Univ. Beograd. Publ. Elektrotehn. Fak.
Ser. Mat. 7 (1996), 31–35.

A BRANCH AND BOUND ALGORITHM FOR SOLVING A GENERALIZED TRAVELING SALESMAN PROBLEM

Vladimir Dimitrijević, Milan Milosavljević, Milan Marković

Following the basic model of the Traveling Salesman Problem (TSP), we consider a more general combinatorial optimization problem: TSP on multipartite (mp) digraphs, known as Generalized TSP (GTSP). The GTSP can be stated as: find a minimum cost cycle which includes exactly one vertex from each supervertex in an mp-digraph. We propose a branch and bound algorithm for solving the GTSP which is based on the minimal rooted tree as a relaxation. Presented experimental results point to the problem dimension up to which it can be solved in a reasonable time, depending on the available computer resources.

1. INTRODUCTION

Let $G_{s,n}(s, n \geq 2)$ be a weighted, complete, multipartite digraph (in the following an mp-digraph). The parameter s is the number of groups of vertices, called *supervertices*, and n is the number of vertices within each supervertex. Arcs join vertices between different supervertices, while there are no arcs between any two vertices within a supervertex. Note that the associated distance matrix C is a block matrix with s^2 square blocks of order n. Each block contains the arcs lengths (distances) between vertices in the corresponding supervertices. Diagonal blocks contain only a sufficiently large positive constant thereby eliminating connections within a supervertex in a solution.

The Generalized Traveling Salesman Problem (GTSP) can be stated as

[0]1991 Mathematics Subject Classification: 05C85

Problem: *Find a minimum cost s-cycle which includes exactly one vertex from each supervertex in an mp-digraph.*

The GTSP is really a *generalized* Traveling Salesman Problem (TSP) because TSP is a special case of GTSP; indeed, TSP is GTSP with vertex sets of cardinality one. This generalization simultaneously combines the decision of vertex selection and vertex sequencing.

Since the vertex set possess a "multiple choice" structure, the GTSP allows alternatives to be considered in the decision process. The first applications of a GTSP were presented in [**5**] for sequencing computer files. Various applications of GTSP are cited in [**9**] (see also [**6**]) with respect to warehouse order picking with multiple stock locations, airport selection and routing for courier planes, postal routing, routing of welfare clients through governmental agencies and certain types of flexible manufacturing scheduling. Other relevant references are [**11**], [**8**], [**7**], [**10**].

In the sequel, we shall describe a branch and bound algorithm for solving the GTSP based on [**2,3**]. The problems of analysis and synthesis of sub-optimal algorithms or heuristics for solving the GTSP, as well as the problems related to the finding good lower bounds for the given relaxation tasks are stil open.

2. A BRANCH AND BOUND ALGORITHM

In [**2,3**] a branch and bound algorithm for the GTSP based on the minimal rooted directed tree as relaxation is proposed. This algorithm is referred to the "open" variant of the GTSP, i.e. when a salesman is interested in finding a minimum cost (s-1)-path which contains exactly one vertex from each supervertex. Also, a branch and bound algorithm for the GTSP, which combines assignments and the Lagrangian relaxation technique, was presented in [**9**] about the same time. The discussion in the paper is limited to the consideration of the open variant of the GTSP [**2,3**].

For the open variant of the GTSP it was reasonable to select the minimal rooted directed tree problem as the relaxation task, since a path is a directed tree with the property that exactly one arc starts at each vertex, except at a vertex t, the so-called *terminal vertex.*

The problem of determining the minimal rooted tree can be solved by J. EDMOND's algorithm having complexity $O(n^2)$ [**4**].

In solving the relaxation task we have used a variant of the algorithm in which a *fictive* equidistante vertex was added to the original mp-digraph G. Thus it was possible to find a minimal arborescence without prior specification of the root.

