Yugoslav Journal of Operations Research
2 (1992), Number 2, 163-185

A SURVEY OF SOME NON-STANDARD TRAVELING SALESMAN PROBLEMS

Dragoš CVETKOVIĆ

University of Belgrade, Faculty of Electrical Engineering,
P. O. Box 816, 11001 Belgrade, Yugoslavia

Vladimir DIMITRIJEVIĆ
Milan MILOSAVLJEVIĆ

Institute of Applied Mathematics and Electronics
11000 Belgrade, Yugoslavia

Abstract. We present a survey of some non-standard traveling salesman problems (TSP) with emphasis on authors' contributions. The results include complexity indices and k-best solutions for the ordinary TSP, the TSP on a chained structure, a multi-TSP on a bandwidth-limited digraph as well as the corresponding general case, a generalized TSP as well as the description of some software for TSP.

Key words and phrases: operations reseaarch, combinatorial optimization, traveling salesman problem, branch and bound, dynamic programming, bandwidth-limited graphs, multipartite graphs, complexity of algorithms, software.

0. INTRODUCTION

The purpose of this expository paper is to present our work on the traveling salesman problem and its variations and to announce our book **[23]** on the same subject. One of the reasons for this is that a part of results is either unpublished or published in Serbo-Croatian language (see the list of references).

Section 1 presents definitions and necessary facts related to the traveling salesman problem. Section 2 introduces several variations of basic problem calling all of them non-standard problems. Section 3 contains the results classified in several subsections.

1. TRAVELING SALESMAN PROBLEM

Suppose a salesman, starting from his home city, is to visit exactly once each city on a given list of cities and then to return home. It is reasonable for him to select the order in which he visits the cities so that the total of the distances traveled in his tour is as small as possible. This problem is called *traveling salesman problem* (TSP).

TSP is a typical problem of *combinatorial optimization*. There is an extensive literature on and an impressive theory of TSP. The theory includes algorithms and heuristics (with an emphasis on complexity questions) for solving TSP as well as several variations and related problems. There are applications of TSP in operations research and engineering. A nice monograph [**58**] summaries various aspects of the work that has been done concerning TSP. See also a recent expository article [**52**].

Finding a shortest traveling salesman's route to pass n cities in such a way that each city is visited exactly once represents traditional formulation of TSP. It is assumed that nonnegative distances $c(i,j) = c_{ij}$ between the cities i, j $(1 \leq i < j \leq n)$ are given by the *distance matrix* $C = \|c_{ij}\|_1^n$ and also that traveling salesman starts his trip from arbitrary city. If the traveling salesman does not return to the starting city then minimal traversed route is called an *open route* or simply a *path*.

We introduce some graph theory interpretations. Standard graph theoretical notions will be used without corresponding definitions. Let G be a *weighted* digraph on n vertices. To each arc (i,j) of G a *length* (*weight* or *cost*) c_{ij} is assigned. We define $c_{ij} = +\infty$ if the arc (i,j) does not exist in G. The matrix $C = \|c_{ij}\|_1^n$ is called the *distance* (*weight* or *cost*) *matrix* of G. The *length* (*weight* or *cost*) $l(H)$ of a subgraph H of G is defined to be the sum of lengths of arcs of H. In particular, the length of a path is the sum of lengths of arcs from which it consists. A path (cycle, circuit) consisting from k arcs (edges) is called a *k-path* (*k-cycle*, *k-circuit*). A TSP is called *symmetric* if the weight matrix is symmetric. Otherwise it is called *asymmetric*. In a symmetric TSP the (weighted) digraph G is considered as an undirected (weighted) graph G.

A cycle (circuit, path) passing through each vertex of G exactly once is called a *Hamiltonian cycle* (*circuit*, *path*). Of course, a Hamiltonian cycle (circuit) is an n-cycle (n-circuit) and a Hamiltonian path is an $(n-1)$-path. A salesman tour (or *route*) is now a Hamiltonian cycle, circuit or path. TSP can be reformulated in the following way:

PROBLEM 1. *In a weighted digraph (graph) find a Hamiltonian cycle (circuit or path) of a minimal length.*

The basic model of TSP enables a variety of applications, if "city" is replaced by the most different kind of objects such as: production job (process), machine, etc. Also, distance can be interpreted as a cost or, more generally, a preference measure between objects [**46**]. Thus the solution of the TSP then produces a total ordering of the objects which minimizes the number of disagreements.

On the other hand, there is a possibility to include additional constraints caused by practical reasons: the number of salesmen, the number of cities visited by each salesman, structure of corresponding graphs, etc. (see Section 2 and [**58**]). Solutions of these combinatorial optimization problems can be explained as ordering of a set of objects according to various criteria.

