Journal of the Operations Research
Society of Japan
Vol. 31, No. 4, December 1988

AN ALGORITHM FOR SINGLE CONSTRAINT MAXIMUM COLLECTION PROBLEM

Seiji Kataoka
Waseda University

Susumu Morito
Waseda University

(Received August 6, 1987; Revised June 7, 1988)

Abstract There are many studies which consider an optimal tour on a given graph including the well-known Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Most of these studies, however, assume that "all" nodes on a given graph should be visited exactly once or at least once. In this paper, we relax this assumption and consider the following problem:

"Given items with known values located at nodes of network, one wants to collect items so that their total value is maximized, under the assumption that a tour starting from the "center" node and returning to the center node is completed within a predetermined time limit."

Because of the added (time) constraint, one may not visit all nodes. The addition of this simple-looking constraint makes the problem difficult as it introduces an added dimention of selecting nodes to visit.

After a brief introduction, Section 2 presents two formulations of the problem, a native formulation and an improved one based on the introduction of self-loops to the graph corresponding to the problem. The latter formulation allows us to utilize solution strategies developed for the standard TSP. A branch and bound solution strategies together with a solution method of a relaxation problem is described in Section 3. A relaxation problem is generally an assignment problem with an added constraint for which an efficient branch and bound procedure is proposed. In particular, a recommended strategy for the selection of the branching variable is identified, and also an efficient procedure to transmit information from a branching problem to its sub-problems is proposed. Section 4 presents results of computational time requirements and basic characteristics of the proposed algorithm are clarified.

1. Introduction

There are many studies which consider an optimal tour on a given graph. Included here are such well-known problems as the Traveling Salesman Problem (TSP) or Vehicle Routing Problem (VRP). Most of these studies assume that "all" nodes on a graph should be visited exactly once or at least once.

In this paper, we study a problem of maximizing the total value collected by visiting nodes of a graph under a single constraint, say, a time constraint. Some known value is originally assigned to each node, and if one visits nodes, the associated values can be collected. One then wants to determine which nodes to visit and how (i.e., in what order) the selected nodes are visited in such a way that the total collected value is maximized and also that one returns to the starting node within the specified time limitation. This problem is very much similar to TSP and to VRP, but has an added dimension of selecting nodes to visit. As one of a family of routing problems, the problem of interest is simple and basic, and thus many practical applications are likely to exist.

D.Gensch[5] treated the problem, which he calls the optimal subtour problems, of finding the shortest tour under a single constraint. Gensch's algorithm does not guarantee to give an optimal solution, as his algorithm does not necessarily optimally solve the relaxed problem (called (R) in this paper), which is an assignment problem with a single linear side constraint. A heuristic algorithm is proposed by B.Golden[7] for the Gensch's problem.

2. Problem Formulations

2.1. Original formulation

Consider an asymmetric graph $G(V,E)$, where V $(|V|=n)$ and E denote node and arc sets, respectively. A directed arc, which we simply call an arc, from node $i\in V$ to node $j\in V$ is denoted as $(i,j)\in E$, whereas $c_i\geq 0$ and $t_{ij}\geq 0$ denote value allocated to node $i\in V$ and time required to travel arc $(i,j)\in E$, respectively. We consider a general problem with asymmetric distances. If distances are symmetric, simply set $t_{ij}=t_{ji}$. With a graphical image of the problem just described, a mathematical
programming formulation ⟨Formulation 1⟩ naturally results by associating variable x_{ij} to arc $(i,j)\in E$, and variable y_i to node $i\in V$. Both types of variables are 0-1 variables. In the subsequent formulation, $\bar{V}$ denotes those nodes selected to visit, i.e. $\bar{V}=\{i|i\in V, \sum_{j=1,j\neq i}^{n} x_{ij}=1\}$. S and $\bar{S}$ are a partition of $\bar{V}$ such that $S\cup\bar{S}=\bar{V}$, $S\cap\bar{S}=\phi$, $S\neq\phi$ and $S\neq\bar{V}$.

⟨Formulation 1⟩

(2.1) Max. $\sum_{i=1}^{n} c_i y_i$

(2.2) s.t. $\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} t_{ij}x_{ij}\leq \mathrm{TC}$

(2.3) $\sum_{j=1}^{n} x_{ij} = y_i$ for all i

(2.4) $\sum_{i=1}^{n} x_{ij} = y_j$ for all j

(2.5) $\sum_{i\in S}\sum_{j\in \bar{S}} x_{ij}\geq 1$ for any partition of $\bar{V}$, $(S,\bar{S})$

(2.6) $x_{ij}\in\{0,1\}$, $y_i\in\{0,1\}$, $y_1=1$ (node 1 is center node.)

⟨Formulation 1⟩, though natural as we described, has problems from a stand point of a solution algorithm.

A relaxed problem without a constraint (2.2) and also without so called multiple subtour elimination constraints (2.5) is nothing but the assignment problem and thus can be solved easily. However, variables y_i, $i\in V$, must be determined prior to solving the assignment problem.

If a Lagrangean relaxation is considered by putting constraints (2.2) and (2.5) into the objective function, Lagrange multipliers only appear as coefficients of x_{ij} and thus coefficients of y_i are not affected by Lagrange multipliers.

