Open access • Journal Article • DOI:10.1080/00207160412331290667

Algorithms to solve the knapsack constrained maximum spanning tree problem

Takeo Yamada, Kohtaro Watanabe, Seiji Kataoka

Institutions: National Defense Academy of Japan

Published on: 01 Jan 2005 - International Journal of Computer Mathematics (Taylor & Francis)

Topics: Continuous knapsack problem, Knapsack problem, Minimum spanning tree, Spanning tree and Cutting stock problem

International Journal of Computer Mathematics
Vol. 82, No. 1, January 2005, 23–34

Taylor & Francis
Taylor & Francis Group

Algorithms to solve the knapsack constrained maximum spanning tree problem

TAKEO YAMADA*, KOHTARO WATANABE and SEIJI KATAOKA

Department of Computer Science, The National Defense Academy Yokosuka, Kanagawa 239-8686, Japan

(Received 19 April 2004)

The knapsack problem and the minimum spanning tree problem are both fundamental in operations research and computer science. We are concerned with a combination of these two problems. That is, we are given a knapsack of a fixed capacity, as well as an undirected graph where each edge is associated with profit and weight. The problem is to fill the knapsack with a feasible spanning tree such that the tree profit is maximized. We prove this problem $\mathcal{NP}$-hard, present upper and lower bounds, develop a branch-and-bound algorithm to solve the problem to optimality and propose a shooting method to accelerate computation. We evaluate the developed algorithm through a series of numerical experiments for various types of test problems.

Keywords: Minimum spanning tree problem; Knapsack problem; Combinatorial optimization

C.R. Categories: G.2.1; G.1.6

1. Introduction

Let us consider an undirected graph [1, 2] $G = (V, E)$, where V is a finite set of vertices and $E \subseteq V \times V$ is the set of edges. Here, each edge is associated with two kinds of numbers, namely profit p: $E \to Z_+$ and weight w: $E \to Z_+$. We also have a knapsack of integer capacity $c > 0$. We assume G to be connected and simple in the sense that there exist neither self-loops nor parallel edges. For a spanning tree T in G, we define its profit and weight as the sum of the profits and weights of the constituent edges, and these are denoted as $p(T)$ and $w(T)$, respectively. A spanning tree T is said to be *feasible* if it satisfies $w(T) \leq c$.

Our problem is to find a spanning tree that maximizes the profit among all the feasible spanning trees. Mathematically, this is formulated as follows.

KCMST:

$$\text{maximize} \quad p(T) \tag{1}$$

$$\text{subject to} \quad w(T) \leq c, \tag{2}$$

$$T \text{ is a spanning tree.} \tag{3}$$

* Corresponding author. Fax: +81-46-844-5911; Email: yamada@nda.ac.jp

International Journal of Computer Mathematics
ISSN 0020-7160 print/ISSN 1029-0265 online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207160412331290667

We call this the *knapsack constrained maximum spanning tree problem*, or KCMST for short, since this is a combination of the *maximum spanning tree problem* and the *knapsack problem*, both of which are fundamental in operations research and computer science. Indeed, for $c = \infty$ KCMST is the maximum spanning tree problem, which can be easily solved by the algorithms of Kruskal [3] or Prim [4]. On the other hand, if constraint (3) is dropped the problem reduces to the 0–1 knapsack problem [5, 6].

Throughout the paper we denote $n := |V|$ and $m := |E|$, and p^* is the optimal objective value to KCMST. First, we prove $\mathcal{NP}$-hardness [7] of the problem.

THEOREM 1 *KCMST is $\mathcal{NP}$-hard.*

Proof Corresponding to the knapsack problem KP:

$$\text{maximize} \quad \sum p_i x_i \tag{4}$$

$$\text{subject to} \quad \sum w_i x_i \leq c, \tag{5}$$

$$x_i \in \{0, 1\}, \tag{6}$$

we construct a graph consisting of the vertex set $V = \{0, 1, 1', 2, 2', \ldots, n, n'\}$ and the edge set $E = \{(0, 1), (0, 1'), (1, 1'), \ldots, (0, n), (0, n'), (n, n')\}$. The profits and weights are defined as $p(0, i) = p_i$, $w(0, i) = w_i$, $p(0, i') = w(0, i') = 0$, $p(i, i') = M$, $w(i, i) = 0$ for $i = 1, \ldots, n$ (figure 1), where M is a sufficiently large number. We note that an optimal solution to KCMST for this graph necessarily includes edges $\{(i, i')|\ i = 1, \ldots, n\}$. Also, an arbitrary feasible spanning tree including all these edges naturally corresponds, in an one-to-one way, to a feasible solution to KP. Thus KP is solved by solving KCMST, which completes the proof since KP is itself $\mathcal{NP}$-hard. ■

Although literature abounds on the maximum (minimum) spanning tree problem [1, 8] and the knapsack problem [5, 6], as well as on their variations [9–11], few previous researches have been published on KCMST. Yamamoto and Kubo [12] formulated the problem, discussed the Lagrangean relaxation, but neither proved $\mathcal{NP}$-hardness nor gave solution algorithms. In this paper, we develop algorithms to solve the problem approximately as well as to optimality.