Now we recall general ideas for finding the best solution in problems of combinatorial optimization via a standard *branch and bound procedure*. It will be applied in 3.4.1 to a multi-TSP and in 3.5 to a generalized TSP. In 3.2 it will be modified for finding k-best solutions. For the sake of simplicity, we restrict ourselves to the following optimization (say minimization) problem on weighted graphs (networks).

Let $\mathcal{A}$ be the set of all subgraphs of a graph G (with weights inherited from G). Let $\mathcal{F} \subseteq \mathcal{A}$ be the set of all subgraphs of G which posses some additional properties. The set $\mathcal{F}$ is specified in our problem and the subgraphs from $\mathcal{F}$ are called *feasible*. If we seek in $\mathcal{F}$ the elements with the weights as in our problem, then these requirements give rise to some typical NP-complete problems. In order to solve such a problem by a branch and bound technique, let furthermore $\mathcal{R}$ ($\mathcal{F} \subseteq \mathcal{R} \subseteq \mathcal{A}$) be a set of subgraphs for which there exists a polynomial time algorithm (say α) for finding the optimal element in $\mathcal{R}$. The set $\mathcal{R}$ corresponds to some relaxed variant of our problem (some feasibility conditions need not hold anymore).

To describe the algorithm (search procedure), we first introduce a *search tree* T as an auxiliary tool. T is as well, a *rooted tree* with the root at a vertex r; all other vertices are the *descendants* of r. If f is any vertex (father), then its out-neighbors (sons) are denoted by $s_1, \dots, s_n$. Each vertex, say f, corresponds to some subset $\mathcal{R}(f)$ of $\mathcal{R}$ and to a subproblem of the original problem (usually obtained by including and/or excluding some edges of G from the solution). The root r corresponds to the whole set $\mathcal{R}$. If f is a father and the solution of the relaxation task on the corresponding subproblems is not feasible and its length is smaller than the current lower bound (set at the beginning), then after branching at f by some branching rules (which "destroy" some "unfeasible details" in the solution of the relaxation task), the set $\mathcal{R}(f)$ is split into mutually disjoint subsets $\mathcal{R}(s_1), \dots, \mathcal{R}(s_n)$ yealding new subproblems and new vertices in the search tree T. By solving the relaxation problem at some tree vertex by the use of the algorithm α, we obtain a lower bound for a feasible solution at this vertex (if any). A global upper bound is provided at beginning by taking any feasible subgraph (usually found by some quick heuristic). The branch and bound algorithm terminates when all subproblems in the search tree T are exhausted.

As usual, we distinguish between a *problem* and an *instance* of a problem. Informally, in an *instance* we are given the "input data" and have enough information to obtain a solution; a *problem* is a collection of all instances. For more formal treatment of this notions see [69]. Under *case of a problem* we shall mean *subclass* of a problem e.g. a subset of the set of instances so that corresponding instances have a common property or, all of them are generated in a similar way. For example, in Section 3.4.2 a *well-solved* (*polynomial*) case of the TSP on bandwidth-limited graphs is considered.

We turn now to complexity questions.

As is well-known, the traveling salesman problem is NP-hard. Existing algorithms for NP-problems have an exponential complexity. As is known, algorithms complexities are estimated on the basis of the necessary elementary steps in the worst particular case (instance) of the considered problem. However, the experience shows that exponential algorithm for NP-problems behave as polynomial ones for a great number of instances. An exponential running time appears in a relatively small number of instances which will be referred to as *hard* instances.

Hard instances require a lot of computing time. Hence, we might have some interest to recognize such instances before the algorithm starts. If we establish on the basis of a certain index that an instance is hard, we could give up of finding the exact solution and to try to find a suboptimal (approximative) solution (by means of some quick heuristics). Here we assume that there exist an efficient (polynomial) algorithm for determining the index in question. Such indices, called *complexity indices*, are considered in Section 3.1.

2. VARIATIONS AND SPECIAL CASES

Under the term *non-standard traveling salesman problems* (non-standard TSP), which appears in the title of this paper, we do not understand a formally defined notion. Our intention is simply to collect under this title several variations and special cases of the (standard) TSP.

We formulate the *m traveling salesmen problem* (m-TSP or multi-TSP):

PROBLEM 2. *Given a weighted graph G and a positive integer m, find a spanning collection of m disjoint paths (salesmen's tours) with minimal length.*

Variations of this problem include limitations on the number of cities visited by each of the m traveling salesmen. In particular, it is interesting to consider the case when each salesman visits a constant number of cities. So we get the following problem:

PROBLEM 3. *Given a weighted digraph (graph) G on $n = km$ (k, m integers) vertices, determine m* disjoint *$(k-1)$-paths with a minimal total length.*

Considerable interest has been shown for well-solved cases of traveling salesman problem (TSP). According to [45], there are two broad categories of well-solved cases of TSP. In one category are the problems that are special because of restrictions on the matrix C of arc lengths; for example, C may be *upper triangular* or *circulant* matrix. In the second category are the problems in which TSP is to be solved over a network (graph) with particular structure, but with no restriction on the lengths of arcs.