To alleviate these difficulties, we transform the graphical representation of the problem as discussed below.

2.2. Alternative formulation

⟨Transformation 1⟩ Assign a value coefficient $c_{ij} \geq 0$ to each arc $(i,j)\in E$. A value of an arc coefficient c_{ij} for each arc $(i,j)\in E$ is c_j, that is, $c_{ij}=c_j$, $i \in V\text{-}\{j\}$.

⟨Transformation 2⟩ Create a self-loop for each node and associate a variable $x_{ii} \in\{0,1\}$, $i\in V$. Cost and time coefficients of a self-loop are $c_{ii}=0$, $t_{ii}=0$, $i\in V\text{-}\{1\}$, respectively. For the center node, set $c_{11}=0$ and $t_{11}=\text{TC}$.

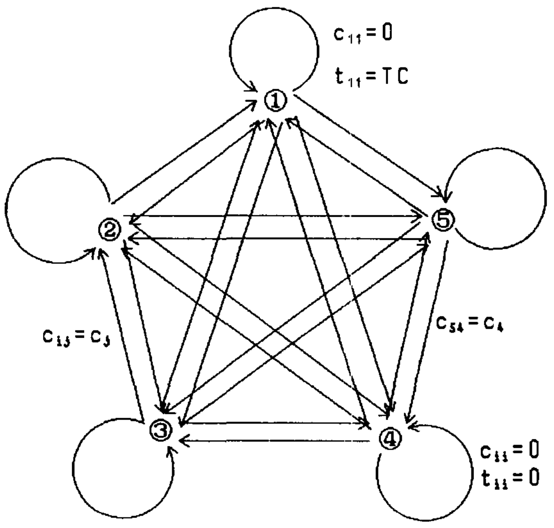


Fig. 2-1 The Transformed Graph

Based on the above transformation of the graph (Fig.2-1), our problem can be formulated as ⟨Formulation 2⟩. We note that self-loops are not regarded as subtours in the subtour elimination constraints (2.11). The problem thus formulated will be called (P).

⟨Formulation 2⟩

(2.7) (P) Max. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

(2.8) s.t. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \leq \text{TC}$

(2.9) $\sum_{j=1}^{n} x_{ij}=1$ for all i

(2.10) $\sum_{i=1}^{n} x_{ij}=1$ for all j

(2.11) $\sum_{i\in S}\sum_{j\in \bar{S}} x_{ij} \geq 1$ for any partition of $\bar{V}$, $(S,\bar{S})$

(2.12) $x_{ij} \in\{0,1\}$

⟨Formulation 2⟩ has the following properties which make it easier to apply a standard solution algorithm of TSP.

⟨Property 1⟩ The number of variables in ⟨Formulation 2⟩ is identical to that in ⟨Formulation 1⟩.

⟨Property 2⟩ Constraints (2.9), (2.10) are those of the assignment problem, and thus the relaxed problem without (2.8) and (2.11) can be solved by any algorithms for the standard assignment problem.

⟨Property 3⟩ When one considers, e.g., to obtain an upper bound, a Lagrangean relaxation by putting the knapsack-type constraint (2.8) and the subtour elimination constraints (2.11) into the objective function, Lagrange multipliers affect coefficients of all arcs $(i,j)\in E$.

⟨Property 4⟩ Nodes which are not to be visited will automatically be detected by self-loops, while satisfying assignment constraints.

3. Basic Solution Strategy and Algorithm for Solving Relaxed Problem

3.1. Basic solution strategy

An application of a branch and bound method will be considered to solve the problem. We first consider problem (R) which is problem (P) without subtour elimination constraints (2.11):

(3.1) (R) Max. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

(3.2) s.t. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \leq \mathrm{TC}$

(3.3) $\sum_{j=1}^{n} x_{ij}=1$ for all i

(3.4) $\sum_{i=1}^{n} x_{ij}=1$ for all j

(3.5) $x_{ij} \in\{0,1\}$

Because of the constraint (3.2), the constraints matrix of problem (R) still does not have the unimodularity, and thus problem (R) must be solved as a 0-1 integer programming problem. Obtaining a stronger upper bound improves the efficiency of a branch and bound method, and thus we use (R) as the relaxation problem, even though (R) still difficult to solve.

We first present the basic branch and bound solution algorithm for solving (P), assuming the existence of an optimization algorithm for solving problem (R), which will be discussed in 3.2.. Our basic branch and bound algorithm is essentially same as that for an asymmetric TSP[11].

An optimal solution of relaxation problem (R) generally consists of a number of subtours. This is clear as problem (R) is original problem (P) without subtour elimination constraints. If the solution consists of more than one subtour, then we select a non-self-loop arc (p,q) included in one of the subtours, and branch to two subproblems, one called the "right" subproblem in which x_{pq} is fixed to 1 so that arc (p,q) will always be taken, and the other called the "left" subproblem in which x_{pq} is fixed to 0 so that we never take arc (p,q). When there exists single subtour excluding self-loops for a subproblem, or no further branching from it is profitable, we stop branching. We continue branching until no unsolved subproblems are remaining. When no unsolved subproblems exist, the best solution obtained thus far is optimal.

