情報処理学会論文誌 Vol.57 No.11 2445–2455 (Nov. 2016)

離散型・時間/費用トレードオフ最小全域木問題

片岡 靖詞[1,a)] 村崎 暢彦[2]

受付日 2016年4月12日, 採録日 2016年7月5日

概要：無向グラフにおいて，各枝にはモードと呼ばれる時間と費用のペアが与えられており，モードは複数ある．そして，あるモードは時間はかかるが費用は安い，別のモードは短時間で済むが費用は高いというトレードオフの関係があるものする．これらの枝モードからたかだか１つを選ぶことで，限られた予算内で，総時間を最小にする全域木を求める問題を考える．時間と費用は与えられたモードからしか選べないことから，この問題を離散型・時間/費用トレードオフ最小全域木問題と呼ぶことにする．時間/費用トレードオフという問題設定は，プロジェクトスケジューリングの分野では古くから研究がなされているが，ネットワーク計画法に同様の設定を導入した問題はほとんどみあたらない．この問題に対し，ラグランジュ緩和をもとに上下界値を求め，釘付けテストを行う．釘付けテストの効果は高く，分枝限定法においても，ごくわずかな枝やモードを決定変数の対象とすればよく，先行研究よりも効率良く解くことに成功した．また，時間と費用というモードの間に凸関数・凹関数・線形関数のような特性のあるモデルについても詳細に計算機実験を行い，各特性がアルゴリズムや最適解の構造に及ぼす影響についても調べた．

キーワード：時間/費用トレードオフ，プロジェクトスケジューリング，最小全域木，ラグランジュ緩和，釘付けテスト，分枝限定法

Discrete Time/Cost Tradeoff Minimum Spanning Tree Problem

Seiji Kataoka[1,a)] Masahiko Murasaki[2]

Received: April 12, 2016, Accepted: July 5, 2016

Abstract: Consider an undirected graph. For each edge, a pair of parameters, time and cost, which we call mode, is given. Normally there exist a number of modes on each edge, and these modes are on the relationship of tradeoff, such that one mode is slow-time and low-cost, and another mode is rapid-time and high-cost. By choosing at most one mode from each edge, we consider a problem of finding a spanning tree that minimizes total time within budget constraint. As the useable modes are prescribed in advance, we call this problem *the discrete time/cost tradeoff minimum spanning tree problem*. The relationship of time/cost tradeoff is often seen in the subject of project scheduling problems and these problems have been studied before. But in the subject of network programming problems, we seldom see studies that have this kind of tradeoff relationship among modes. For this problem, we propose upper and lower bounds procedures based on Lagrangian relaxation, and apply these bounds to the pegging test. The effectiveness of the pegging test is so high that, by eliminating many fixed variables and reducing the size of problem, the branch-and-bound algorithm succeeds in solving the problem quicker than the results of past research. Furthermore, we observe several models with specific time/cost relationships, such as convex, concave and linear functions.

Keywords: time/cost tradeoff, project scheduling, minimum spanning tree, Lagrangian relaxation, pegging test, branch-and-bound algorithm

1. はじめに

最小全域木問題は，グラフ・ネットワークの基本的な問題の１つであり，諸制約が付加された問題の緩和問題や部分問題として現れ，また通信網や回路の設計などにも応用

[1] 防衛大学校情報工学科
Department of Computer Science, National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan
[2] 陸上自衛隊
Japan Grand Self Defense Force
[a)] seiji@nda.ac.jp

例が知られている [2], [3], [25]．それらの問題では各枝に何らかの数値データが与えられるが，現実には別の要因などがともなうことがり，事前に確定した数値データを付与できるとは限らない．たとえば，枝として通信ケーブルの設置を考える場合，設置方法によって通信リスクがともなうことがある．高い場所に柱を立てたり床下や天井裏に収めるなど費用をかければ通信リスクを軽減できる．一方，現地の形状に合わせてむき出し状態でケーブルを張るなどすると安上がりだが通信リスクは高くなる．このほかにも折衷案となる通信リスクと設置費用のペアがいくつかあり，これらのペアのことモードと呼ぶ．そして，各モード間における通信リスクと設置費用とはトレードオフの関係にあり，本研究では各枝を採る・採らないに加え，採る場合は適切なモードを選択することで，限られた総予算の中で総通信リスクを最も小さくするような全域木を求める問題を考える．

各モード間を構成する 2 種類のデータがトレードオフの関係にある問題設定は，プロジェクトスケジューリングの分野においては PERT・CPM として古くから研究されており [20], [22]，そして時間/費用トレードオフという用語を含む問題群もいくつも知られている [5], [23]．トレードオフ関係というのは，前述の例におけるリスクと費用，プロジェクトスケジューリングにおける時間と費用，そのほかにも考えられるが，本論文ではこの問題のことを，多くの研究があるプロジェクトスケジューリングでの呼称に準じ，またいくつかの限られたモードのうちから 1 つを選ぶという意味で，**離散型・時間/費用トレードオフ最小全域木問題**（Discrite Time/Cost trade-off Minimum Spanning Tree: D-T/CMST）と呼ぶことにする．

まず，Değirmenci ら [5] に基づき，プロジェクトスケジューリングにおける資源の配分に関わる研究を振り返る．プロジェクトスケジューリングにおいて，資源（資金・人手・機械設備など）の配分に関わる問題は，資源計画問題（resource planning problem）と時間/費用トレードオフ問題（time/cost trade-off problem）とに大別できる．資源計画問題は，さらに資源の投入割合を決める資源割合問題（resource leveling problem）と使える資源の時間的要因を考慮して割り当てる資源割当問題（resource allocation problem）とがある．これら資源計画問題についての全般的な解説は，Herroelem ら [12] や Hartmann ら [10] に詳しい．一方，時間/費用トレードオフ問題も，決められた完了時刻で仕上げるための最小費用を求めるデッドライン問題（deadline problem），限られた予算内で完了時刻を最小化する予算問題（budget problem），そして時間/費用の非劣解を求める時間/費用曲線問題（time/cost curve problem）の 3 つがあり，これらの問題はどれも NP 困難であることが証明されている [4], [6]．時間/費用トレードオフ問題についても全般的な解説は，Vanhoucke ら [23] に詳しい．

本研究で扱う問題は，時間/費用トレードオフ型の予算問題に対応しているが，プロジェクトスケジューリングにおける予算問題の研究には，古くは Robinson [19]，Elmaghraby [7] などがあり，これらは動的計画法によるものである．また，Skutella [21] は近似解法を提案しており，厳密解法としては Hazir ら [11] がベンダースの分解原理を利用した解法を提案している．一方，近年では商用ソフトも充実しており，より大規模な数理計画問題が高速に解かれるようになってきている．そのような状況下で Değirmenci ら [5] は，時間/費用トレードオフ型の予算問題を線形整数計画法として定式化し，商用ソフトも活用した分枝限定法を提案している．また Hafizoğlu ら [9] では，デッドライン問題においても，定式化をもとにした分枝限定法を試みている．これらを見ても，プロジェクトスケジューリングの分野においても，数理計画的アプローチを試みているものは，比較的最近のことであり研究数も少ないのが現状である．

一方，グラフ・ネットワーク問題においても，各枝に時間・費用といった 2 種類以上のデータが与えられた問題はいくつか知られている．たとえば Yamada ら [24] による knapsack constrained maximum spanning tree problem では，時間を目的関数，費用をナップサック型制約として総時間最小となる全域木を求める問題である．この問題には，the minimum spanning tree problem subject to side constraint [1] など名称の異なる研究がいくつか存在するが，いずれも各枝に 2 種類のデータはあるものの，本研究で扱う D-T/CMST のようにトレードオフ関係にある複数のモードという概念はない．さらに，resource constrained MST [8] という問題もあり，これはナップサック制約に相当する資源制約が複数あり，それらを満足する最小木を求めるものである．これらの資源制約を目的関数に持っていったものとして，multi-objective MST [17] というものもあり，複数ある目的関数値の min-max あるいは max-min となる全域木を求める問題である．しかし，いずれも 1 つの枝に複数のデータが割り当てられている点で類似はしているものの，複数のモードという概念はなく，直接的に D-T/CMST の効率的なアルゴリズム開発につながる可能性は低い．