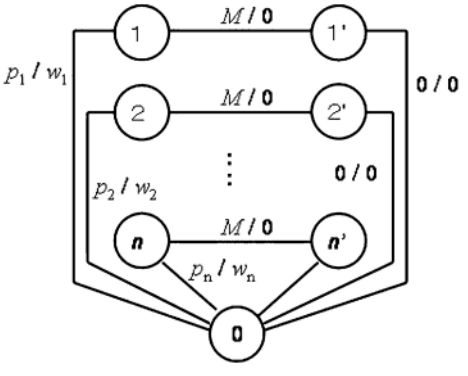


Figure 1. Proof of $\mathcal{NP}$-hardness.

2. Lagrangean relaxation

The *Lagrangean relaxation* [13, 14] to *KCMST* is given as follows.

Lagrange:

$$\text{maximize} \quad L(\lambda, T) := p(T) + \lambda(c - w(T)) \tag{7}$$

$$\text{subject to} \quad T \text{ is a spanning tree.} \tag{8}$$

For a fixed λ, this is simply a maximum spanning tree problem which is easily solved. Let T_λ denote an optimal solution to this problem with the objective value $L(\lambda)$. Then, the following proposition holds.

PROPOSITION 1

(i) *For an arbitrary* $\lambda \geq 0$, $L(\lambda)$ *gives an upper bound to KCMST, i.e.* $L(\lambda) \geq p^*$.
(ii) $L(\lambda)$ *is piecewise linear and convex in* $[0, \infty)$.
(iii) *If* $L(\lambda)$ *is differentiable at* λ,

$$\frac{\mathrm{d}L(\lambda)}{\mathrm{d}\lambda} = c - w(T_\lambda). \tag{9}$$

Proof (i) is simply a basic principle of the Lagrangean relaxation. For a fixed spanning tree T, $L(\lambda, T)$ is linear with respect to λ, and thus $L(\lambda)$ is obtained by taking the upper envelope of the linear functions corresponding to all the spanning trees (figure 2). Hence, $L(\lambda)$ is piecewise linear and convex, and when differentiable its derivative is given by equation (9). ■

Let $L(\lambda)$ be minimum at $\lambda^\flat \geq 0$. From (ii) and (iii) of Proposition 1, we note that T_λ is feasible if $\lambda > \lambda^\flat$, and infeasible if $\lambda < \lambda^\flat$. Note that $\lambda^\flat < \infty$ if and only if $L(\lambda)$ is bounded from below. In this case, let $T^\flat := T_{\lambda^\flat + 0}$. This gives a feasible solution, which we call the *Lagrangean solution*, and the corresponding *Lagrangean lower bound* is denoted as $\underline{p}^\flat := p(T^\flat)$.

The Lagrangean function also satisfies the following proposition.

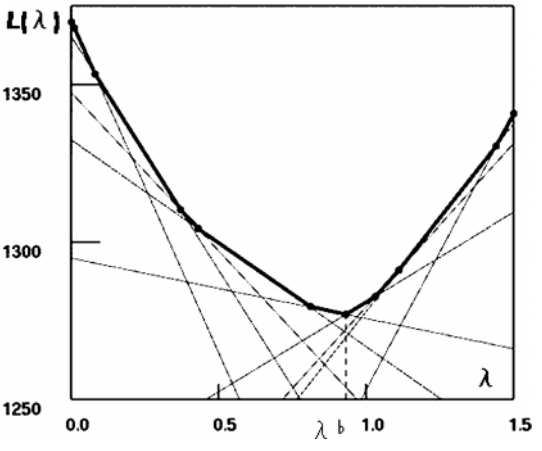


Figure 2. Lagrangean function $L(\lambda)$.

PROPOSITION 2

(i) *If* $\lambda^{\flat} = \infty$, *KCMST is infeasible.*
(ii) *If* T_0 *is feasible* (*i.e.* $w(T_0) \leq c$), *then* $\lambda^{\flat} = 0$ *and* T_0 *is optimal to KCMST.*
(iii) *If* $w(T^{\flat}) = c$, $T^{\flat}$ *is optimal to KCMST.*

Proof First, if $\lambda^{\flat} = \infty$ we have $\underline{p}^{\flat} = -\infty$ and thus $c < w(T)$ for all spanning trees T, which proves (i). Next, if T_0 is feasible it is obtained in KCMST by ignoring the knapsack constraint (2). Then, (ii) is straightforward from Proposition 1. In addition, if $w(T^{\flat}) = c$ holds, the upper and lower bounds coincide by $\bar{p} = L(\lambda^{\flat}) = p(T^{\flat}) = \underline{p}^{\flat}$. Thus, (iii) is proved. ■

The optimal $\lambda^{\flat}$ can be obtained by the following *bisection method*.