3.2. Relaxation of problem (R)

We do not attempt to solve problem (R) directly as an integer program. Instead, to solve relaxation problem (R), we consider an LP-relaxation of (R), which we denote $(\bar{R})$. Here, we treat a minimization problem using the transformation $c'_{ij}=M-c_{ij}(\geq 0)$, where M is assumed to be a large constant.

$$
\begin{array}{llll}
(3.6) & (\bar{R}) \quad \text{Max.} & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
(3.7) & \qquad\quad \text{s.t.} & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \leq \text{TC} & \\
(3.8) & & \sum_{j=1}^{n} x_{ij}=1 & \text{for all } i \\
(3.9) & & \sum_{i=1}^{n} x_{ij}=1 & \text{for all } j \\
(3.10) & & 0\leq x_{ij} \leq 1 &
\end{array}
$$

Problem $(\bar{R})$ can be regarded as a linear relaxation of the assignment problem with a single additional linear constraint. Kobayashi[10] studied a more general problem of the minimum cost flow problem with a single additional constraint, and thus we solve problem $(\bar{R})$ by specializing the algorithm proposed by Kobayashi.

First, consider the following parametric problem $\bar{R}(\phi)$ by putting the "time" constraint (3.7) in the objective function with Lagrange multiplier ϕ:

$$
\begin{array}{llll}
 & \bar{R}(\phi) \quad \text{Min.} & \sum_{i=1}^{n}\sum_{j=1}^{n} c'_{ij}x_{ij} + \phi\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} c^{*}_{ij}(\phi)x_{ij} & \\
 & \qquad\quad \text{s.t.} & \sum_{i=1}^{n} x_{ij}=1 & \text{for all } j \\
(3.11) & & \sum_{j=1}^{n} x_{ij}=1 & \text{for all } i \\
 & & 0\leq x_{ij} \leq 1 & \\
 & & \text{where } c^{*}_{ij}(\phi)=c'_{ij}+\phi t_{ij} &
\end{array}
$$

Here, we denote an optimal solution of $\bar{R}(\phi)$ as $x^{*}_{ij}(\phi)$ and assume that $\eta=\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x^{*}_{ij}(\phi)$. Then, if for some ϕ , $x^{*}_{ij}(\phi)$ gives η=TC, then $x^{*}_{ij}(\phi)$ solves $(\bar{R})$. This implies that $(\bar{R})$ can be solved if $\bar{R}(\phi)$ can be solved parametrically in ϕ (Kobayashi[10]). Problem $\bar{R}(\phi)$ for a fixed value of ϕ is nothing but the linearly relaxed assignment problem, and thus an integer optimal solution $x^{*}_{ij}(\phi)$ can be found easily.

The relationship between ϕ and η is depicted in Fig.3-1 which forms a staircase graph. This is because:

(1) Within some interval of ϕ, the optimal integer solution remains identical, thus giving a horizontal portion of the graph.

(2) For some value of ϕ, there exist at least two distinct integer optimal solutions whose convex combination also gives an optimal non-integer solution. This corresponds to the vertical section of the graph.

(3) η is monotone non-increasing in ϕ.

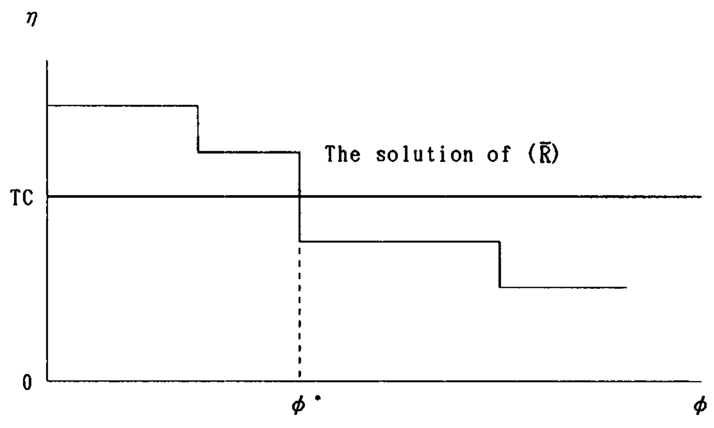


Fig. 3-1 Relationship between ϕ and η

An optimal solution of $(\bar{R})$ corresponds to the intersection of a horizontal straight line η=TC and the staircase graph as in Fig.3-1. We assume that these two graphs intersect at $\phi=\phi^*$. Let η_0 be the value of η corresponding to the optimal solution of $\bar{R}(0)$ with $\phi=0$. In case where there exist alternative integer optimal solutions of $\bar{R}(0)$, η_0 is the smallest value of η among these optimal solutions. If $\eta_0 <$TC, η=TC would never intersect with the staircase graph because of its non-increasing property. The solution which gives η_0 would then be the optimal solution of the original problem. Similarly the two graphs would not intersect if eta corresponding to the optimal solution of $\bar{R}(\infty)$ exceeds TC. Then $x_{ii}=1$ $(i \in V)$ is optimal, indicating that one cannot visit any node due to the time constraint.