著者らが調べた範囲で D-T/CMST と等価なモデルとして先行研究といえるものには Kataoka-Yamada [13]（以降 KY と略記）がある．KY ではモード数を 2 に限定しており，2 重枝グラフに変換したうえで，knapsack constrained MST [24] を適用した解法を提案している．本研究では，モード数が 2 よりも多い場合を扱うので，時間/費用のトレードオフ関係に，資金投入によって最初は効果的に時間短縮ができるが次第に逓減してくる凸性を持つものや，資金を投入すればするほど時間短縮が進む凹性を持つものなど，さまざまな場面を想定して興味深い諸性質を導く．ま

た，解法としても KY のように多重枝グラフに変換することなく，多モードであることを考慮しながらも，通常のグラフを用いたアルゴリズムを提案する．

本論文の構成は，2 章で問題の定義と定式化を行う．そして 3 章では，通常のグラフにおいて，多モード性であることを考慮しながら上下界値を求める手順について説明する．4・5 章は，釘付けテストと分枝限定法であるが，ここでも頻繁に繰り返し解く最小木問題や解として得られる全域木は単重枝グラフで求めるが，問題の多モード性を考慮しながら解く際の注意点を詳しく述べる．6 章は計算機実験であり，まずは 2 モード問題で KY と比較し，提案するアルゴリズムの優越性を示す．さらに多モードという特性を生かし，2 モード問題では得られない諸性質を詳しく見ていく．7 章で本論文のまとめを行う．

2. 問題の定義と定式化

点集合を V，枝集合を E とする無向グラフ $G=(V,E)$ を考える．各枝にはモード数 m_e（$e \in E$）の時間と費用のペアがあり，モード k の時間と費用をそれぞれ t_e^k，c_e^k（$e \in E$，$k=1,2,\ldots,m_e$）とする．そしてこれらは，式 (1) のようにトレードオフ関係が成り立っているものと仮定する．

$$t_e^k \geq t_e^\ell, \quad c_e^k \leq c_e^\ell, \quad 1 \leq k < \ell \leq m_e \qquad \forall e \in E \quad (1)$$

この仮定を満たしていないモードは，他のあるモードに比べて，費用が高くしかも時間がかかることを意味するので，そのようなモードが最適解に選ばれることはない．

変数 x_e^k は枝 e がモード k として全域木を構成するときに 1，そうでないときに 0 となる 0-1 変数とし，また点集合 S（$\subseteq V$）に対し，両端点とも S に含まれている枝集合を $E(S)$ とすると，D-T/CMST は次のように定式化することができる．

$$\min \sum_{e\in E}\sum_{k=1}^{m_e} t_e^k x_e^k \quad (2)$$

$$\text{s.t.} \sum_{e\in E}\sum_{k=1}^{m_e} c_e^k x_e^k \leq B \quad (3)$$

$$\sum_{k=1}^{m_e} x_e^k \leq 1 \quad \forall e \in E \quad (4)$$

$$\sum_{e\in E}\sum_{k=1}^{m_e} x_e^k = n-1 \quad (5)$$

$$\sum_{e\in E(S)}\sum_{k=1}^{m_e} x_e^k \leq |S|-1 \quad \forall S \subseteq V \quad (6)$$

$$x_e^k \in \{0,1\} \quad (7)$$

式 (2) は目的関数で，式 (3) は費用の総和が決められた総予算 B 以内でなければならないことを示す．式 (4) は，モードは各枝からたかだか 1 つしか選ぶことができないことを示す．式 (5) と (6) は，全域木であることを示す．ここで，枝 e の両端の点が S になっているとき，式 (6) は式 (4) と一致するので，式 (4) は無視できる．このことは，上下界値を求める際に，多数のモードがあっても，うまく考慮しながら全域木を求めれば，選ぶべきモードはたかだか 1 つに決められることを意味する．この性質が，本研究のアルゴリズム効率化に寄与している．

一方，KY では各枝 e に対しモード数 m_e だけの多重枝を設け，それぞれに時間 t_e^k，費用 c_e^k（$k=1,\cdots,m_e$）を与えれば，D-T/CMST はナップサック制約付き最小全域木問題 [1], [24] を用いて解けることを示している．このことは，D-T/CMST が NP 困難であることを示すと同時に，KY のアルゴリズムの基本的な考え方ともなっている．

3. 上・下界値

予算制約式 (3) にラグランジュ乗数 λ（≥ 0）を掛けて目的関数に組み込み，最小全域木問題を解くことにより，D-T/CMST の下界値を求めることができる．KY [13] でもラグランジュ緩和を用いて下界値を求めているが，彼らは枝のモード数が 2（$m_e=2$，$e \in E$）に限定した場合を扱っており，2 重枝グラフに変換して最小全域木を求めている．また，多重枝グラフに変換すると枝数も多くなり，Prim や Kruskal のアルゴリズム [2] を用いるとしても，2 本の枝で構成される小さな閉路やカットの扱いが大きな負担になる．さらには，多重枝グラフに変換してしまうと，この後のラグランジュ双対問題や釘付けテスト，そして分枝限定法を行う際に最小全域木問題を何度も繰り返し解くことは効率的ではない．

定式化の際，式 (4) は全域木であることを満足していれば無視できることを説明した．つまり，式 (3) をラグランジュ緩和すれば，多モードのうち最も目的関数の係数が小さくなるモードだけが最小全域木の枝のモードとして候補となる．したがって，最小全域木を求める際には，式 (8) に従って，$w_e(\lambda)$ を各枝の重みとしたグラフ G 上で解き，その最適値を $MST(\lambda)$ とする．

$$w_e(\lambda) = \min_{k=1,2,\ldots,m_e} \{t_e^k + \lambda c_e^k\} \qquad \forall e \in E \quad (8)$$

ただし，このとき $w_e(\lambda)$ として選ばれた枝 e のモード番号を k_e として記憶しておくことが重要である．また，枝に対しても設定されたモード番号を $e^k \in T$ のように明示しておく．

このようにして，λ を与えることで $w_e(\lambda)$（$e \in E$）が決まり，$w_e(\lambda)$ において最小全域木および最適値 $MST(\lambda)$ が計算できるので，D-T/CMST のラグランジュ緩和による下界値は式 (9) のように求められる．

$$L(\lambda) = MST(\lambda) - \lambda B \quad (9)$$

この $L(\lambda)$ を λ の関数と見なすと，区分線形凹関数を描

くことが知られており [2]，最大の下界値を与える λ^* を求めるためのラグランジュ双対問題も，ラグランジュ乗数 λ が 1 つしかないので，2 分探索法で求めることができる．

2 分探索を行う中で，λ の値が変わるたびに $w_e(\lambda)$ も毎回求め直すことになるが，現在の λ よりも大きな値に λ が変わったときは $k = 1, \ldots, k_e$ までのモードを，現在の λ よりも小さく λ が変わったときは $k = k_e, \ldots, m_e$ のモードで $w_e(\lambda)$ を求め直せばよい．なぜなら，ある枝 e と 2 つの λ^k，λ^ℓ $(\lambda^k \geq \lambda^\ell)$ に対し，λ^k のときは第 k モードで $w_e(\lambda^k)$ が得られ，λ^ℓ のときは第 ℓ モードで $w_e(\lambda^\ell)$ が得られるとする．$w_e(\lambda)$ の定義より $t_e^k + \lambda^k c_e^k \leq t_e^\ell + \lambda^k c_e^\ell$ かつ $t_e^\ell + \lambda^\ell c_e^\ell \leq t_e^k + \lambda^\ell c_e^k$ である．したがって，$\lambda^\ell(c_e^\ell - c_e^k) \leq t_e^k - t_e^\ell \leq \lambda^k(c_e^\ell - c_e^k)$ となる．$\lambda^\ell \leq \lambda^k$ なので，$c_e^\ell - c_e^k \geq 0$，$t_e^k - t_e^\ell \geq 0$．これは仮定 (1) より $k \leq \ell$ のときに成り立つ．したがって，λ^k において $w_e(\lambda^k)$ が求められているとき，より小さな λ^ℓ を与えた場合に $w_e(\lambda^\ell)$ を求めるには，モード番号 ℓ は k 以上の所で探せばよい．