A characteristic of EDMOND's algorithm is that in its *forward* phase it finds a minimal arborescence by successively including a *minimal arc* at each vertex i. (The minimal arc of a vertex is the shortest arc terminating at this vertex.) In the case where a cycle is formed, all the vertices of the cycle are merged into a a new vertex called a *pseudo-vertex.* In the next iteration, the algorithm continues on the new digraph with a reduced number of vertices: pseudo-vertex and all the remaining vertices which are outside of the pseudo-vertex. All the vertices in the reduced digraph are treated equally.

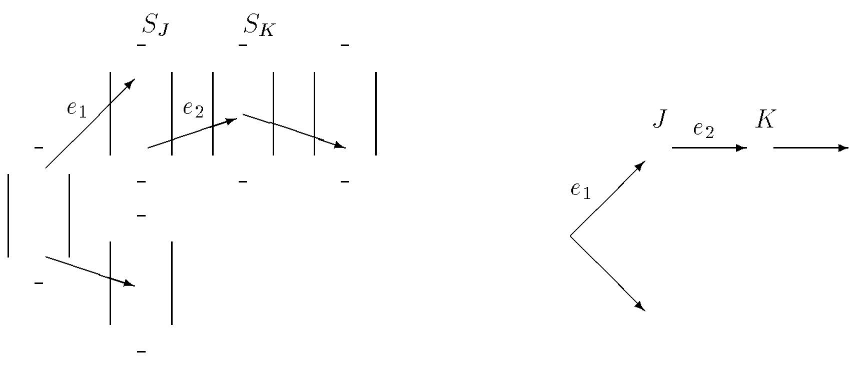

Fig. 1. Expanded and corresponding merged arborescence

Within the initialization of the algorithm this characteristic of the "shrinking vertices" was used to transform every supervertex into a pseudo-vertex, thus reducing the dimension of the digraph. In other words, finding a minimal arborescence was always performed on the digraph with only s vertices. This is a favorable property of the chosen relaxation task.

In the second, *backward* phase, the algorithm expands the pseudo-vertex only to the level of supervertex. Namely, it does not expand any pseudo-vertex obtained at the initialization step by shrinking vertices within the supervertex. In this way, one obtains a minimal (s-1)-arborescence without insight into the structure of a supervertex. Now, one can apply any branching rule which is suitable for the standard TSP branch and bound algorithm (with this kind of relaxation).

The first vertex in the minimal arborescence with more than one starting arcs was chosen as a basis for branching in the search tree.

Let I_k be the set of already included and E_k the set of already excluded arcs of digraph G at step k of the branch and bound algorithm. Let $A = \{e_1, e_2, \ldots, e_l\}$ be the set of starting arcs at vertex i. The applied branching rule generates $l+1$ new subproblems, i.e. sons of the problem k in the search tree:

$$\begin{array}{llll}
E_{k1} & = E_k \cup A - \{e_1\}\ , & I_{k1} & = I_k \cup \{e_1\}\ ; \\
E_{k2} & = E_k \cup A - \{e_2\}\ , & I_{k2} & = I_k \cup \{e_2\}\ ; \\
\vdots & & \vdots & \\
E_{kl} & = E_k \cup A - \{e_l\}\ , & I_{kl} & = I_k \cup \{e_l\}\ ; \\
E_{k,(l+1)} & = E_k \cup A \qquad\quad , & I_{k,(l+1)} & = I_k\ .
\end{array}$$

In the case, when the (s-1)-arborescence is a path, then supervertices (i.e. corresponding pseudo-vertices) are expanded and one goes into their structure, looking for a supervertex with discontinuity, see Fig. 1. If there is no discontinuity (in each supervertex the arc enters and leaves it at the same vertex) this solution was a candidate for an optimal one. In the other case, the following branching rule was applied for the first detected supervertex with discontinuity:

$$\begin{array}{ll} E_{k1} = E_k \cup \{e_1\} \quad , & I_{k1} = I_k \cup \{e_2\} \ ; \\ E_{k2} = E_k \cup \{e_2\} \quad , & I_{k2} = I_k \cup \{e_1\} \ ; \\ E_{k3} = E_k \cup \{e_1, e_2\} \ , & I_{k3} = I_k \ , \end{array}$$

where the arc e_1 enters and e_2 leaves the supervertex J.