We try to find an optimal solution of $(\bar{R})$ by solving $\bar{R}(\phi)$ parametrically using the following two steps systematically:

1) a step to increase ϕ

2) a step to decrease η

Each of these steps can be formulated as follows, where the dual of $\bar{R}(\phi)$ is denoted as $D(\phi)$, for a given $\bar{\phi}$, $\bar{v}$ as an optimal solution of $D(\bar{\phi})$, $\bar{x}$ as an optimal solution of $\bar{R}(\bar{\phi})$.

Primal Problem (a step to decrease η)

$$\begin{array}{lll} RP(\bar{\phi},\bar{v}) & \text{Min.} & \eta=\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x_{ij} & \\ & \text{s.t.} & \sum_{i=1}^{n} x_{ij}=1 & \text{for all } j \\ & & \sum_{j=1}^{n} x_{ij}=1 & \text{for all } i \\ & & x_{ij}=0 & \text{if } \bar{v}_j\text{-}\bar{v}_i < c^*_{ij}(\phi) \\ & & x_{ij} \geq 0 & \text{if } \bar{v}_j\text{-}\bar{v}_i=c^*_{ij}(\phi) \end{array}$$

Note that any feasible solution of $RP(\bar{\phi},\bar{v})$ is an optimal solution of $\bar{R}(\bar{\phi})$.

Dual Problem (a step to increase ϕ)

$$\begin{array}{llll} RD(\bar{x}) & \text{Max.} & \phi & \\ & \text{s.t.} & v_j\text{-}v_i\text{-}\phi t_{ij} \leq c'_{ij} & \text{if } \bar{x}_{ij}=0 \\ & & v_j\text{-}v_i\text{-}\phi t_{ij} = c'_{ij} & \text{if } \bar{x}_{ij}=1 \\ & & v_i,v_j,\phi \geq 0 & \end{array}$$

Note that if $(\bar{\phi},\bar{v})$ is any feasible solution of $RD(\bar{v})$, then $\bar{v}$ is an optimal solution of $D(\bar{\phi})$.

Algorithm SOL$\bar{R}$

STEP1 Solve $\bar{R}(0)$ and obtain an integer optimal solution, $\bar{x}$. Let η_0 be the value of η corresponding to $\bar{x}$, or be the smallest value of η if there exist alternative integer optimal solutions. If η_0 <TC, $\bar{x}$ solves $(\bar{R})$, and STOP. Otherwise, go to STEP2.

STEP2 Solve $RD(\bar{x})$. If a finite optimal solution $(\bar{\phi},\bar{v})$ is found, go to STEP3. Otherwise ($\phi=\infty$), STOP.

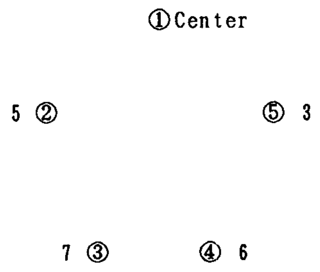
STEP3 Solve $RP(\bar{\phi},\bar{v})$. Let $\bar{x}$ be the optimal solution and η be the corresponding value of η (smallest value in case of alternative solutions). If $\eta \leq$TC, compute x such that η=TC, based on information obtained before solving $RP(\bar{\phi},\bar{v})$ and STOP. Otherwise, return to STEP2.

Refer to Kobayashi[10] for the solution algorithms of $RD(\bar{x})$ and $RP(\bar{\phi},\bar{v})$. We note that in STEP1, $\bar{R}(0)$ can be solved by minimizing η with x_{ii} set to 0 for all $i \in V$. Also note that algorithm SOL$\bar{R}$ is applicable even when some x_{ij} are fixed.

We now examine basic characteristics of SOL$\bar{R}$ which can be exploited to improve the algorithm itself. Firstly, $RP(\bar{\phi},\bar{v})$ solves the linearly relaxed assignment problem with only those variables for which the corresponding constraints of $RD(\bar{x})$ are satisfied with equality at the optimal solution of $RD(\bar{x})$, and with all other variables set to 0. As a result, the effective number of variables in $RP(\bar{\phi},\bar{v})$ is small, and thus $RP(\bar{\phi},\bar{v})$ can be solved with ease. This implies that the effective number of variables in $RP(\bar{\phi},\bar{v})$ is relatively small, and thus efforts to solve $RP(\bar{\phi},\bar{v})$ repeatedly while solving $(\bar{R})$ within the branch and bound framework to solve (R) is not substantial. Instead, a major portion of computational requirements emanates from efforts to solve $RD(\bar{x})$ when $(\bar{R})$ is solved repeatedly in the overall branch and bound algorithm for (R). Therefore, reduction of the computational requirements of $RD(\bar{x})$ is expected to improve the overall algorithmic efficiency. Also it is known that, when the optimal solution of $(\bar{R})$ found by algorithm SOL$\bar{R}$ is fractional (i.e., non-integer), at least one of the variables taking fractional value corresponds to a self-loop. The reason why this is the case and the algorithmic improvement based on this property would be discussed in the next section.