また，2 分探索法の途中で得られた全域木のうち，予算制約を満足しているものは実行可能解であり，そのうち総時間が一番小さくなるものを近似解（上界値）とできる．この近似解は，与えられた問題が実行可能であれば，ラグランジュ双対問題を解く 2 分探索法の中で必ず得ることができる．なぜなら，最適な λ^* より大きな $\bar{\lambda}$ $(> \lambda^*)$ において得られる全域木を $\bar{T}$ とすると，$\sum_{e \in \bar{T}} t_e^{k_e} + \lambda c_e^{k_e} - \lambda B$ は区分線形凹関数 $L(\lambda)$ のある線形区間の関数を示しており，その区間（直線）では微分可能であり，$\bar{\lambda}$ はその区間内にある．その区間において $\sum_{e \in \bar{T}} t_e^{k_e} + \lambda c_e^{k_e} - \lambda B$ を λ で微分すると，$\sum_{e \in \bar{T}} c_e^{k_e} - B$ となり，$\bar{\lambda}$ における傾きは右下がり，つまり負になるので，$\sum_{e \in \bar{T}} c_e^{k_e} < B$ となる．したがって，2 分探索で最適な λ^* をはさみうちにする大きい方の $\bar{\lambda}$ $(> \lambda^*)$ において得られる全域木 $\bar{T}$ が予算制約を満たす．したがってラグランジュ双対問題において λ^* を求めたときには，必ず近似解も得られている．特に $\sum_{e \in \bar{T}} c_e^{k_e} = B$ が成り立ったとき，$\bar{T}$ は最適解である．

得られた近似解（上界値）は，局所探索法によってさらに改善することができる．一般に木構造を解とする問題の局所探索法は，現在の木に補木枝 e を 1 本加え，できた閉路 $C(T+e)$ の中から別の 1 本の枝を取り除くことで木を逐次変遷させていくことにより行う．本問題でも同様の局所探索法を行うが，枝に複数のモードが設定されているので，木に入っていない枝 $(e \notin T)$ を加える際にも，m_e モードあるうちのどれにするか選ばなければならない．また，$e \in T$ の枝であっても別のモードと交換することにより解が改善できる場合を考慮する必要がある．特に $e \in T$ の枝を選ぶときには，式 (1) の仮定より，現在の木に用いられている枝 e のモード番号 k_e よりも大きな番号のモードと交換することで改善の可能性がある．これらに注意して次のアルゴリズム LocalSearch を示す．

Algorithm *LocalSearch*

Input: T, initially the T is obtained through Lagrangian dual algorithm.
Return: Inproved T
do
  for each $e \in E$ **do**
    if $e \notin T$ **then**
      for $k = 1$ to m_e **do**
        Try exchanging e of mode k and each edge in $C(T+e)$, if improved, keep the edge and mode.
      end for
    else if $e \in T$ **then**
      for $k = k_e + 1$ to m_e **do**
        Try exchanging e of mode k and k_e, if improved, keep the mode k.
      end for
    end if
  end for
  if The edge-and-mode that improves T most is found **then** Exchange them and improve T.
  end if
while T is improved.

4. 釘付けテストとその効果

釘付けテストは，ある枝やモードを採る（採らない）と固定して求めた下界値が暫定値（その時点での最良上界値）$BestUB$ を超えれば，その枝やモードは最適解において採らない（採る）と判定し，問題を縮小するものである．ラグランジュ緩和による下界値の計算には，ラグランジュ双対問題を解いたときの λ^* を用い，各枝の重みは，式 (8) に従って $w_e(\lambda^*)$ とし，そのときの最小全域木を解いた T を枝の固定基準とする．

釘付けテストも，各枝に複数のモードがあることを考慮し，$e \notin T$ の枝を採ると固定するときは，m_e モードのそれぞれに対して採ると固定して下界値を求める必要がある．また $e \in T$ のときは，現在の T においてモード k_e で枝が採られているので，e^k を採らないと固定するときは，モード k_e 以外のモードと交換する場合と，どのモードでも木に入らないように固定する場合の 2 通りあることに注意する．特に e のどのモードでも木に入らないように固定する場合は，$T \setminus \{e\}$ で定義される基本カットセットのうち，最小の $w_e(\lambda^*)$ を持つ枝を探す必要があるが，このような場合は $n-1$ 回だけなので，アルゴリズムの中では枝 e についてはどのモードでも解に入らないようにして最小全域木を解いている．

また，釘付けテストを行った際に，ある枝・モードを固定して求めた下界値が，上界値を超えられなかったとき，

Algorithm *PeggingTest*

Input: λ^*, T, $BestUB$
Return: F_1, F_0, $p_e^k \forall e \in E \setminus (F_1 \cup F_0)$
  $F_1 = F_0 = \emptyset$
  for each $e \in E$ **do**
    if $e \notin T$ **then**
      $w_{\max} = \max_{f \in C(T+e)-e} w_f(\lambda^*)$
      for $k = 1$ **to** m_e **do**
        if $L(\lambda^*) - w_{\max} + (t_e^k + \lambda^* c_e^k) > BestUB$ **then**
          $F_0 = F_0 \cup \{e^k\}$
        else
          $p_e^k = BestUB - (L(\lambda^*) - w_{\max} + (t_e^k + \lambda^* c_e^k))$
      end for
    else if $e \in T$ **then**
      for $k = 1$ **to** m_e except k_e **do**
        if $L(\lambda^*) - w_e(\lambda^*) + (t_e^k + \lambda^* c_e^k) > BestUB$ **then**
          $F_0 = F_0 \cup \{e^k\}$
        else
          $p_e^k = BestUB - (L(\lambda^*) - w_e(\lambda^*) + (t_e^k + \lambda^* c_e^k))$
      end for
      if $MST(\lambda^* : E \setminus \{e\}) - \lambda^* B > BestUB$ **then**
        $F_1 = F_1 \cup \{e^k\}$
      else
        $p_e^k = BestUB - (MST(\lambda^* : E \setminus \{e\}) - \lambda^* B)$
    end if
  end for

釘付けはできず未固定となる．だが，そのときの不足分を p_e^k として記録しておく．この値は分枝限定法において，分枝変数を選ぶ際の判断基準として活用する．

次のアルゴリズム PeggingTest では，採ると固定される枝・モード集合を F_1，採らないと固定される枝・モード集合を F_0 とする．当然のことながら，ある枝 e がモード k 採ると固定された場合，その枝では k 以外のモードではすべて採られないと固定される．

5. 分枝限定法

分枝限定法は，問題を組織的にいくつかの子問題に分割し，上下界値の情報などを利用して最適解が存在しえない子問題を早い段階で見切ることにより，最終的に最適解を求めようというものである．分枝限定法を構築する際には，分枝ルールや分枝変数の選び方など，全体的な効率にも大きく影響するいくつかの戦略がある．

本論文では詳細を割愛するが，分枝ルールとして以下のことを予備的な実験で確かめている．ある枝 e に関してどのモードで固定するか分枝する際，e をモード 1 で採る，e をモード 2 で採る，$\cdots$，e をモード m_e で採る，枝 e は最小木の解としてどのモードでも採らないという分枝ルールが考えられる．この分枝ルールは複数ナップサック問題でも行われていた方法である [16]．しかしながら，列挙木の 1 つの頂点から幅広く分枝することは，最適解を含む可能性がある子問題も増える傾向がある．このため，列挙木が大きくなって計算時間も増大することが予備的な実験で分かった．したがって，以降説明する分枝ルールは，1 つの分枝頂点からは，ある枝 e をモード k として採る/採らないという 2 分枝ルールに従うことにした．この 2 分枝方式は，複数ナップサック問題においても優越性が報告されている [15], [18]．