ALGORITHM BISECTION

Step 0: Start with $\lambda_L := 0$ *and* λ_H *such that* $\mathrm{d}L(\lambda_H)/\mathrm{d}\lambda \geq 0$.
Step 1: Set $\lambda := (\lambda_L + \lambda_H)/2$, *and find* T_λ.
Step 2: If $w(T_\lambda) \leq c$ *set* $\lambda_H := \lambda$; *otherwise set* $\lambda_L := \lambda$.
Step 3: If $|\lambda_H - \lambda_L| < \varepsilon$ *output* $\lambda^{\flat} := \lambda_H$ *and stop*; *otherwise go back to Step 1.*

Here $\varepsilon > 0$ is an arbitrary *tolerance level* for $\lambda^{\flat}$, which we take as $\varepsilon = 0.01$. We start with an appropriate interval between λ_L and λ_H where $L(\lambda)$ is negatively and positively sloped, respectively (Step 0). We take the midpoint and solve the Lagrangean relaxation problem to find the corresponding tree T_λ (Step 1). We update either the lower or upper bound depending on whether $w(T_\lambda) \leq c$ holds or not (Step 2). This is repeated until the interval is sufficiently small (Step 3).

Thus we obtain the 'best' upper bound $\bar{p} := \lfloor L(\lambda^{\flat}) \rfloor$.

Example 1: Figure 3 shows a plane graph with $n = 20$ and $m = 46$, where edges are numbered in italic. The edge weights and profits are shown in table 1; these are uniformly and

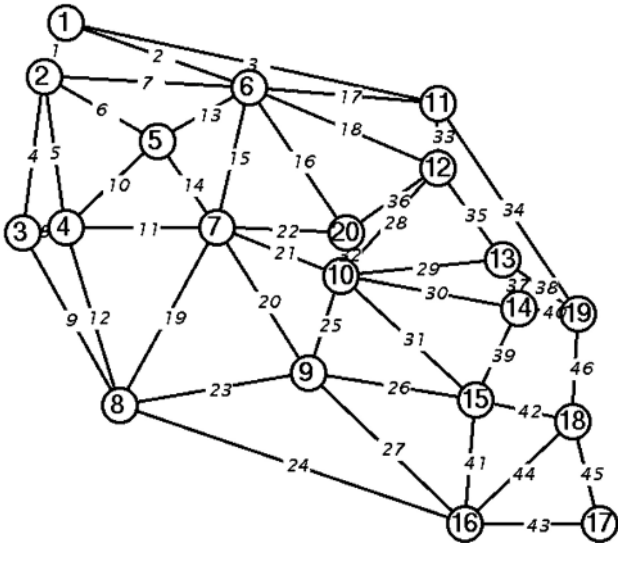


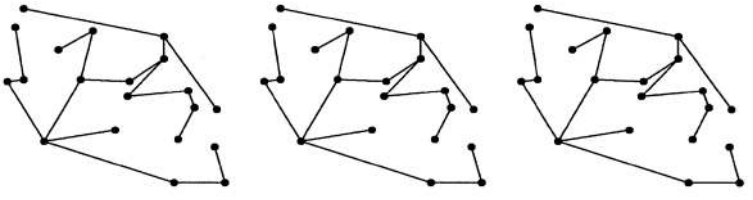
Figure 3. Graph for Example 1.

Table 1. Data for the graph of figure 3.

Edge	Root	Top	Profit	Weight	Edge	Root	Top	Profit	Weight
e_1	v_1	v_2	27	16	e_{24}	v_8	v_{16}	23	16
e_2	v_1	v_6	43	77	e_{25}	v_{10}	v_9	48	73
e_3	v_1	v_{11}	79	66	e_{26}	v_{15}	v_9	46	37
e_4	v_3	v_2	70	46	e_{27}	v_9	v_{16}	3	22
e_5	v_4	v_2	96	37	e_{28}	v_{12}	v_{10}	61	4
e_6	v_2	v_5	26	80	e_{29}	v_{13}	v_{10}	54	35
e_7	v_6	v_2	7	73	e_{30}	v_{10}	v_{14}	94	84
e_8	v_3	v_4	80	55	e_{31}	v_{10}	v_{15}	34	37
e_9	v_8	v_3	65	5	e_{32}	v_{20}	v_{10}	60	65
e_{10}	v_5	v_4	26	24	e_{33}	v_{11}	v_{12}	72	14
e_{11}	v_7	v_4	51	81	e_{34}	v_{11}	v_{19}	89	68
e_{12}	v_4	v_8	49	50	e_{35}	v_{12}	v_{13}	65	55
e_{13}	v_5	v_6	24	19	e_{36}	v_{12}	v_{20}	95	63
e_{14}	v_5	v_7	6	42	e_{37}	v_{14}	v_{13}	63	14
e_{15}	v_6	v_7	38	5	e_{38}	v_{13}	v_{19}	12	55
e_{16}	v_6	v_{11}	32	48	e_{39}	v_{14}	v_{15}	62	27
e_{17}	v_6	v_{12}	43	65	e_{40}	v_{19}	v_{14}	1	93
e_{18}	v_6	v_{20}	49	35	e_{41}	v_{15}	v_{16}	44	76
e_{19}	v_8	v_7	84	3	e_{42}	v_{15}	v_{18}	52	97
e_{20}	v_9	v_7	48	79	e_{43}	v_{16}	v_{17}	74	67
e_{21}	v_7	v_{10}	8	3	e_{44}	v_{18}	v_{16}	37	48
e_{22}	v_7	v_{20}	77	1	e_{45}	v_{17}	v_{18}	23	24
e_{23}	v_9	v_8	62	17	e_{46}	v_{19}	v_{18}	94	100

Table 2. Behavior of bisection.