Let G be a loopless directed graph (digraph) on vertices $1, 2, \ldots, n$. G is called a *bandwidth-limited* graph if there exists a positive integer $s (s < n - 1)$ such that for any arc (i, j) in G we have $|i - j| \leq s$. The smallest such integer s is called the *bandwidth* of G and is denoted by ω. Problem 3 considered on a bandwidth-limited digraph represents a well-solved case of TSP.

We shall formulate a generalized traveling salesman problem (GTSP) after giving some definitions.

Let $G_{s,n}$ ($s, n \geq 2$) be a weighted, complete, multipartite digraph (in the following — mp-digraph). 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 associated distance matrix C is a block matrix with s^2 square blocks of order n. Each block contains arcs lengths (distances) between vertices in the corresponding supervertices. Diagonal blocks contain only a sufficiently large positive constant e.g. $+\infty$ (connections within a supervertex).

The GTSP can be stated as

PROBLEM 4. *Find a minimum cost s-cycle which includes exactly one vertex from each supervertex in an mp-digraph.*

In almost all problems of combinatorial optimization, it can be of interest to find not only the best solution, but also some other (namely *suboptimal*) solutions. Usually, to specify them, we use the following criteria:

1. metric criterion (lengths of all other solutions differ from the length of the optimal one at most by a given number, say ϵ);
2. ranking criterion (a given number, say k, of best solutions is to be found; they are usually refered as k-best solutions).

We shall restrict ourselves (without loss of generality) to the following, still very general, problem.

PROBLEM 5. *Let a weighted graph G and a set $\mathcal{F}$ of its (weighted) subgraphs be given. Find all subgraphs H from $\mathcal{F}$ whose weights satisfy one of the criteria given by* 1 *and* 2.

3. A SURVEY OF RESULTS

We shall present several non-standard traveling salesman problems. The non-standardness is caused by topological limitations (TSP is to be solved over a network (graph or digraph) with particular structure) or by modification of the objective function (e.g. multi-TSP, k-best solutions etc.).

Complexity indices for TSP, by which we can, under some conditions, distinguish between "easy" and "hard" instances, are described in Section 3.1.

An algorithm for finding suboptimal solutions for TSP (see Problem 5) is presented in Section 3.2.

Section 3.3 contains a treatment of a TSP on digraphs with chained structure.

A multiple TSP with limitations on the number of cities each salesman should pass (i.e. Problem 3 from Section 2) is treated in 3.4. General case is considered in 3.4.1 while Section 3.4.2. is devoted to a multi-TSP in which the cost matrix has a limited bandwidth.

A generalized TSP on a complete multipartite digraph is considered in Section 3.5. This generalization simultaneously combines the decision of vertex selection and vertex sequencing. The generalized model assumes the vertices have been grouped into disjoint *vertex sets* which we shall call *supervertices*. The Generalized Traveling Salesman Problem (GTSP) is then to find a minimum cost cycle which includes exactly one vertex from each vertex set i.e. supervertex. It is *generalized* because TSP is a special case of GTSP; it is GTSP with vertex sets of cardinality one.

We report in 3.6, following **[17]**, on the implementation of a programming package, called TSP-SOLVER, for the traveling salesman problem. Various variants of TSP can be treated by TSP-SOLVER: both symmetric and asymmetric cases, one- or multiple-TSP, one or first k-best solutions, bandwidth limited distance matrix and others special cases, algorithms and heuristics. The system is user-friendly and offers the user, among other things, some possibilities to intervene during the solving a problem.

We hope to be able to complete soon the book **[23]** in which we shall present non-standard TSP with more detail.

3.1. Complexity Indices for TSP

This section contains no real results; it describes intuitively based ideas and numerical results partly justifying them.

There are no theoretical results described in the literature which would indicate the existence of efficient complexity indices for a particular instance of NP-problems, in spite of the fact that this type of problems is of an obvious practical importance. More generally, we do not see how a theory of complexity indices for instances of NP-problems can be based on known results. However, discussion on easy and hard cases (sets of instances) can be found, for example, in **[3]**, **[39]**, **[49]**, **[58]**.