本研究での分枝変数の選択方法は，ラグランジュ緩和問題で得られた最小全域木 T のうち，採ると固定された枝・モード集合 F_1 に属さない枝で，p_e^k の小さなものから優先的に選ぶことにした．p_e^k の値は，釘付けテストで固定できなかったときに，下界値が上界値に満たなかった差分値である．これら p_e^k 値の小さい枝は，単独では固定できなくても，別の枝やモードの固定条件が重なることにより，早い段階で子問題が見切られる可能性が高まると考えられる．

また，6 章の計算機実験結果でも示すが，本問題では釘付けテストの効果が十分高い．したがって，採ると固定した F_1 に属する枝・モードの総時間を $FixedT(:= \sum_{e^k \in F_1} t_e^k)$，総費用を $FixedC(:= \sum_{e^k \in F_1} c_e^k)$ とするとき，ある子問題において $FixedT$ が暫定値 $BestUB$ を超える，$FixedC$ が予算 B を超えるという条件を満足したときは，下界値などを求める前に見切ることができる．また，採らない枝・モード F_0 が増えると，最小木を解くことが実行不可能になるなどのケースも出てくる．このときも子問題を見切る．

さらに，この見切り条件を強化するために，残りの採るべき枝数から，$FixedT$ や $FixedC$ の最低限の増加量を評価する．ある子問題において決定しなければならない残りの枝数は $n - 1 - |F_1|$ 本である．そこで，釘付けテスト直後において，未固定で残っている枝のうち t_e^k の小さい方から $rest$ 分の和を $t_r(rest)$，同様に c_e^k も小さい方から $rest$ 分の和を $c_r(rest)$ として記録しておく．$t_r(rest)$，$c_r(rest)$ は，残り $rest$ 分の枝・モードを採らなければならないときに，固定される時間や費用の下限増加量を示す．

下界値は釘付けによって縮小されたグラフ $G(F_1, F_0)$，つまり F_1 の枝は縮約し，F_0 の枝・モードは除く．それ以外の枝は λ^* を用いて $w_e(\lambda^*)$ に設定した最小木問題を解くことで得る．実際には λ^* は各子問題で若干変わるが，各子問題でラグランジュ双対問題を解く手間は無視できず，またルート部で十分な下界値と釘付けができているので，そのまま λ^* を用いることにした．また F_1, F_0 を固定したグラフの最小木から下界値を得るためには式 (10) のように計算できる．

$$L(\lambda^* : F_1, F_0) = MST(\lambda^* : F_1, F_0) + FixedT - \lambda^*(B - FixedC) \quad (10)$$

Algorithm BAB(F_1, F_0)

Input: F_1 and F_0 in the current subproblem
Return: Optimal solution. Optimal value is $BestUB$.
$rest = n - 1 - |F_1|$
if $FixedT + t_r(rest) \geq BestUB$ **then return**
if $FixedC + c_r(rest) > B$ **then return**
Solve $MST(\lambda^* : F_1, F_0)$
if MST is feasible **then**
  if $\sum_{e^k \in T} c_e^k \leq B$ **then**
    if $\sum_{e^k \in T} t_e^k < BestUB$ **then**
      $BestUB = \sum_{e^k \in T} t_e^k$
    end if
  end if
  $LB := MST(\lambda^* : F_1, F_0) + FixedT - \lambda(B - FixedC)$
  if $LB > BestUB$ **then return**
  $e^k = \arg\min\{p_e^k | e \in T \setminus F_1, k = 1, \cdots, m_e\}$
  if no e^k is found **then return**
  BAB($F_1 \cup \{e^k\}, F_0$)
  BAB($F_1, F_0 \cup \{e^k\}$)
end if
return

これらの事項を組み込んで，次の再帰アルゴリズム BAB を示す．最初の F_1，F_0 は釘付けテストで得られた結果を引数として設定する．

6. 計算機実験

計算機実験は，2 モードと多モードの 2 つの視点で行う．6.1 節ではモード数をどの枝も 2 に限定し，KY（Kataoka-Yamada [13]）と同じ例題・同じ計算機環境で両者を比較し，アルゴリズムの性能比較を行う．評価の方法は，グラフの種類や規模・相関・予算制約の強弱・時間/費用の分布範囲を変化させ，釘付けテストの効果や計算時間などを見る．6.2 節は多モードモデルを扱い，モード数による影響や時間–費用の関係に凸関数や凹関数などの関係がある場合について考察する．

6.1 2 モードモデル：KY との比較

この節での実験では，モード数はすべての枝において $m_e = 2$ とする．2 モードに限定すれば，先行研究である KY との性能比較ができる．例題の生成法も KY に合わせ，費用 c_e^k を $[1,R]$（$R = 10^2, 10^3, 10^4$）の一様整数乱数で与え，所要時間 t_e^k（$k = 1, 2$）は，費用 c_e^k に応じて次の 3 種類の相関を与えるようにする．

無相関：各枝の t_e^k も独立した $[1, R]$ の一様整数乱数で与える．
弱相関：$t_e^k := \lfloor 0.8c_e^k + [1, 0.2R] \rfloor$
強相関：$t_e^k := \lfloor 0.8c_e^k + 0.1R/10 + [1, 0.01R] \rfloor$

いずれの場合も，t_e^k を発生させた後，仮定 (1) を満足するように c_e^k，t_e^k を並べ替え，モード番号も付け替える．また，予算 B は全域木の平均的な目的関数値を $R(n-1)/2$ と見なし，パラメータ α を用いて式 (11) のように設定する．

$$B = \alpha Rn \quad (\alpha = 0.2, 0.4, 0.6) \tag{11}$$

さらに使用計算機 DELL Precision T7500（Intel Xeon X5680（3.3 GHz）× 2）や乱数シードも KY とまったく同じものを用いる．

以下の表において，graph は点の数 n，枝の数 m とするとき，P_n^m は平面的なグラフ，K_n は完全グラフを意味する．gap はラグランジュ緩和による下界値と局所探索法において得られた上界値の差であり，この値が 0 になると釘付けテストをする前に最適に解けたことを意味する．fix0，fix1，unfix はそれぞれ 0 に固定された枝，1 に固定された枝，未固定の枝の割合（%）を示す．unfix が 0 になると，分枝限定法に入る前に，釘付けテストまでで最適解が得られたことを意味する．#subp は分枝限定法における分枝数，cpu は計算時間（秒）である．最後の KYcpu は KY の実行時間である．また表内の数値は各パラメータ設定において 10 問解いた平均値であるが，KY とはまったく同じ乱数シードを用いているので，すべての個々の問題において最適値は一致したことも確かめている．

表 1 はグラフの形状・規模の違いと費用–時間の相関の影響をみるための実験結果である．ラグランジュ緩和による下界値と局所探索による上界値との差 gap は，グラフの規模が大きくなるほど改善の可能性も高まり，gap がより小さくなる傾向が見られた．表中 - とあるのは，ラグランジュ緩和の時点で gap が 0.0 になってしまったため，その後に続く釘付けテストや分枝限定法に関する計測ができなかったことを意味する．そして gap の値が小さいと釘付けテストの効果も期待できる．実際に釘付けテストの効果は高く，釘付けされない unfix の割合は平面的グラフで数パーセント，完全グラフでは 1 パーセントを切る場合も多く見られた．ただし相関が強くなるとモードの違いによる対費用効果の差がなくなるため，平面的グラフではかなりの枝が unfix として残された．しかし完全グラフでは枝数そのものが多いせいもあって，割合としては費用–時間の相関の強さにかかわらず安定して釘付けに成功している．計算時間を見ると，2 重グラフに変換して扱っている KY と比較していずれの場合においても勝った結果が得られた．

表 2 は予算制約 B の変動に対する振舞いを見るためのものである．費用–時間の相関は無相関，乱数の範囲は $R = 10^3$ としている．ナップサックタイプの問題では，予算が半ばくらいのとき，入れる/入れないの選択肢が増えるため難しくなる傾向が観察されることが多いが，本問題では解の形状が木であるため，選ばれる枝の本数は決まっている．したがって予算 B が大きいほど（α が大きいほ

表 1 各相関に関する実験結果（$R = 10^3$，$\alpha = 0.4$）
Table 1 The results for correlations ($R = 10^3$, $\alpha = 0.4$).