3.3. Selection rule of the branching variable

A branch and bound procedure is developed to solve the relaxed problem (R) using information obtained from $(\bar{R})$. We note here that the coefficient matrix (c_{ij}) has a special structure such that non-diagonal entries of an arbitrary column are identical, as was observed earlier in the discussion of the graph transformation (Fig.3-2). We now propose a selection rule of the branching variable which exploits this special structure.



c_{ij}	1	2	3	4	5
1	0	5	7	6	3
2	0	0	7	6	3
3	0	5	0	6	3
4	0	5	7	0	3
5	0	5	7	6	0

Fig. 3-2 An example of objective coefficient matrix (c_{ij})
A number beside node i indicate value c_i.

$\bar{R}(\phi^*)$ has at least two alternative optimal integer solutions, and the maximum and the minimum of the corresponding value of η are denoted by η_U, and η_L, respectively. The corresponding optimal solutions are similarly denoted by $x_U(\phi^*)$ and $x_L(\phi^*)$. Refer to Fig.3-3, where $\alpha=|\eta_U\text{-TC}|/|\eta_U\text{-}\eta_L|$. Note that $x_L(\phi^*)$ gives a feasible solution of (R) and thus can be regarded as a candidate for the incumbent solution.

When the final value of η=TC is reached in STEP3 of algorithm SOL$\bar{R}$ and if the corresponding optimal solution of $(\bar{R})$ is non-integer, at least one of self-loops starts to emerge as shown in Fig.3-3. To see that this is in fact the case, we assume that the variable which takes a fractional value in the optimal solution of $(\bar{R})$ does not correspond to any of self-loops. Then, because of the special structure of the cost coefficient matrix (c_{ij}), the value of $cx^*(\phi^*)$ would remain identical while η is reduced from η_U to η_L. This implies that the optimal value of objective function for $(\bar{R})$ coincides with the objective function value $cx_L(\phi^*)$ of a feasible integer solution $x_L(\phi^*)$ of (R), which indicated that $(\bar{R})$ has an optimal integer solution $x_L(\phi^*)$ and thus (R) is solved and no further branching is necessary. Therefore, if the branching is to continue, at least one of variables which take fractional values in an optimal solution of $(\bar{R})$ must correspond to a self-loop.

When the optimal solution of $(\bar{R})$ takes the form as shown in Fig.3-3, anyone of 4 arcs

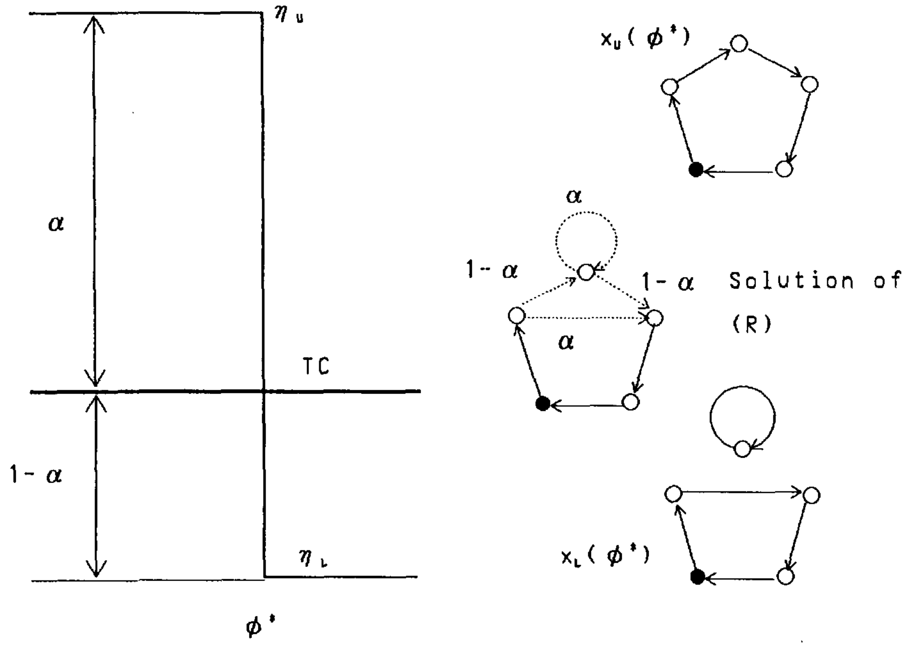


Fig. 3-3 A process to decrease η

The weight for each node $1 \leqq c \leqq 10$, uniformly distributed

The time for each arc $50 \leqq t \leqq 70$, uniformly distributed

Time constraint TC=50*n

Each entry reflects an avearge time (in second) for 10 instances, using personal computer NEC, PC-9801VM2, MS-FORTRAN, Ver. 3. 3 compiler.

Time 1 arbitrary branching rule

Time 2 self-loop branching rule

Table 3-1 The effect of the branching rules

n	5	6	7	8	9	10	15	20	25	30
Time 1	25.9	65.1	170.5	189.5	163.5	—	—	—	—	—
Time 2	4.7	5.2	10.8	9.0	6.9	28.6	25.7	43.5	71.3	131.9

can be regarded as a candidate for branching variables. Considering the special structure of the matrix (c_{ij}), however, it is expected to be more beneficial to select a fractional variable that corresponds to a self-loop as a branching variable rather than to select arbitrarily any one of fractional variables. This strategy has an additional merit that the depth of enumeration levels will never exceed n, as there exist at most n self-loops. A simple experiment was performed to observe the effect of this branching rule. Table 3-1 shows that the self-loop branching strategy performs 10 to 20 times faster than the arbitrary branching rule. It is expected that the computational time does not explode until n reaches approximately 30.