Iteration	λ_L	λ_H	λ	$L(\lambda)$	$dL(\lambda)/d\lambda$
1	0.00	10.00	5.00	2091.00	220.00
2	0.00	5.00	2.50	1541.00	220.00
3	0.00	2.50	1.25	1307.75	119.00
4	0.00	1.25	0.63	1291.38	−65.00
5	0.63	1.25	0.94	1277.25	60.00
6	0.63	0.94	0.78	1281.22	−65.00
7	0.78	0.94	0.86	1278.25	−16.00
8	0.86	0.94	0.90	1277.63	−16.00
9	0.90	0.94	0.92	1277.31	−16.00
10	0.92	0.94	0.93	1277.16	−16.00



(a) $T^\flat$ (b) $T^\sharp$ (c) $T^\star$

Figure 4. Heuristic and exact solutions to the instance of Example 1. (a) Lagrangean solution $T^\flat$, (b) 2-opt solution $T^\sharp$ and (c) optimal solution T^*.

independently distributed over [1, 100], and the knapsack capacity is set to $c = 600$. Figure 2 is actually $L(\lambda)$ for this example. Table 2 shows the behavior of Bisection. From this we obtain $\lambda^\flat = 0.94$ with $\bar{p} = 1277$. The corresponding $T^\flat$ is shown in Figure 4(a) with $\underline{p}^\flat = 1221$ and $w(T^\flat) = 540$.

3. A heuristic algorithm

In section 2, we have already stated how a feasible solution $T^\flat$, and thus a lower bound $\underline{p}^\flat$, can be found. This approximate solution may further be improved by applying any of the *heuristic algorithms* [15, 16], which is now standard in combinatorial optimization. Here we mention the *2-opt local-search method*, which starts with $T^\flat$ as an initial solution and continues to improve the solution as far as possible.

For an arbitrary spanning tree, an edge included (not included, respectively) in that tree is referred to as a *tree* (*co-tree*, respectively) *edge*. Adding an arbitrary co-tree edge to the tree induces a cycle in that graph. Then, by removing a tree-edge from that cycle we obtain another spanning tree. We say that these two spanning trees are *neighbors* of each other (figure 5). For a feasible spanning tree T we define its *neighborhood* $N(T)$ by

$$N(T) := \{T' | T' \text{ is a feasible neighbor of } T\} \tag{10}$$

The local-search algorithm is as follows.

ALGORITHM LOCAL_SEARCH

Step 0: Start with $T := T^\flat$.
Step 1: If there exists $T' \in N(T)$ *such that* $p(T') > p(T)$ *go to Step 2*; *otherwise output* T *and stop.*
Step 2: Set $T := T'$ *and go to Step 1.*

Let $T^\sharp$ be the output of this algorithm. This is referred to as the *2-opt solution*, and the corresponding *2-opt lower bound* is denoted as $\underline{p}^\sharp := p(T^\sharp)$.

Example 2: For $T^\flat$ of Example 1, Local_Search improves solution by exchanging, step-by-step, tree and co-tree edges as $e_{13} \leftrightarrow e_{10}$, $e_{15} \leftrightarrow e_{18}$ and $e_{29} \leftrightarrow e_{35}$. The objective value increases as $1221 \rightarrow 1223 \rightarrow 1234 \rightarrow 1245$. The output $T^\sharp$ is shown in Figure 4(b), where we have $\underline{p}^\sharp = 1245$ with $w(T^\sharp) = 595$.

4. A branch-and-bound algorithm

Now we can construct a *branch-and-bound* (henceforth B&B) algorithm using the heuristic solution as well as the upper and lower bounds discussed in the previous sections.

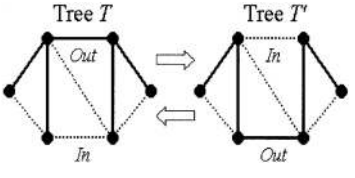


Figure 5. Neighboring trees.

Let T be an arbitrary feasible spanning tree. Initially this may be taken as $T := T^\sharp$, and explicitly written as $T = \{e^1, e^2, \ldots, e^{n-1}\}$. From this we generate a series of *subproblems* in the following way. The ith subproblem is to find an optimal solution that includes $\{e^1, \ldots, e^{i-1}\}$, but does not include e^i $(i = 1, \ldots, n-1)$. Clearly the KCMST is solved if we have solved all these subproblems, and to accomplish this we repeat the same thing recursively with respect to the subproblems. Such a decomposition of a combinatorial optimization problem into a set of subproblems can be seen in the literature [17, 18].