相関	graph	gap	fix0	fix1	unfix	#subp	cpu	KYcpu
無	P_{50}^{127}	56.3	74.2	14.2	12.0	446.4	0.00	0.01
	P_{100}^{260}	49.3	75.2	14.4	10.7	1055.2	0.02	0.02
	P_{200}^{560}	20.6	79.8	15.8	4.5	1029.4	0.04	0.48
	P_{400}^{1120}	19.2	80.3	16.0	3.9	2012.2	0.16	0.93
	P_{600}^{1680}	11.1	81.1	16.8	2.1	2352.6	0.29	3.22
	P_{800}^{2240}	17.6	80.4	16.1	3.5	3038.6	0.52	7.86
	P_{1000}^{2800}	10.6	81.1	16.9	2.1	5236.6	1.23	12.29
	K_{40}	12.3	96.6	1.7	1.7	249.0	0.01	0.02
	K_{80}	4.0	98.5	1.0	0.5	336.4	0.02	0.17
	K_{120}	3.0	99.0	0.7	0.3	165.6	0.03	0.46
	K_{160}	1.9	99.3	0.6	0.1	92.6	0.03	1.12
	K_{200}	1.4	99.5	0.5	0.0	130.3	0.06	4.07
弱	P_{50}^{127}	41.9	72.0	12.1	16.3	863.4	0.01	0.03
	P_{100}^{260}	30.7	73.7	13.3	13.2	1725.6	0.03	0.08
	P_{200}^{560}	29.4	78.7	14.6	6.8	1568.0	0.06	0.25
	P_{400}^{1120}	12.3	79.9	15.7	4.4	2551.4	0.20	1.30
	P_{600}^{1680}	9.7	80.4	16.1	3.5	2117.9	0.26	5.25
	P_{800}^{2240}	9.7	80.4	16.2	3.4	3064.3	0.51	4.23
	P_{1000}^{2800}	8.9	80.5	16.3	3.2	2044.8	0.43	6.50
	K_{40}	8.6	96.9	2.1	1.0	87.6	0.00	0.01
	K_{80}	1.4	98.6	1.1	0.2	129.4	0.01	0.08
	K_{120}	3.9	99.1	0.7	0.2	180.3	0.03	0.20
	K_{160}	2.8	99.3	0.6	0.1	61.3	0.02	0.22
	K_{200}	2.7	99.5	0.5	0.1	165.8	0.08	2.25
強	P_{50}^{127}	9.1	62.5	2.5	35.3	3648.6	0.03	1.04
	P_{100}^{260}	3.2	67.1	5.5	24.8	3275.9	0.06	1.93
	P_{200}^{560}	4.1	66.9	3.0	30.2	8911.3	0.36	61.04
	P_{400}^{1120}	2.2	67.5	3.2	29.4	3843.8	0.30	109.47
	P_{600}^{1680}	10.1	67.0	3.9	29.1	15856.7	2.19	1734.38
	P_{800}^{2240}	1.8	71.4	7.2	21.4	94597.4	15.73	1675.32
	P_{1000}^{2800}	28.6	60.9	0.0	39.1	678648.0	151.22	—
	K_{40}	2.0	97.3	2.3	0.5	41.3	0.00	0.01
	K_{80}	1.7	98.6	1.1	0.2	169.8	0.01	2.53
	K_{120}	1.8	99.1	0.8	0.1	83.0	0.01	26.99
	K_{160}	0.0	-	-	-	-	0.00	191.80
	K_{200}	1.2	99.5	0.5	0.0	68.0	0.03	1703.55

ど），局所探索による枝交換の幅が広がり，gap は小さくなる傾向が見られた．しかしながら，この傾向はそれほど顕著な差ともいえず，計算時間には大きな違いは見られなかった．また KYcpu では平面的グラフで $\alpha = 0.4$ のときに若干計算時間を要しているが，これも大きな違いはない．また，ここでも本研究での計算時間の方が KY より勝っているという結果が得られた．

表 3 は費用および時間を与える乱数の範囲 R を変えたときの振舞いを見るための実験結果である．費用–時間の相関は無相関，予算パラメータは $\alpha = 0.4$ としている．費用–時間のとる値の幅が狭いと，gap も小さく，完全グラフでは上下界が一致して最適に解けてしまう例も目立った．一方，費用–時間のとる値の幅が広がると，おのずと gap も大きくなる傾向があり，gap が大きくなると釘付けテストの効果も下がるかと思われがちである．しかしながら，実験結果を見ると gap の影響はほとんど受けておらず，多くの枝やモードを固定することに成功している．その結果，分枝限定法における分枝数も，計算時間もそれほど影響を受けずに高速に最適解を得ている．一方，KY では，費用–時間のとる値の幅が広がるに従って徐々に計算効率が劣化していることが観察される．

表 2 予算制約変動に関する実験結果（無相関，$R = 10^3$）
Table 2 The results for budget constraint (No correlation, $R = 10^3$).

α	graph	gap	fix0	fix1	unfix	#subp	cpu	KYcpu
0.2	P_{50}^{127}	84.2	75.1	14.4	12.1	310.8	0.00	0.01
	P_{100}^{260}	89.6	75.2	14.6	10.4	611.6	0.01	0.04
	P_{200}^{560}	33.2	79.5	15.4	5.1	759.2	0.03	0.57
	P_{400}^{1120}	34.8	79.8	15.7	4.5	2137.0	0.17	0.99
	P_{600}^{1680}	27.8	80.3	16.1	3.6	1357.7	0.17	3.18
	P_{800}^{2240}	23.9	80.4	16.2	3.4	3348.3	0.58	5.60
	P_{1000}^{2800}	16.5	81.0	16.7	2.3	5665.2	1.27	6.84
	K_{40}	24.4	96.6	1.8	1.7	263.8	0.00	0.01
	K_{80}	12.5	98.4	1.0	0.7	360.3	0.02	0.20
	K_{120}	7.2	99.0	0.7	0.3	270.7	0.05	0.49
	K_{160}	3.4	99.3	0.6	0.1	157.7	0.05	1.30
	K_{200}	3.6	99.4	0.4	0.1	197.7	0.10	3.59
0.4	P_{50}^{127}	56.3	74.2	14.2	12.0	446.4	0.00	0.01
	P_{100}^{260}	49.3	75.2	14.4	10.7	1055.2	0.02	0.02
	P_{200}^{560}	20.6	79.8	15.8	4.5	1029.4	0.04	0.48
	P_{400}^{1120}	19.2	80.3	16.0	3.9	2012.2	0.16	0.93
	P_{600}^{1680}	11.1	81.1	16.8	2.1	2352.6	0.29	3.22
	P_{800}^{2240}	17.6	80.4	16.1	3.5	3038.6	0.52	7.86
	P_{1000}^{2800}	10.6	81.1	16.9	2.1	5236.6	1.23	12.29
	K_{40}	12.3	96.6	1.7	1.7	249.0	0.01	0.02
	K_{80}	4.0	98.5	1.0	0.5	336.4	0.02	0.17
	K_{120}	3.0	99.0	0.7	0.3	165.6	0.03	0.46
	K_{160}	1.9	99.3	0.6	0.1	92.6	0.03	1.12
	K_{200}	1.4	99.5	0.5	0.0	130.3	0.06	4.07
0.6	P_{50}^{127}	37.7	77.2	16.7	6.5	133.0	0.00	0.00
	P_{100}^{260}	18.5	78.6	17.1	4.5	299.2	0.01	0.02
	P_{200}^{560}	14.8	80.8	16.5	2.8	400.0	0.02	0.07
	P_{400}^{1120}	13.7	81.1	16.8	2.1	786.4	0.06	0.34
	P_{600}^{1680}	7.6	81.5	17.1	1.4	327.4	0.04	1.38
	P_{800}^{2240}	6.8	74.2	17.2	1.3	827.4	0.14	2.76
	P_{1000}^{2800}	6.7	83.0	17.0	1.6	451.2	0.10	4.11
	K_{40}	7.4	96.8	1.9	1.2	161.0	0.00	0.00
	K_{80}	1.8	98.7	1.2	0.2	41.1	0.00	0.06
	K_{120}	1.3	99.1	0.8	0.1	92.4	0.02	0.28
	K_{160}	1.1	99.3	0.6	0.1	14.0	0.00	4.52
	K_{200}	1.2	99.5	0.5	0.0	50.0	0.02	6.42

6.2 多モードモデルの特性

本節ではモード数が 2 より多いモデルについての実験結果を示す．多モードモデルでは時間–費用の間に凸関数や

表 3 乱数範囲に関する実験結果（無相関，$\alpha = 0.4$）
Table 3 The results for ranges of parameters (No correlation, $\alpha = 0.4$).

