは，点を 1 辺が $2|V|$ の正方形領域内の整数格子点に一様乱数で与え，点 v の利得 $p(v)$ は 1 から 20 の整数値を一様乱数で与え，枝 e のコストは，2 点間のユークリッド距離の整数部分 $+1$ としている．また，グラフの連結性を保証するために，最小全域木は必ず含めるようにし，残りの枝はコストの小さい方から順に選んでいる．枝の数 $|E|$ および総コスト上限 C は，結果の表に示しているように与える．

5.1 ヒューリスティック解法の性能

ヒューリスティック解法では，主に解の精度と計算時間に着目し，2 章で説明した 2 種類の手法について比較を行う．**表 1** において，Heuristic1 とあるのは構築型を示し，Heurisitc2 とあるのは逐次改善型を示している．ここで取り上げた例題については，事前に最適値を計算しており，preci. とある列には，最適値に対する各ヒューリスティック解法で得た目的関数値の割合を示している．また，CPU は実行時間（秒）である．表中の数値は，各パラメータにおいて 10 回試行し，その平均値を示している．ただし，$|V| \geq 40$ においては，最適値を求めるのに多大な時間を要するため，各 1 回のみの結果である．

表 1 ヒューリスティックアルゴリズムの比較
Table 1 The results of heuristic algorithms.

$\|V\|$	$\|E\|$	C	Heuristic1		Heuristic2	
			preci.	CPU	preci.	CPU
10	20	20	1.00	0.00	1.00	0.00
10	20	30	1.00	0.00	1.00	0.03
10	30	20	1.00	0.00	1.00	0.00
10	30	30	0.87	0.00	1.00	0.01
20	50	50	0.96	0.01	1.00	0.18
20	50	100	1.00	0.01	1.00	0.03
20	100	50	0.95	0.01	0.95	0.01
20	100	100	1.00	0.01	1.00	0.86
30	100	50	0.95	0.00	1.00	0.30
30	100	100	0.92	0.00	0.92	3.18
30	200	50	1.00	0.00	1.00	0.40
30	200	100	0.80	0.01	0.84	1.80
40	100	100	0.98	0.00	0.99	3.10
40	100	200	0.94	0.06	0.94	15.00
40	100	300	0.94	0.09	1.00	25.00
40	200	100	0.95	0.01	0.95	0.16
40	200	200	0.95	0.14	0.95	22.10
40	200	300	0.95	0.21	0.99	25.80
50	150	100	0.92	0.00	0.97	1.41
50	150	200	0.91	0.03	0.91	12.30
50	150	300	0.92	0.18	0.98	59.80
50	150	400	0.90	0.41	0.99	76.50
50	300	100	0.93	0.01	0.93	3.44
50	300	200	0.86	0.09	0.93	19.70
50	300	300	0.94	0.38	1.00	74.80
50	300	400	0.98	0.66	0.98	83.60

Heuristic1 と Heuristic2 との比較において，Heuristic2 は Heuristic1 の解をもとに改善したものなので，目的関数値は必ず等しいか改善されていることが分かるが，その度合いは小さく，構築型の Heuristic1 だけでもほぼ最適に近い解を算出している．

実行時間については，特に問題のサイズが大きくなるにつれ，Heuristic2 は Heuristic1 に比べて数 100 倍以上の時間を要している．また，同じサイズであっても，総コスト上限 C の値が大きくなるにつれ，実行時間も増大する傾向がうかがわれる．しかし，解の精度と実行時間とはトレードオフの関係にあるので，これだけの結果からは，どちらの手法が優れているかは断言できない．

また，総コスト上限の違いについて見てみると，Heuristic1，Heuristic2 とも C の値が大きくなるほど CPU 時間が増加する傾向が見られるが，解の精度については影響が見られない．

グラフの枝密度の違いについては，枝の数が多くなるほど解の精度が悪くなり，実行時間も要する例が若干増えてくるが，どちらも大きな差があるとはいいがたい．ヒューリスティック解法の場合，枝密度に関しては，比較的安定な振舞いを示していることが分かる．

5.2 2 つの分枝限定法

分枝限定法においては，主に実行時間に着目して，2 つの手法の比較を行う．**表 2** において，formu. とあるのは，3 章で説明したように問題を数理計画による定式化を行い，線形緩和を用いた分枝限定法（定式化型）を示し，divi. とあるのは，4 章で説明した問題の

表 2 分枝限定法の比較（CPU：秒）
Table 2 The results of branch-and-bound procedures.

$\|V\|$	$\|E\|$	C	C/MST	CPU	
				formu.	divi.
10	20	20	0.44	11.70	0.10
10	20	30	0.54	6.21	0.03
10	30	20	0.41	1.01	0.05
10	30	30	0.53	35.50	0.06
20	50	50	0.33	603.00	0.28
20	50	100	0.70	368.00	1.06
20	100	50	0.39	785.00	0.58
20	100	100	0.74	1794.00	2.51
30	100	50	0.19	979.40	0.88
30	100	100	0.41	19245.00	16.10
30	200	50	0.19	1124.00	0.83
30	200	100	0.41	19992.00	20.00

性質を利用した分枝限定法（分割型）を示している．また，C/MST は，最小木の値に対する総コスト上限の割合を示しており，この値から最適解がグラフのどれくらいの点あるいは枝をカバーしているかのおおよその見当をつけることができる．このことについては，次の 5.3 節でさらに詳しく観察する．