We consider a more general situation where we have a pair of disjoint edge sets F and R. A spanning tree T is said to be (F, R)-*admissible* if it includes all the edges of F, but does not include those of R. These are referred to as *fixed* and *restricted* edges, respectively. Then, the subproblem $P(F, R)$ is to find an optimal solution that is feasible as well as (F, R)-admissible. Note that the upper and lower bounds introduced in the previous sections are easily modified to account for (F, R)-admissibility. Let these be denoted as $\underline{p}^\sharp(F, R)$ and $\bar{p}(F, R)$, respectively, and $T^\sharp(F, R)$ is the corresponding 2-opt solution. In addition, $T^\dagger$ and $p^\dagger$ represent the *incumbent* solution and the corresponding objective value, respectively. Initially these are $T^\dagger := \emptyset$ and $p^\dagger := -\infty$.

The following algorithm solves $P(F, R)$.

ALGORITHM B&B(F, R)

Step 1: If $P(F, R)$ is infeasible or $\bar{p}(F, R) \leq p^\dagger$, terminate $P(F, R)$.

Step 2: If $P(F, R)$ is solved to optimality, update, if necessary, the incumbent solution (and $p^\dagger$) and terminate.

Step 3: Otherwise, find an (F, R)-admissible $T^\sharp(F, R)$ by the heuristic method of section 3. *If necessary, update the incumbent.*

Step 4: Use $T^\sharp(F, R)$ to divide the problem into a set of mutually disjoint sub-subproblems, and apply B&B recursively to these sub-subproblems.

In this algorithm, if 2-opt solution is written as $T^\sharp(F, R) = F \cup \{e^{k+1}, \ldots, e^{n-1}\}$ with $k := |F|$, the sub-subproblems are generated in Step 4 as $P(F_i, R_i)$, where $F_i := F \cup \{e^{k+1}, \ldots, e^{i-1}\}$ and $R_i := R \cup \{e^i\}$ for $i = k+1, \ldots, n-1$.

By calling B&B $(\emptyset, \emptyset)$, KCMST is solved. We note that in this algorithm subproblems are examined in *depth-first* fashion [19, 20], due to the nature of recursive calls.

Example 3: For the problem of Example 1, we start with $\underline{p}^\sharp = 1245$ and $\bar{p} = 1277$. B&B finds $p^* = 1263$ after generating 303 subproblems. The optimal spanning tree is shown in figure 4(c), where we have $w(T^*) = 594$.

5. Shooting method

In the standard B&B method, the initial value for the incumbent value is usually set to $p^\dagger := -\infty$. In the shooting method we make a *guess* for this value, and run B&B using this value as if it is actually an initial incumbent value, *i.e.*, $p^\dagger \leq p^*$. We may fail to obtain an optimal solution, since p^* is unknown and therefore the guess may not be correct. If this happens, we simply repeat B&B with the guess lowered.

More precisely, let B&B $[p]$ denote the algorithm B&B $(\emptyset, \emptyset)$ run with $p^\dagger := p \in [\underline{p}, \bar{p}]$ used as the initial incumbent value, and $\phi(p)$ be the number of subproblems generated by B&B $[p]$. For the lower bound we may use either $\underline{p}^\flat$ or $\underline{p}^\sharp$, but in the sequel we denote this simply as $\underline{p}$. Then, we have the following.

PROPOSITION 3

(i) *If $p \leq p^*$, B&B [p] correctly solves KCMST.*
(ii) *Otherwise, if $p > p^*$ B&B [p] fails, i.e., it terminates without finding the solution to KCMST.*
(iii) *$\phi(p)$ is a non-increasing function of p in $[\underline{p}, \bar{p}]$.*

Proof These are straightforward from the behavior of the B&B method. ■

Then, the shooting method is as follows.

ALGORITHM SHOOTING_METHOD

Step 1: Start with the interval $[\underline{p}, \bar{p}]$.
Step 2: Set $p^\dagger := \alpha \underline{p} + (1-\alpha)\bar{p}$.
Step 3: Run B&B $[p^\dagger]$. If optimal solution is found, output it and stop.
Step 4: Otherwise, update $\bar{p} := p^\dagger$ and go back to Step 2.

Here α is a constant in [0, 1]. After some preliminary numerical tests, we fix this value as $\alpha = 0.3$. At each iteration of Shooting_Method the interval $[\underline{p}, \bar{p}]$ shrinks by the factor of $1-\alpha$, and as soon as $p^\dagger \leq p^*$ is satisfied KCMST is solved to optimality.