The starting point in coding the branch and bound algorithm was a general implicit enumeration algorithm for solving any *NP*-hard combinatorial optimization problem. This general program is a part of the programming package "TSP-SOLVER" [**1**]. Implemented modifications refer to: relaxation task, checking feasibility of solution, and branching rules.

At the present stage of the algorithm development we choose the depth-first search variant of the branch and bound algorithm, without any heuristic for the upper bound estimate.

3. EXPERIMENTAL RESULTS

We have created the set of test problems by specifying the number of supervertices s and the number of vertices n within a supervertex. The arc costs were generated according to non-Euclidean method as in [**6**], drawn from a uniform [0,1000] distribution and then rounded to the nearest integer. For each combination of s and n ten examples were created. In Table 1, the average values of VAX8800 CPU-time and the number of solved relaxation tasks referred to the ten solved examples are presented.

Table 1: Efficiency of the proposed branch and bound algorithm

Problem dimensions		Average CPU-time	Average number of
s	n	VAX 8800 (in sec)	solved relaxations tasks
7	4	4.71	340
8	4	12.24	699
9	4	33.62	1373
10	4	123.02	4113
11	4	309.02	8288
12	4	646.35	14779
13	4	1426.85	27485

REFERENCES

1. D. Cvetković, M. Čangalović, V. Dimitrijević, L. Kraus, M. Milosavljević, S. Simić: TSP-SOLVER - *a programing package for the traveling salesman problem.* These Publications, **1** (1990), 41–47.
2. V. Dimitrijević, M. Marković, M. Milosavljević: *A traveling salesman problem on the multipartite digraph* (Serbian). Zbornik radova XV simpozijuma o informacionim tehnologijama, pp. 202-1 - 202-8, Sarajevo '91, 1991.
3. V. Dimitrijević, M. Marković, M. Milosavljević: *The optimal solving of the traveling salesman problem on the multipartite digraph* (Serbian). Zbornik radova XVIII konferencije SYM-OP-IS '91, 8-11.X.1991, pp. 106–109, ed. R. Stanojević and J. Vuleta, Beograd, 1991.
4. M. Gondran, M. Minoux: *Graphs and Algorithms.* John Wiley and Sons, Chichester - New York - Brisbane - Toronto - Singapore, 1984.
5. A. L. Henry-Labordere: *The record balancing problem: a dynamic programming solution of a generalized travelling salesman problem.* RIRO, vol. (B-2) (1969), 43–49.
6. G. Laporte, H. Mercure, Y. Nobert: *Generalized travelling salesman problem through n sets of nodes: the asymmetrical case.* Discrete Appl. Math, **18** (1987), 185–197.
7. A. M. Meilahs, M. V. Pospelova: *A generalization of the traveling salesman problem* (Russian). Dynamic System and Control, Saransk, 137-140, 1988.
8. C. E. Noon, J. C. Bean: *An efficient transformation of the generalized traveling salesman problem.* Working Paper, Department of Management. University of Tennessee, Knoxville, 1989.
9. C. E. Noon, J. C. Bean: *A Lagrangian based approach for the asymmetric generalized traveling salesman problem.* Operations Research, **39** (1991) No. 4, 623–632.
10. S. S. Srivastava, S. Kumar, R. C. Garg, P. Sen: *Generalized travelling salesman problem through n sets of nodes.* CORS J., **7** (1969), 97–101.
11. P. Toth, M. Fischetti: *The generalized traveling salesman problem* (to appear).

Institute of Applied Mathematics and Electronics,
Kneza Miloša 37, 11000 Belgrade,
Yugoslavia

(Received May 26, 1995)