3.4. Determination of initial value of ϕ

When $(\bar{R})$ is solved by algorithm SOL$\bar{R}$, we have thus far assumed the initial parameter value of $\phi=0$. When the value of TC is relatively small, the process of increasing ϕ from 0 in STEP2 and of eventually reaching ϕ^* is expected to take time. For example, in Fig.3-4, solving a problem with time limitation of TC=TC2 would require more efforts than solving the problem with TC=TC1 (TC1>TC2) provided that all other conditions remain identical.

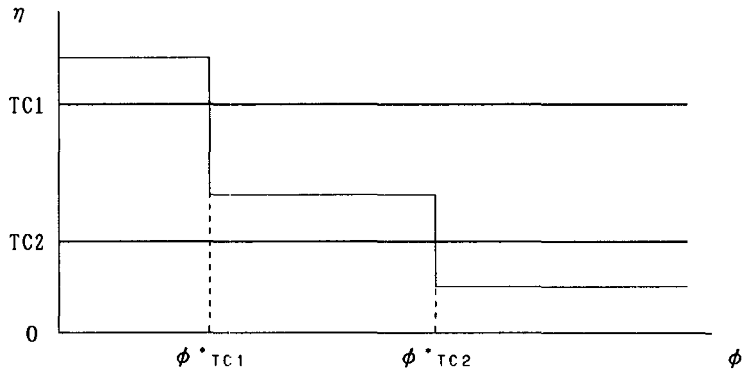


Fig. 3-4 Relationship between the magintude of TC and ϕ^*

Note, however, that algorithm SOL$\bar{R}$ will find an optimal solution of $(\bar{R})$ as long as the starting value of ϕ is nonnegative and less than or equal to ϕ^*. A procedure is now presented to get an initial value of ϕ so that $(\bar{R})$ can be solved more efficiently.

Here, the self-loop selected as a branching variable is denoted as x_{rr}, ϕ_1^* as the value of ϕ corresponding to the optimal solution of the right subproblem $(\bar{R} \mid x_{rr}=1)$, and ϕ_0^* as the value of ϕ corresponding to the optimal solution of the left subproblem $(\bar{R} \mid x_{rr}=0)$. In the subsequent discussion, we assume that the value of η corresponding to the optimal integer solution of $\bar{R}(\phi^*)$ is not equal to TC, as otherwise problem (R) is already solved. We note the following observations:

(1) $x_U(\phi^*)$ gives an optimal solution of $\{\bar{R}(\phi^*) | \; x_{rr}=0\}$ and η_U >TC.

(2) $x_L(\phi^*)$ gives an optimal solution of $\{\bar{R}(\phi^*) | \; x_{rr}=1\}$ and η_L <TC.

(3) $(\phi_1-\phi_2)(\eta_1-\eta_2)\leq 0$ where $\eta_k(k=1,2)$ denotes the value of η corresponding to the optimal solution of $\bar{R}(\phi_k)$.

The fact that η=TC holds for the optimal solutions of $(\bar{R} \mid x_{rr}=0)$ and $(\bar{R} \mid x_{rr}=1)$, together with the above observations indicates that $(\phi_0^*-\phi^*)(\mathrm{TC}-\eta_U)\leq 0$ or $(\phi_1^*-\phi^*)(\mathrm{TC}-\eta_L)\leq 0$. This would then imply the following inequality:

(3.12) $$\phi_1^* \leq \phi^* \leq \phi_0^*$$

This suggests that STEP1 of algorithm SOL$\bar{R}$ for the left subproblem $(\bar{R} \mid x_{rr}=0)$ can be initiated with the starting value of ϕ equal to ϕ^* of its "parent" problem $(\bar{R})$. Similarly, the right subproblem can be solved by the following algorithm SOL$\bar{R}'$ which exploits the information ϕ^* of the parent subproblem. Here $RP'(\bar{\phi},\bar{v})$ is a maximization (instead of minimization) problem for the same constraints and the objective function as $RP(\bar{\phi},\bar{v})$, and $RD'(\bar{x})$ a minimization problem.

```
Algorithm SOL$\bar{R}'$
STEP1 Set $\phi=\phi^*$, and go to STEP3.
STEP2 Solve $RD'(\bar{x})$, (Note, in this case, that $RD'(\bar{x})$ always has an optimal solution.) and
      go to STEP3.
STEP3 Solve $RP'(\bar{\phi},\bar{v})$ and obtain its optimal solution $\bar{x}$. Compare the corresponding value
      of $\eta$. If $\eta \geq$TC, calculate $x$ such that $\eta$=TC and STOP. Otherwise, return to STEP2
```

Using the mechanism just described, information obtained in the parent problem can be transmitted to subproblems through a single parameter ϕ, thus allowing an efficient branch and bound procedure.