R	graph	gap	fix0	fix1	unfix	#subp	cpu	KYcpu
10^2	P_{50}^{127}	8.9	75.8	15.5	9.1	350.1	0.00	0.01
	P_{100}^{260}	5.1	75.9	15.0	9.3	2596.9	0.01	0.02
	P_{200}^{560}	4.0	80.0	16.0	4.2	449.6	0.02	0.05
	P_{400}^{1120}	2.0	80.7	16.5	2.7	530.8	0.04	0.61
	P_{600}^{1680}	1.4	81.8	17.4	0.9	190.2	0.02	1.52
	P_{800}^{2240}	2.4	80.7	16.5	2.9	2579.7	0.42	1.57
	P_{1000}^{2800}	1.8	81.1	16.8	2.0	255.8	0.06	1.81
	K_{40}	4.1	97.0	2.0	1.0	64.0	0.00	0.00
	K_{80}	1.1	98.7	1.2	0.2	67.0	0.01	0.02
	K_{120}	0.0	-	-	-	-	0.00	0.01
	K_{160}	0.0	-	-	-	-	0.00	1.02
	K_{200}	1.1	99.4	0.4	0.1	425.0	0.22	8.70
10^3	P_{50}^{127}	56.3	74.2	14.2	12.0	446.4	0.00	0.01
	P_{100}^{260}	49.3	75.2	14.4	10.7	1055.2	0.02	0.02
	P_{200}^{560}	20.6	79.8	15.8	4.5	1029.4	0.04	0.48
	P_{400}^{1120}	19.2	80.3	16.0	3.9	2012.2	0.16	0.93
	P_{600}^{1680}	11.1	81.1	16.8	2.1	2352.6	0.29	3.22
	P_{800}^{2240}	17.6	80.4	16.1	3.5	3038.6	0.52	7.86
	P_{1000}^{2800}	10.6	81.1	16.9	2.1	5236.6	1.23	12.29
	K_{40}	12.3	96.6	1.7	1.7	249.0	0.01	0.02
	K_{80}	4.0	98.5	1.0	0.5	336.4	0.02	0.17
	K_{120}	3.0	99.0	0.7	0.3	165.6	0.03	0.46
	K_{160}	1.9	99.3	0.6	0.1	92.6	0.03	1.12
	K_{200}	1.4	99.5	0.5	0.0	130.3	0.06	4.07
10^4	P_{50}^{127}	582.9	74.2	14.1	12.1	427.8	0.00	0.02
	P_{100}^{260}	446.8	75.6	14.7	9.9	704.0	0.01	0.06
	P_{200}^{560}	229.2	79.4	15.5	5.2	2212.6	0.08	0.71
	P_{400}^{1120}	185.8	80.3	16.0	3.8	1709.0	0.14	3.96
	P_{600}^{1680}	155.8	80.6	16.3	3.2	3929.0	0.49	7.42
	P_{800}^{2240}	114.0	81.0	16.7	2.4	2793.2	0.47	11.14
	P_{1000}^{2800}	93.6	81.1	16.9	2.1	4797.8	1.05	50.62
	K_{40}	142.1	96.5	1.6	0.2	300.8	0.01	0.02
	K_{80}	41.5	98.4	1.0	0.6	680.0	0.04	0.38
	K_{120}	23.5	99.0	0.7	0.3	755.6	0.13	2.26
	K_{160}	20.8	99.2	0.5	0.2	1049.4	0.34	7.82
	K_{200}	3.6	99.4	0.4	0.1	570.3	0.29	6.60

凹関数などの関係がある場合についてなど，より興味深い設定が可能になる．しかし，公平に比較できる別の先行研究結果が存在しない．したがって，実験に用いるグラフもKYにとらわれることなく，より一般性を持つものとして，グラフの密度を設定できるようにし，枝密度の違いによる振舞いも観察できるようにした．これは完全グラフを基本にして，各枝を指定した割合で採る/採らないに振り分けて生成したものである．また，6.1 節の実験では各 P_n^m や K_n は 1 種類ずつであり，それらに対して時間–費用の割当てを乱数により 10 通り生成して実験を行った平均である．それに対し，本節の実験では，各枝密度に対して 5 種類のグラフを生成し，それらの各グラフに対して時間–費用の割当てを乱数により 10 通り生成して実験をしている．つまり**表 4**，**表 5** の各数値は，50 回実験を行った平均値になっている．しかし，fix0，fix1，unfix を計算するときにおいては，分母値は生成されたグラフごとに異なってしまうため公平性を欠く．そこで，本実験においては，実際に生成された枝数ではなく，理論的な枝数，つまり (枝密度/100)×$n(n-1)/2$ をグラフごとに分母値が変わっても共通して用いることのできる分母値としている．したがって，fix0+fix1+unfix の値が微妙に 100 になっていないものが出ることを補足しておく．

表 4 問題の規模およびモード数増加に関する実験結果（$R = 10^3$，$\alpha = 0.4$）
Table 4 The results for sizes of problems and the number of modes ($R = 10^3$, $\alpha = 0.4$).

mode	n	den	gap	fix0	fix1	unfix	#subp	cpu
2	200	5	21.7	87.08	8.60	2.96	960.4	0.04
		10	14.2	93.62	4.35	1.40	990.4	0.06
		20	9.9	96.59	2.14	0.75	664.5	0.07
	400	5	11.1	93.99	4.48	1.08	1317.8	0.20
		10	6.4	96.57	2.27	0.46	903.1	0.23
		20	3.9	98.08	1.15	0.20	636.3	0.29
	600	5	7.8	95.93	3.04	0.60	1624.4	0.53
		10	4.3	97.64	1.54	0.25	1345.6	0.77
		20	2.8	98.49	0.76	0.11	723.8	0.79
	800	5	4.7	96.97	2.34	0.31	1617.2	0.91
		10	3.9	98.26	1.16	0.19	1344.8	1.42
		20	1.8	98.92	0.59	0.05	911.3	1.83
4	200	5	27.39	92.65	4.08	1.98	1518.8	0.07
		10	10.59	96.44	2.22	0.73	863.6	0.05
		20	8.9	98.21	1.07	0.39	1158.6	0.12
	400	5	10.3	96.63	2.16	0.71	2034.7	0.31
		10	10.1	99.88	1.08	0.44	1692.4	0.44
		20	5.4	98.73	0.55	0.14	1177.2	0.54
	600	5	7.2	97.71	1.49	0.37	1793.4	0.60
		10	5.8	98.50	0.74	0.18	1770.7	0.99
		20	3.5	98.95	0.38	0.71	1158.2	1.24
	800	5	5.6	98.25	1.13	0.23	1885.2	1.07
		10	3.3	98.93	0.58	0.07	2109.4	2.12
		20	3.4	99.26	0.28	0.05	889.8	1.81
8	200	5	17.34	95.18	1.84	1.57	2664.1	0.12
		10	14.7	97.64	0.93	0.80	2161.3	0.14
		20	9.0	98.71	0.50	0.26	1661.7	0.17
	400	5	8.9	98.08	1.05	0.42	2326.0	0.35
		10	7.7	98.53	0.52	0.21	2705.9	0.69
		20	7.4	99.05	0.25	0.12	2061.3	0.94
	600	5	6.9	98.62	0.72	0.23	2460.1	0.81
		10	5.8	98.97	0.36	0.11	2191.9	1.23
		20	5.0	98.97	0.18	0.05	1972.2	2.01
	800	5	5.4	98.92	0.55	0.14	2371.6	1.33
		10	4.9	99.21	0.27	0.70	2336.5	2.29
		20	5.2	99.40	0.13	0.04	4985.3	9.96

表 4 は，モード数を 2，4，8 の 3 種類，およびグラフ

表 5 時間と費用の想定に関する実験結果（$n = 400$，den $= 10$，$R = 10^3$，$\alpha = 0.4$）
Table 5 The results for time/cost relationships ($n = 400$, den $= 10$, $R = 10^3$, $\alpha = 0.4$).