Example 4: For the problem of Example 1, we first try $p^\dagger = (\lfloor 0.3 \cdot 1245 + 0.7 \cdot 1277 \rfloor =) 1267$. B&B [1267] fails after generating 55 subproblems, since $1267 > p^*$. We try again with $p^\dagger = (\lfloor 0.3 \cdot 1245 + 0.7 \cdot 1267 \rfloor =) 1260$, and after examining 108 more subproblems obtain the same optimal solution as in Example 3.

6. Numerical experiments

We have implemented the B&B algorithm in ANSI C language and conducted a series of numerical tests on an IBM RS/6000 44P Model 270 workstation. The algorithm with/without shooting is henceforth denoted as B&B/Shooting and B&B/No-shooting, respectively. The test problems are *plane graph* $P_{n,m}$ and *complete graph* K_n. Figure 3 is an example of $P_{20,46}$. For edge weights and profits we consider three cases.

(a) *Uncorrelated*: $p(e)$ and $w(e)$ $(e \in E)$ are independently and uniformly distributed over [1, 100].
(b) *Weakly correlated*: $w(e)$ are similarly distributed over [1, 100], and $p(e) := \lfloor 0.8w(e) + v(e) \rfloor$, where $v(e)$ is uniformly random over [1, 20].
(c) *Strongly correlated*: $w(e)$ are distributed similarly, and $p(e) := \lfloor 0.9w(e) + 10 \rfloor$.

Knapsack capacities are set to $c = 35n$ for plane graphs, and $c = 20n - 20$ for complete graphs.

Table 3 shows the result of experiments of B&B/Shooting for the uncorrelated case, where each row is the average of 10 randomly generated instances. Here, $\mathrm{Err}_{\mathrm{L}}^{\flat}$ represents the *relative error* of the Lagrangean lower bound against the optimal value, *i.e.*,

$$\mathrm{Err}_{\mathrm{L}}^{\flat} := \frac{100(p^* - \underline{p}^{\flat})}{p^*} \quad (\%). \tag{11}$$

Table 3. Result of experiments: uncorrelated case.

Problem	c	$\mathrm{Err}_{\mathrm{L}}^{\flat}$	$\mathrm{Err}_{\mathrm{L}}^{\sharp}$	$\mathrm{Err}_{\mathrm{U}}$	#Rep	p^*	#Sub	Sec.
$P_{50,127}$	1,750	0.8239	0.3361	0.1165	2.5	3,689.8	320.4	0.43
$P_{100,260}$	3,500	0.5458	0.2307	0.0376	2.1	7,456.4	889.5	1.78
$P_{200,560}$	7,000	0.4046	0.1822	0.0054	1.2	14,928.9	1,492.9	5.46
$P_{400,1120}$	14,000	0.1262	0.0684	0.0020	1.0	30,111.9	3,051.4	24.44
$P_{600,1680}$	21,000	0.1195	0.0525	0.0016	1.2	44,926.1	5,294.8	75.25
$P_{800,2240}$	28,000	0.2947	0.1241	0.0007	1.0	59,793.4	21,721.3	466.37
$P_{1000,2800}$	35,000	0.1654	0.0729	0.0004	1.0	74,799.2	20,351.1	592.77
K_{40}	780	0.2205	0.0762	0.0299	1.3	3,673.3	187.9	0.87
K_{60}	1,180	0.3799	0.0985	0.0088	1.0	5,686.3	179.1	1.89
K_{80}	1,580	0.2668	0.1250	0.0052	1.0	7,682.7	352.6	6.54
K_{100}	1,980	0.1476	0.0423	0.0010	1.0	9,686.5	376.7	12.48
K_{120}	2,380	0.1205	0.0410	0.0017	1.0	11,701.9	468.5	23.69
K_{140}	2,780	0.0561	0.0226	0.0000	1.0	13,717.3	832.4	60.95
K_{160}	3,180	0.0897	0.0388	0.0000	1.0	15,714.3	9800.1	476.26
K_{180}	3,580	0.1004	0.0446	0.0006	1.0	17,724.2	5152.7	636.54
K_{200}	3,980	0.0400	0.0167	0.0005	1.0	19,733.1	2356.8	375.26

$\mathrm{Err}_{\mathrm{L}}^{\sharp}$ is analogously defined for 2-opt lower bound $\underline{p}^{\sharp}$, and

$$\mathrm{Err}_{\mathrm{U}} := \frac{100(\bar{p} - p^*)}{p^*} \quad (\%). \tag{12}$$

In addition, #Rep is the number of shooting trials repeated, p^* is the optimal objective value, #Sub is the total number of subproblems generated and Sec. is the CPU time in s.

Tables 4 and 5 are the results of B&B/Shooting for the weakly and strongly correlated cases, respectively. Superscribed in the column of Sec. are the number of subproblems solved within 2000 CPU seconds. For example, we were able to solve 8 instances out of 10 for $P_{600,1680}$ with weakly correlated edges. In these cases, the average was taken over the instances solved within the time limit. Superscripts are omitted if all 10 instances were solved.