4. Design of Experiments and Results

4.1. Objective of experiments

The objective of experiments is to study characteristics of the proposed algorithm considering effects on execution time requirements of factors such as time limit parameter TC, range of travel time parameter t_{ij}, and of value parameter c_i, as well as sliding of value parameter c_i, which determine a problem instance. All these parameters are assumed to be integers. The results presented are obtained using a personal computer NEC-9801VM2 (10MHz) with Pro-Fortran77 compiler.

4.2. Numerical experiments

⟨Experiment 1⟩

Factor: Time Limit Parameter TC

We consider instances with n=10 nodes, c_i uniformly distributed between $5\leq c_i \leq 15$, t_{ij} uniformly distributed between $30\leq t_{ij} \leq 70$, and time constraint parameter TC set to one of selected values 60, 80, 100, 150, 200, 250, 300, 350, 400. Here, the minimum value of

TC=60 is based on the minimum possible value of any meaningful trips between the origin and any one of other nodes. In tables, the number of nodes visited includes the origin, thus the number of nodes visited is 1 if there exist no feasible tours within TC. And here, the case with TC=250 roughly corresponds to visiting half of nodes, as the average of t_{ij} is 50. Thus, the case with TC=250 can be regarded as an "average" of various cases. Experiments 2 and 3 which follow study algorithmic performance based on the "average" case. Each entry of Table 4-1 reflects an average of 50 repetitions.

Table 4-1 indicates that the closer one gets to the average the more computer time is required. That is, more time is required when the number of visited nodes is roughly same as that of unvisited nodes. When TC exceeds 400, one can often find without difficulties a feasible tour which visits all nodes, and thus no other better solutions exist and algorithm stops. The reason why the case of TC=60 is taking some time (as opposed to the case of TC=400), is because the algorithm do not terminate until ϕ, which is gradually increased from 0 as explained in 3.2. or 3.4., reaches infinity.

⟨Experiment 2⟩

Factor: Range of Time Parameter t_{ij}

We consider instances with n=10 nodes, and c_i uniformly distributed between 5 and 15. Time parameters t_{ij} are also uniformly distributed with mean 50, and to see effects of the magnitude of their ranges, the following five types of ranges will be considered;

$t_{ij}=50$ (no variability)

$40 \leq t_{ij} \leq 60$

$30 \leq t_{ij} \leq 70$

$20 \leq t_{ij} \leq 80$

$10 \leq t_{ij} \leq 90$.

Time constraint parameter TC is set to 250, i.e., the average case as explained in ⟨Experiment 1⟩. Experiments are repeated 50 times for each parameter combination (Table 4-2).

When t_{ij} has no variability, i.e., t_{ij}=50, the optimal solution can be obtained by visiting 5 nodes such that the corresponding c_i's are top five. The heuristic algorithm to generate an initial incumbent solution always produced this optimal solution. Moreover, the corresponding optimal value coincides with the optimal value of the relaxation problem without subtour elimination constraints, and thus the algorithm terminates quickly. The tendency that the greater variability in t_{ij} results in faster execution reflects the fact that the chance of quickly finding a tour which visits all nodes increases as the variability in t_{ij} increases. This is because the algorithm generally tries to make a tour by selecting those arcs whose t_{ij}'s are small. This tendency can be observed by studying the number of visited nodes for each instance.

⟨Experiment 3⟩

Factor: Range of Value Parameter c_i

Here, we consider instances with n=10, and t_{ij} uniformly distributed between 30 and 70. Value parameters c_i are also uniformly distributed with mean 10, and with the following four types of range;

c_i=10 (no variability)

$9 \leq c_i \leq 11$

$5 \leq c_i \leq 15$

$1 \leq c_i \leq 19$

As in ⟨Experiment 3⟩, TC is fixed constant at 250, and the number of repetitions for each case is 50 (Table 4-3).

Table 4-3 shows that problems can be solved quickly when all c_i's are identical. This is because the greatest common divisor of c_i's denoted as $GCD(c_i)$ is calculated first in the program developed, and is used to fathom subproblems. More specifically, if there exist any solutions better than an incumbent solution with the objective function value, say, z_0, their values of the objective function should be at least $z_0+GCD(c_i)$. If $z_0+GCD(c_i)$ exceeds the optimal value of the objective function of LP-relaxed subproblem $(\bar{R})$, the subproblem can be fathomed.

⟨Experiment 4⟩

Factor: Magnitude of Value Parameter c_i (with fixed range)

Here, we consider instances with n=10, and t_{ij} uniformly distributed between 30 and 70. Value parameters c_i are also uniformly distributed with range 20, and with the following four types of range;

$1 \leq c_i \leq 21$

$10 \leq c_i \leq 30$

$20 \leq c_i \leq 40$

As in ⟨Experiment 4⟩, TC is fixed constant at 250, and the number of repetitions for each case is 50 (Table 4-4).

Table 4-4 shows that the higher the mean of c_i, the more computations would be required. This is because, as c_i's are slided "up" while fixing the length of interval, the difference between the optimal objective function value of $(\bar{R})$ and $cx_L(\phi^*)$ tends to be larger. As a result, the chance is reduced that a condition "$cx_L(\phi^*)+GCD(c_i)>$the optimal objective function value of $(\bar{R})$" is satisfied which indicates fathoming of subproblem $(\bar{R})$, thus requiring more computations.