								使用されたモード番号 (%)							
func	gap	fix0	fix1	unfix	#subp	cpu	trun	1	2	3	4	5	6	7	8
一様	7.7	98.5	0.5	0.2	2705.9	0.69	0	0.6	2.9	6.3	13.2	19.5	23.5	22.6	11.5
凸 (5)	3.2	95.7	0.4	0.4	103541.7	28.28	0	0.4	4.5	17.2	28.0	35.8	13.7	5.4	2.4
凸 (10)	3.4	95.8	0.5	0.4	68415.6	17.44	0	0.0	3.7	17.3	30.5	27.7	13.0	4.1	1.2
凸 (20)	4.1	99.0	0.5	0.6	25093.2	6.61	0	0.0	3.9	18.9	33.4	30.1	13.2	3.1	0.7
凹 (5)	28.3	98.3	0.4	0.6	227941.8	58.77	1	44.1	0.0	0.0	0.0	0.0	0.0	0.4	55.6
凹 (10)	27.5	98.3	0.4	0.6	405463.4	102.48	2	44.1	0.0	0.0	0.0	0.0	0.0	0.4	55.6
凹 (20)	27.7	98.3	0.4	0.5	119662.3	30.17	3	44.1	0.0	0.0	0.0	0.0	0.0	0.3	55.6
線形	94.2	97.8	0.4	1.1	722603.9	235.16	36	39.0	0.1	0.0	0.0	0.1	0.3	3.1	57.3

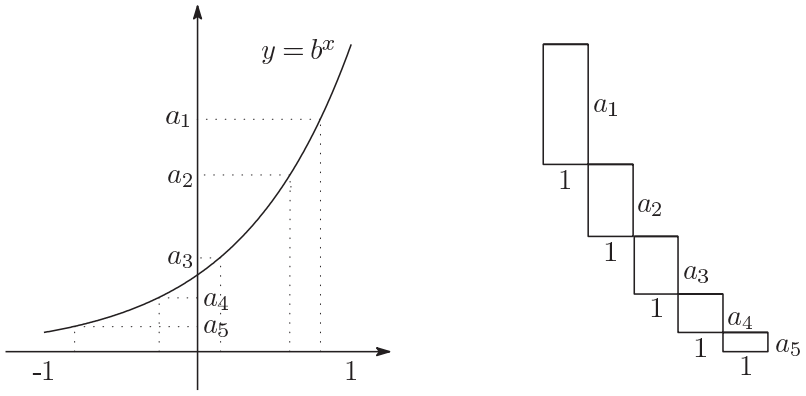

図 1 凸関数の作り方
Fig. 1 The way to make convex function.

の枝密度（den）を完全グラフに対して 5，10，20%で生成した問題を解いた結果である．費用–時間の乱数の幅は $R = 10^3$，予算パラメータは $\alpha = 0.4$ としている．モード数が増加しても，本研究のアルゴリズムでは，KY のように多重枝グラフに変換などしていないため，モード数増加による計算効率の劣化はほとんど見られない．また枝密度の増加は決定変数を増やすことになるが，釘付けテストが効果的に機能しているため，割合としては枝密度が高くなるほど，ほとんどの枝が固定されているという結果が得られた．そのため，全体的な計算時間は枝密度による影響も特徴的な傾向は見られず，本研究のアルゴリズムはモード数に対して頑健性が高いといえる．

これまでの多モード問題では，最安–最遅計画と最高–最速計画との間の各モードには，単に式 (1) で示される仮定，つまり横軸に費用，縦軸に時間をとった平面上に各モードを示すと，これらが単調減少に並んでいるだけであった．しかし，多モードであれば，これらに凸関数や凹関数，あるいは線形関数の関係を持たせることができる．

本研究では次のように凸関数（凹関数）を生成した．例ではモード数 6 の場合について説明する．まず，底を b とした指数関数 $y = b^x$ を考え，$-1 \le x \le 1$ の一様乱数により，5 つ（生成したいモード数 $6-1$）の値を得て大きい方から a_1，a_2，a_3，a_4，a_5 とする．これらは 1 を中心にした b から $1/b$ の値になっている．これらの値を用いて，底辺 1，高さ a_ℓ（$\ell = 1, 2, 3, 4, 5$）の長方形を考え，左上と右下の頂点が接するように並べる（**図 1**）．

次に，2 モードモデルのときと同様に [1,R] の一様乱数により，6 つの費用の値と 2 つの時間の値を生成し，$c^1 \le c^2 \le c^3 \le c^4 \le c^5 \le c^6$, $t^1 \ge t^6$ となるように番号付けをする．そして，図 1 のように並べた長方形をそれぞれ $c^2 - c^1$, $c^3 - c^2$, $\cdots$, $c^6 - c^5$ 倍する．最後に縦のサイズが $t^1 - t^6$ に収まるように調整する．凹関数の場合は，長方形の積み方を逆順にして，同じ要領で作る．

このように生成することで，指数の底 b の値により凸（凹）の膨らみ具合を調整することができる．適切な b の値であるが，図 1 は b の値が 4 で書かれている．$b = 2$ では見た目にはほとんど直線であり，凸の膨らみが感じられない程度にしかならない．したがって，実験では $b = 5, 10, 20$ を使用した．また，点の数は 400，枝密度は 10%，モード数は 8，乱数の範囲は [1,1000]，予算制約は $\alpha = 0.4$ とした．実験回数は，表 4 のときと同じ，与えられた枝密度に従って 5 種類のグラフ，各グラフに対して時間–費用の割当てを 10 通り生成し，50 回の実験を行った平均値を示している．

表 5 において，func 列は時間–費用間の関数関係を示し，凸 (b)（凹 (b)）では底の値を括弧の中に示している．また

新たに trun という評価項目を付加しているが，これは 50 問のうち，1,200 秒以内に解けず，実行を打ち切った問題数である．打ち切られた問題は，平均値を出すために用いられていないため，trun が 0 でない問題については，もし解が求められるまで実行すれば，#subp や cpu は表に示された数値よりも大きくなる．終わりの 8 列は，最適解として使われた枝のモード番号の割合（%）を示しており，モード番号 1 は最安–最遅，モード番号 8 は最高–最速である．

最初の行は，表 1 でも用いた一様乱数型で，仮定式 (1) を満足しているだけのものであり，一番高速に解けている．最適解として採られている枝のモードには偏りがあり，必要であれば予算を使い，可能な限り時間の短いモードを選ぼうとしていることが分かる．そして func の違いは，特徴的な結果が得られた．

凸関数では上下界値の gap が小さくなり，最適解に採用されているモードは中央寄りの傾向が見られる．つまりモード番号 1，2 というのは，少し予算をかけることでより時間の短いモードから選ぼうとし，モード番号 7，8 というのは，これ以上予算をつぎ込んでも時間短縮にはつながらないので選ぼうとしない．この傾向は凸の膨らみ具合が大きくなるほど顕著である．しかし，中央部ではモード間の差異が小さく，一様乱数型と比べると，計算時間を要する．

凹関数では gap はさらに大きくなり，最適解に採用されているモードは凸型とは正反対で，両極端のモードに集中している．この傾向も両極端部の関数の傾きを見れば説明ができ，予算をかけるべき枝と，そうでない枝とが二分されることが観察できる．この二分性のために計算時間もより多く必要となる．