We were able to solve KCMST for graphs with up to 1000 vertices in reasonable CPU time, but as in ordinary knapsack problems, the computation becomes harder as the profits and weights are more strongly correlated. From tables 3–5, we observe the following.

- Upper bound ($\bar{p}$) is very close to the optimal objective value (p^*) for almost all cases tested. The relative error $\mathrm{Err}_{\mathrm{U}}$ is usually less than 1%, and often much smaller.

Table 4. Result of experiments: weakly correlated case.

Problem	c	$\mathrm{Err}_{\mathrm{L}}^{\flat}$	$\mathrm{Err}_{\mathrm{L}}^{\sharp}$	$\mathrm{Err}_{\mathrm{U}}$	#Rep	p^*	#Sub	Sec.
$P_{50,127}$	1,750	4.1238	1.1609	0.0680	1.3	2,058.8	512.2	0.81
$P_{100,260}$	3,500	2.8292	0.5857	0.0145	1.1	4,131.9	1,156.8	2.71
$P_{200,560}$	7,000	1.0505	0.2632	0.0024	1.0	8,358.1	2,906.7	13.11
$P_{400,1120}$	14,000	0.8110	0.1824	0.0012	1.0	16,720.4	4,602.5	47.15
$P_{600,1680}$	21,000	0.8169	0.1670	0.0004	1.0	25,156.5	22,442.4	371.84^{8}
$P_{800,2240}$	28,000	0.4233	0.1031	0.0006	1.0	33,544.2	24,887.8	509.22^{5}
K_{20}	380	5.6257	0.7709	0.2132	1.4	609.7	114.7	0.25
K_{40}	780	4.0664	0.4950	0.0228	1.0	1,313.2	206.1	1.17
K_{60}	1,180	5.3884	0.5215	0.0050	1.0	2,013.6	475.2	6.09
K_{80}	1,580	4.7316	0.3388	0.0037	1.0	2,715.8	1,693.4	38.15
K_{100}	1,980	2.3565	0.1727	0.0000	1.0	3,416.1	5,174.2	377.61^{8}
K_{120}	2,380	3.6588	0.2062	0.0000	1.0	4,121.6	8,644.4	451.06^{8}

Table 5. Result of experiments: strongly correlated case.

Problem	c	$\mathrm{Err}_{\mathrm{L}}^{\flat}$	$\mathrm{Err}_{\mathrm{L}}^{\sharp}$	$\mathrm{Err}_{\mathrm{U}}$	#Rep	p^*	#Sub	Sec.U
$P_{50,127}$	1,750	9.3194	0.1559	0.0049	1.0	2052.7	4,903.9	2.84
$P_{100,260}$	3,500	16.5958	0.2649	0.0000	1.0	4114.9	34,7796.0	405.45[8]
K_{20}	380	18.0839	0.3401	0.0378	1.0	529.2	363.8	0.53
K_{30}	580	14.5540	0.3212	0.0000	1.0	809.4	36,970.2	99.63
K_{40}	780	8.6530	0.1193	0.0183	1.0	1089.8	50,636.5	226.30[6]

- Lower bounds ($\underline{p}^{\flat}$, $\underline{p}^{\sharp}$) are also within less than 1% of relative errors in the uncorrelated case. However, the Lagrangean solution ($\underline{p}^{\flat}$) deteriorates in the correlated cases. Even in such a case, 2-opt heuristics gives a solution of less than 1% error from the optimal.
- Usually, we obtain more accurate bounds as the size of problem (n) increases.

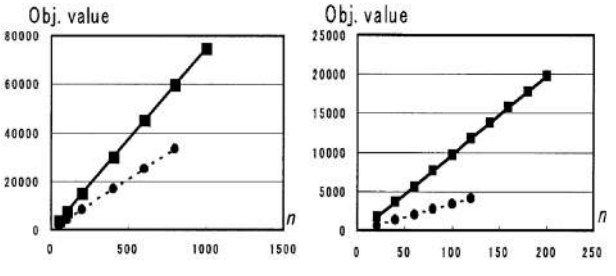
Figure 6 compares the optimal objective value as the function of n. Here figure 6(a, b), respectively) is the result for plane (complete, respectively) graphs, and the solid (broken, respectively) lines represent the uncorrelated (weakly correlated, respectively) case. From these we see the following.

- For the cases tested, the optimal objective value is almost linear with respect to n.
- Correlation between profits and weights makes the optimal value lower.

Figure 7 shows the number of subproblems generated, whereas figure 8 gives the CPU time in s. Again, (a) is for plane graphs and (b) is for complete graphs in these figures, and thick solid (broken, respectively) lines represent the results of B&B/Shooting for uncorrelated (weakly correlated, respectively) case, whereas the thin solid line is the results of B&B/No-shooting for uncorrelated case. In figures 7(b) and 8(b), values on the thin solid line need some caution for $n \geq 160$, since these are averages of only solved (less than 10) instances.