Table 4-1 The results of <Experiment 1> ("# of V.N." is the nubmer of visited nodes)

TC	60	80	100	150	200	250	300	350	400
CPU Time (sec)	17.10	34.24	48.86	77.42	89.08	71.40	46.52	16.78	0.54
# of V.N.	1.00	1.64	2.22	3.34	4.70	6.18	7.50	8.72	9.92

Table 4-2 The results of <Experiment 2> ("# of V.N." is the nubmer of visited nodes)

The range of t	t=50	40≦t≦60	30≦t≦70	20≦t≦80	10≦t≦90
CPU Time (sec)	7.54	102.8	71.40	35.94	3.74
# of V.N.	5.00	5.04	6.18	7.42	8.94

Table 4-3 The results of <Experiment 3>

The range of c	c=10	9≦c≦11	5≦c≦15	1≦c≦19
CPU Time (sec)	10.28	61.28	71.40	54.18

Table 4-4 The results of <Experiment 4>

The range of c	1≦c≦21	10≦c≦30	20≦c≦40
CPU Time (sec)	55.82	72.74	85.36

5. Conclusion

This paper considered a variant of the traveling salesman problem in which one wants to maximize the total value collected by visiting nodes within a given "time" limitation. Efficiency of the proposed algorithm was evaluated by extensive computational experiments. We conclude the paper with the following conclusions.

(i) For the problem of interest, we showed that the choice of a proper formulation, as opposed to a more naive formulation, allows us to exploit existing results for, e.g., the traveling salesman problem.

(ii) The branching variable selection rule which gives higher priority to variables corresponding to self-loops produces an efficient branch and bound algorithm for the problem.

(iii) Information of a parent problem can be communicated to its subproblems in an extremely simple fashion using only one parameter.

Acknowledgments

The authors would like to express their appreciations to anonymous referees for their helpful comments which improved the clarity of the paper.

References

[1] Barr, R. R., Glover, D. and Klingman, D.: The Alternating Bases Algorithm for Assignment Problems. *Mathematical Programming*, Vol.13, No.1 (1977), 11-13.

[2] Everett III, H.: Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, Vol.11, No.2 (1963), 399-417.

[3] Garey, M. R. and Johnson, D. S.: *Computers and Intractability* : A Guide to the Theory of NP-Completeness, W.H.Freeman and Company, San Francisco, 1978.

[4] Gavish, B. and Pirkul, H.: Efficient Algorithms for Solving Multiconstraint Zero-one Knapsack Problems to Optimality. *Mathematical Programming*, Vol.31, No.1 (1985), 78-105.

[5] Gensch, D. H.: An Industrial Application of the Traveling Salesman's Subtour Problem. *AIIE Transactions*, Vol.10, No.4 (1978), 362-370.

[6] Glover, F., Karney, D., Klingman, D. and Russell, R.: Solving Singly Constrained Transshipment Problems. *Transportation Science*, Vol.12, No.4 (1978), 277-297.

[7] Golden, B., Levy, L. and Dahl, R.: Two Generalizations of the Traveling Salesman Problem. *OMEGA*, Vol.9, No.4 (1981), 439-445.

[8] Iri, M. and Kobayashi, T.: *Network Theory* (in Japanese), Nikkagiren-shuppan, 1976.

[9] Kao, E. P. C.: A Preference Order Dynamic Program for Stochastic Traveling Salesman Problem. *Operations Research*, Vol.26, No.6 (1978), 1033-1045.

[10] Kobayashi, T.: The Lexico-Shortest Route Algorithm for Solving the Minimum Cost Problem with an Additional Linear Constraint. *Journal of the Operations Research Society of Japan*, Vol.26, No.2.5 (1983), 167-185.

[11] Konno, H. and Suzuki, H.: *Integer Programming Problems and Combinatorial Optimization* (in Japanese), Nikkagiren-shuppan, 1982.

[12] Lawler, L.: *Combinatorial Optimization* : Networks and Matroids, Holt, Rinehart and Winston, New York, 1977.

[13] Lawler, L.: *Traveling Salesman Problem* : A Guided Tour of Combinatorial Optimization Problem, John Wiley and Sons, Chichester, 1985.

[14] Masch, V.: Cyclic Method of Solving the Transshipment Problem with an Additional Linear Constraint. *Networks*, Vol.10, No.1 (1980), 17-31.

[15] Murty, K. G.: Fundamental Problem in Linear Inequalities with Applications to the Traveling Salesman Problem. *Mathematical Programming*, Vol.2, No.3 (1972),

296-308.

[16] Murty, K. G.: On The Tour of a Traveling Salesman. *SIAM Journal of Control*, Vol.7, No.1 (1969), 1122-1131.

[17] Murty, K. G.: Adjacency on Convex Polyhedra. *SIAM Review*, Vol.13, No.3 (1971), 377-386.

[18] Tarjan, P. E.: *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983.

[19] Tone, K.: *Mathematical Programming* (in Japanese), Asakura-shoten, 1978.

Seiji KATAOKA: Department of Industrial Engineering and Management, School of Science and Engineering,
Waseda University, Shinjuku-ku,
Tokyo, 169, Japan.