最後に線形関数は，gap が他と際立って大きい．釘付けテストはある程度機能しているが，計算時間の増大および 1,200 秒で打ち切らざるをえなかった問題数も目立っており，解くことが困難な問題群である．これは線形関数の場合，モードの違いによる費用対効果に差がないからであろう．ただし，最適解として採られた枝のモード番号は，凹関数の場合に似ており，最安–最遅のモードあるいは最高–最速のモードのいずれかに偏っている．これは線形関数での対費用効果は，ある枝のモード間では一定であっても，枝の間では対費用効果の高い枝・低い枝という差があるため，対費用効果の高い枝に対して予算を投入するからであろう．ただし，凹関数のときほど二極化はなく，何本かの枝が中間のモード番号を選んでいる．この状況が最適解の特定を組合せ的に困難にしているものと思われる．

7. まとめ

本論文ではグラフの各枝に何種類かの時間–費用がトレードオフの関係になっているモードが与えられ，いずれかを選びながら，限られた予算以内で総時間が最小の全域木を求める問題を提案した．時間–費用がトレードオフの関係にある問題は，プロジェクトスケジューリングの分野では古くから見られる状況設定ではあるが，ネットワーク計画法において同様の設定を導入した問題はほとんどみあたらない．

本問題に対し，ラグランジュ緩和による下界値と局所探索法による上界値を求めた．この上下界値のギャップは精度が良く，釘付けテストは十分効果的に機能しており，続く分枝限定法によって決めなければならない未固定の枝やモードはごくわずかな数に絞り込むことに成功した．

計算機実験では，先行研究として KY（Kataoka-Yamada [13]）を比較対象にした．KY は 2 モードの問題に特化しており，2 重グラフに変換してから扱っている．一方，本研究では多モードの場合でも多重グラフに変換などはせず，そのままのグラフで扱っている．このため，6.1 節では KY とまったく同じ例題・計算環境において勝った結果を得た．また，多モードモデルでは，時間–費用の間に凸関数，凹関数あるいは線形関数などの関係を持たせ，より詳細に問題およびアルゴリズムの特性を見ることができる．計算機実験の結果では，凸関数ではモード番号が中央部に，凹関数ではモード番号が両極端になることが観察された．特に対費用効果に差がない線形関数は解くことが難しいことが分かった．

モード数が十分多い場合の線形関数は，最安・最遅あるいは最高・最速という両極端のモードの自由なペアを採ることができる連続型問題に発展させることができる．プロジェクトスケジューリングの場合は，連続型は簡単に解ける問題であることが多いが [20], [22]，最小全域木の場合は，プロジェクトスケジューリングとは，本質的に異なる．このことについては，あらためて研究の必要性がある [14]．

参考文献

[1] Aggarwal, V., Aneja, Y.P. and Nair, K.P.K.: Minimal Spanning Tree Subject to a Side Constraint, *Computers & Operations Research*, Vol.9, pp.287–296 (1982).

[2] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B.: *Network Flows – Theory, Algorithms, and Applications*, Prentice Hall, Englewood Criffs (1993).

[3] Ball, M.O., Magnanti, T.L., Monma, C.L. and Nemhauser, G.L. (Eds.): *Network Models*, Handbooks in Operations Research and Management Science, Vol.7, North-Holland (1995).

[4] De, P., Dunne, E.J., Ghoshi, J.B. and Wells, C.E.: Complexity of Discrete Time-Cost Tradeoff Problem for Project Networks, *Operations Research*, Vol.45, pp.302–306 (1997).

[5] Değirmenci, G. and Azizoğlu, M.: Branch and Bound Based Solution Algorithms for the Budget Constrained Discrete Time/Cost Trade-off Problem, *Journal of the Opeerational Research Society*, Vol.64, pp.1477–1484 (2013).

[6] Deineko, V.G. and Woeginger, G.J.: Hardness of Approximation of the Discrete Time/Cost Trade-off Problem, *Operations Research Letters*, Vol.29, pp.207–210

(2001).
[7] Elmaghraby, S.E.: Resource Allocation via Dynamic Programming in Activity Networks, *European Journal of Operational Research*, Vol.64, pp.199–215 (1993).
[8] Guignard, M., Rosenwein, M.: An Application of Lagrangean Decomposition to the Resource-Constrained Minimum Weighted Arborescence Problem, *Networks*, Vol.20, pp.345–359 (1990).
[9] Hafizoğlu, A.B. and Azizoğlu, M.: Liniear Programming Based Approaches for the Discrete Time/Cost Trade-off Problem in Project Networks, *Journal of the Operational Research Society*, Vol.61, pp.676–685 (2010).
[10] Hartmann, S. and Briskorn, D.: A Survey of Variatinos and Extensions of the Resource Constrained Project Scheduling Problem, *Duropean Journal of Operational Research*, Vol.207, pp.1–14 (2010).
[11] Hazir, O., Haouari, M. and Erel, E.: A Discrete Time/Cost Trade-off Problem – a Decomposition-based Solution Algorithm for the Budget Version, *Computers and Operations Research*, Vol.37, pp.649–655 (2010).
[12] Herroelem, W., De Reyck, B. and Demeulemeester, E.: Resource Constrained Project Scheduling – A Survey of Recent Developments, *Computers and Operations Research*, Vol.25, pp.279–302 (1998).
[13] Kataoka, S. and Yamada, T.: Algorithms for the Minimum Spanning Tree Problem with Resource Allocation, *Operations Research Perspectives*, Vol.3, pp.5–13 (2016).
[14] 片岡靖詞，村崎暢彦：連続型・時間/費用トレードオフ最小全域木問題，情報処理学会論文誌，Vol.57, No.10, pp.2260–2271 (2016).
[15] Kellerer, H., Pferschy, U. and Pisinger, D.: *Knapsack Problems*, Springer, Berlin (2004).
[16] Martello, S. and Toth, P.: *Knapsack Problems – Algorithms and Computer Implementations*, John Wiley & Sons, Chichester (1990).
[17] Li, Y., Zou, C.Y., Zhang, S. and Vai, M.I.: Research on Multi-Objective Minimum Spanning Tree Algorithm Based on Ant Algorithm, *Research Journal of Applied Sciences, Engineering and Technology*, Vol.5, pp.5051–5056 (2013).
[18] Pisinger, D.: An Exact Algorithm for Large Multiple Knapsack Problems, *European Journal of Operational Research*, Vol.114, pp.528–541 (1999).
[19] Robinson, D.R.: A Dynamic Programming Solution to Cost-Time Trade-off for CPM, *Management Science*, Vol22, pp.158–166 (1975).
[20] 関根智明：PERT/CPM，日科技連 (1973).
[21] Skutella, M.: Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem, *Mathematics of Operations Research*, Vol.23, pp.909–929 (1998).
[22] 刀根 薫：PERT 講座 I 基礎編，東洋経済新報社 (1966).
[23] Vanhoucke, M. and Debels, D.: The Discrete Time/Cost Trade-off Problem – Extensions and Heuristic Procedures, *Journal of Scheduling*, Vol.10, pp.311–326 (2007).
[24] Yamada, T., Watanabe, K. and Kataoka, S.: Algorithms to Solve the Knapsack Constrained Maximum Spanning Tree Problem, *International Journal of Computer Mathematics*, Vol.82, pp.23–34 (2005).
[25] Wo, B.Y. and Chao, K.M.: *Spanning Trees and Optimization Problems*, Chapman & Hall/CRC (2004).

片岡 靖詞 （正会員）

防衛大学校情報工学科准教授．1985年早稲田大学理工学部工業経営学科卒業，1987年同大学大学院修士課程修了，1990年同大学院博士課程満期退学．1990年防衛大学校情報工学科．1993年博士（工学）．各種の組合せ最適化アルゴリズム開発に従事．日本オペレーションズ・リサーチ学会，INFORMS 各会員．

村崎 暢彦

陸上自衛隊．2010年3月防衛大学校情報工学科卒業，同年陸上自衛隊入隊，幹部候補生学校入校．2011年3月～2013年3月第11通信中隊勤務．2013年4月～2015年3月防衛大学校理工学研究科前期課程．2015年3月～9月技術研究本部勤務．2015年10月～現在，防衛装備庁勤務．