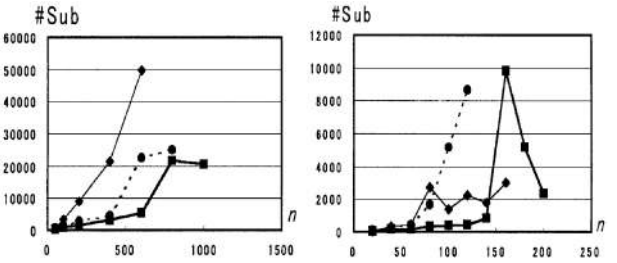
From these we conclude the following:

- Correlation between profit and weight usually makes the number of subproblems and CPU time larger, and thus makes problem harder to solve.
- For plane graphs, shooting makes the number of subproblems generated and CPU time smaller.
- This effect of shooting is not so clear for complete graphs. However, for $n \leq 140$ where the algorithms solved all 10 instances, B&B/Shooting is superior to B&B/No-shooting both in #Sub and CPU time.



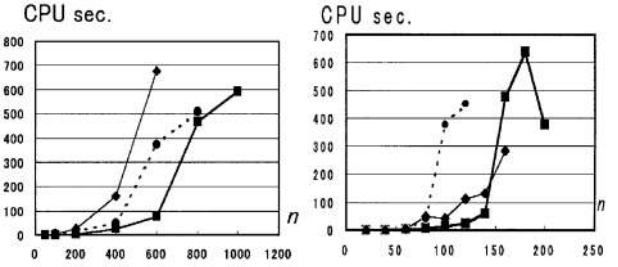
(a) Plane graphs (b) Complete graphs

Figure 6. Optimal objective values obtained by B&B/Shooting.



(a) Plane graphs (b) Complete graphs

Figure 7. Number of subproblems generated.



(a) Plane graphs (b) Complete graphs

Figure 8. CPU time in s.

7. Conclusion

We have formulated the problem KCMST, proved it is $\mathcal{NP}$-hard, derived upper and lower bounds, developed a B&B algorithm, proposed a shooting method and conducted some numerical experiments. As a result, we were able to solve KCMST with up to 1000 vertices in less than 2000 s. The shooting method was found to be effective, especially for plane graphs.

However, the B&B algorithm presented here is quite unsophisticated. To solve larger problems to optimality, more advanced algorithms are required. Investigation of the polyhedral structure [21] of the problem may lead to such an algorithm.

References

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B., 1993, *Network Flows: Theory, Algorithms, and Applications* (Englewood Cliffs, NJ: Prentice-Hall).
[2] Busacker, R.G. and Saaty, T.L., 1965, *Finite Graphs and Networks: An Introduction with Applications* (New York: McGraw-Hill).
[3] Kruskal, J.B., 1956, *Proceedings of the American Mathematical Society*, **7**, 8–50.
[4] Prim, R.C., 1957, *Bell System Technical Journal*, **36**, 1389–1401.
[5] Martello, S. and Toth, P., 1990, *Knapsack Problems: Algorithms and Computer Implementations* (Chichester: John Wiley & Sons).
[6] Salkin, U.M. and de Kluyver, C.A., 1975, *Naval Research Logistics*, **22**, 127–144.
[7] Garey, M.R. and Johnson, D.S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco: Freeman and Company).
[8] Graham, R.L. and Hell, P., 1985, *Annals of the History of Computing*, **7**, 43–57.
[9] Samphaiboon, N. and Yamada, T., 2000, *Journal of Optimization Theory and Applications*, **105**, 659–676.
[10] Yamada, T., Futakawa, M. and Kataoka, S., 1998, *European Journal of Operational Research*, **106**, 177–183.
[11] Yamada, T., Watanabe, K. and Kataoka, S., 2002, *Information Processing Society of Japan*, **43**, 2864–2870.

[12] Yamamoto, Y. and Kubo, M., 1997, *Invitation to the Traveling Salesman's Problem* (in Japanese) (Tokyo: Asakura).
[13] Nemhauser, G.L. and Wolsey, L.A., 1988, *Integer and Combinatorial Optimization* (New York: John Wiley & Sons).
[14] Wolsey, L.A., 1998, *Integer Programming* (New York: John Wiley & Sons).
[15] Aarts, E. and Lenstra, J.K. (Eds.), 1997, *Local Search in Combinatorial Optimization* (Chichester, UK: John Wiley & Sons).
[16] Osman, I.H. and Kelly, J.P. (Eds.), 1996, *Meta-heuristics: Theory and Applications* (Boston, MA: Kluwer).
[17] Hamacher, H.W. and Queyranne, M., 1985, *Annals of Operations Research*, **4**, 123–143.
[18] Lawler, E.L., 1972, *Management Science*, **18**, 401–405.
[19] Baase, S., 1993, *Computer Algorithms: Introduction to Design and Analysis*, 2nd ed. (Reading, MA: Addison-Wesley).
[20] Sedgewick, R., 1998, *Algorithms in C*, 3rd ed. (Reading, MA: Addison-Wesley).
[21] Korte, B. and Vygen, J., 2000, *Combinatorial Optimization* (Berlin: Springer).