定式化型と分割型とを比較した場合，実行時間においては分割型の方が圧倒的に早く最適解を得ている．このことは問題のサイズが大きくなるにつれ，さらに顕著になる傾向がある．また，グラフの枝の数 $|E|$ の違いによっては，実行時間に目だった差は見られないが，同じサイズの問題であっても総コスト上限 C の違いによっては，両分枝限定とも C の値が大きい方が，実行時間も要していることが観察される．このことをさらに詳しく見るため，C/MST の値に着目して，次の 5.3 節で実験を行う．

5.3 総コスト上限と最小木の割合

5.2 節で見てきたように，同じサイズの問題であっても，総コスト上限の値によって計算時間に大きく影響することがある．それをより詳しく見るために，**図 6** では，$|V| = 20$，$|E| = 30$ の場合について横軸に最小木の値に対する総コスト上限の割合をとり，縦軸には分割型分枝限定法を用いた場合の計算時間を示している．

図 6 より，最小木の値に対する C が 0.6 あたりの問題が，最適解を得るのに一番困難であり，また困難な問題が分布している範囲も意外に狭いことが分かる．今回の問題では根が指定されているため，最小木の値の半分よりも若干大きめであるときに，すべての点の約半分を解に含むであろうことが予想される．この予想は，組合せの数から考えれば，グラフの約半分の点を解に含む場合が最も代替解が多く，またその数はピーク値付近において急激に増加することからも説明ができる．

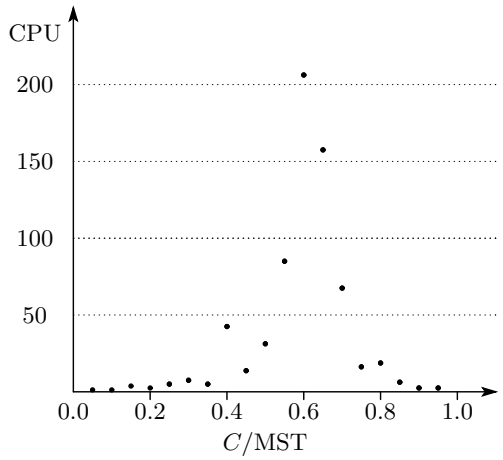

図 6 総コスト上限/最小木コストの違いによる CPU 時間（秒）の変化
Fig. 6 CPU time for the ratio of C to the value of minimum spanning tree.

6. ま と め

本研究では，最大利得部分木問題を提案し，冒頭にその $\mathcal{NP}$-困難性を証明した．ヒューリスティック解法としては，構築型と逐次改善型の 2 つを提案し，精度と CPU 時間に着目した計算機実験を行った．また，グラフの密度や総コスト上限の変化に対する振舞いも同時に観察した．さらに厳密解法として，数理計画法による定式化を行い，線形緩和による分枝限定法と，問題の構造を利用して，枝（点）の数を限定した問題に分割するアプローチの 2 つの分枝限定法を提案した．その際に，点や枝の距離を利用した上下界値などを求める貪欲的な手順を開発し，分枝限定法における効果的な利用法を示した．計算機実験の結果，分割型の方が圧倒的に早く最適解を得るという優れた結果を示した．また，総コスト上限の値によって効率が大きく左右されることを観察し，それについてより詳しい計算機実験も行った．

今後の課題としては，分割型をもとにして，さらに下界値を改善する手法の開発および分枝限定法による厳密解法の開発を行い，より大規模な問題にも対応していくことが考えられる．

参 考 文 献

1) Ahuja, R., Magnanti, T. and Orlin, J.: *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, New Jersey (1993).
2) Balas, E. and Toth, P.: Branch and Bound Methods (in Chapter 10), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Lawler, E., Lenstra, J., Rinnooy Kan, A. and Shmoys, D. (Eds.), John Wiley & Sons (1984).
3) Kataoka, S., Araki, N. and Yamada, T.: Upper and Lower Bounding Procedures for Minimum Rooted k-Subtree Problem, *European Journal of Operational Research*, Vol.122, pp.561–569 (2000).
4) Kataoka, S., Yamada, T. and Morito, S.: Minimum Directed 1-Subtree Relaxation for Score Orienteering Problem, *European Journal*

of Operational Research, Vol.104, pp.139–153 (1998).

(平成 11 年 9 月 2 日受付)
(平成 12 年 12 月 1 日採録)

片岡 靖詞（正会員）

昭和 60 年早稲田大学理工学部工業経営学科卒業．同大学院工業経営学専攻修士課程修了，同博士課程を経て，平成 2 年防衛大学校情報工学科に任官．現在，防衛大学校情報工学科助教授．博士（工学）．最適化ハンドブック（朝倉書店）整数計画法邦訳，OR 辞典組合せ最適化担当等，整数計画法，組合せ最適化の分野で活躍中．

星崎 康広

平成 8 年早稲田大学理工学部工業経営学科卒業．平成 10 年同大学院経営システム工学専門分野修士課程修了．同年富士写真フイルム（株）入社．オペレーションズ・リサーチを研究し，現在，社内コンサルタントとして活動中．日本オペレーションズ・リサーチ学会会員．

森戸 晋

昭和 44 年早稲田大学理工学部工業経営学科卒業．昭和 51 年米国 Case Western Reserve 大学 OR 学科 Ph.D. 同大学助教授，筑波大学社会工学系助教授を経て，現在，早稲田大学理工学部経営システム工学科教授．組合せ最適化や離散型シミュレーションの応用を中心とするオペレーションズリサーチの研究に従事．