The idea on complexity indices has been initiated in **[60]** in relation to the traveling salesman problem. The indices offered have been intuitively justified and their validity supported by some experimental results. After that in a series of papers **[19]**, **[20]**, **[21]**, **[18]** the problem of the complexity indices has been studied and it is partially theoretically justified or experimentally verified. The largest eigenvalue of the adjacency matrix of a minimal spanning tree has been introduced in **[19]** as a complexity index for the traveling salesman problem and its validity supported by some results from the theory of graph spectra **[24]**. Complexity indices based on the assignment problem are considered in **[21]**. The most comprehensive treatment of this research is given in **[14]** and **[32]**. The following text surveys shortly these works.

In solving TSP one usually uses one of the numerous variants of the branch and bound algorithm **[58]**. Principally speaking, the most known relaxations used in branch and bound algorithms for the traveling salesman problem are the task of finding a minimal spanning tree and the assignment problem. In the first one we solve a variant of traveling salesman problem in which the length of a Hamiltonian path is optimized. Actual running time necessary for a particular problem to be solved depends, of course, on the algorithm chosen. Hence, for each variant of branch and bound method a complexity or even more indices can be introduced.

To illustrate best our main idea on complexity indices we have selected the minimal spanning tree relaxation in spite of the fact that other relaxations perform better. We also bound ourselves to the symmetric case, i.e. we assume that we are given a complete undirected graph without loops with weights (lengths) defined on edges. A minimal spanning tree of the graph is a spanning subgraph of G which is a tree and in which the sum of weights of edges is minimal.

If a minimal spanning tree is a path, it represents also a solution to TSP. However, a path is also a tree with a minimal branching extent (in an intuitive sense). The main idea of [60], further developed in [19], is based on the expectation that branch and bound algorithm will run *the longer the more minimal spanning tree deviates from a path*, i.e. the greater "branching extent" it has. Accordingly, any graph invariant characterizing well the "branching extent" in an intuitive sense, can be considered as a complexity index for the traveling salesman problem.

In [60], [19] the following invariants have been considered:

D the number of vertices of degree 2 in the minimal spanning tree;

λ_1 the largest eigenvalue of the adjacency matrix of the minimal spanning tree.

The quantity D is maximal ($D = n - 2$, where n is the number of vertices) if the tree reduces to a path, but it attains its minimal value $D = 0$ on a great number of trees. The largest eigenvalue λ_1 reflects more precisely the branching extent of a tree. Given n the number of vertices of a tree, the quantity λ_1 varies between $2\cos\frac{\pi}{n+1}$ and $\sqrt{n-1}$, both bounds being attained on exactly one tree (a path and a star, respectively). Since the path P_n has the least branching extent in the intuitive sense and the star $K_{1,n-1}$ has the maximal one, the quantity λ_1 has at least a good property that it characterizes extremal trees in the above sense. Any invariant which is considered as a branching extent parameter should fulfill this criterion.

Complexity index D is based on vertex degrees while the index λ_1 is based on the spectrum of the minimal spanning tree. These two kinds of indices can be related as shown in [20].

As already said, there are no strict theoretical results described in the literature supporting the idea that the discussed invariants really can serve as complexity indices for the TSP. Therefore some experiments using a computer have been undertaken [20], [14], [21], [18].

A number of instances of the TSP have been generated by means of a random generator. For each instance we have computed the considered indices and the number N of the solved relaxation tasks when running the branch and bound algorithm.

The input graphs had a symmetric weight matrix. Weights were generating using a uniform distribution in the interval (0,1). Partial results are given below in the form of tables. A table is given for each group of graphs with a fixed number of vertices. The first and the second data column give the linear correlation coefficient between mentioned indices and quantities N and $\log N$. Finally, the Spearman rank correlation coefficient is given in the third column.

Table 1: 200 graphs on 8 vertices

	N	$\log N$	N (Spearman)
D	0.361	0.605	0.500
λ_1	0.433	0.598	0.514
S_4	0.501	0.612	0.536

Table 2: 100 graphs on 12 vertices

	N	$\log N$	N (Spearman)
D	0.293	0.529	0.439
λ_1	0.378	0.543	0.502
S_4	0.469	0.587	0.5453

The presented results show that correlation coefficients, generally speaking, decrease when the number of n vertices increase. This is to be expected since the information about the problem, contained in a minimal spanning tree, obviously decreases when n increases. (A complete graph on n vertices contains $\frac{1}{2}n(n-1)$ edges while spanning tree only $n-1$ edges; hence the proportion of the edges in the tree compared with the total number of edges is $2/n$). Therefore more sophisticated complexity indices should be introduced. A reasonable idea would be to study several spanning subgraphs consisting of the "short" edges of the input graph and to use invariants of such subgraphs as complexity indices [**14**], [**18**].

Let G be a weighted graph on n vertices with m edges. Let edges $e_1, e_2, \dots, e_m$ of G be ordered by non-decreasing weights. Let G_k be the graph having the same vertices as G and edges $e_1, e_2, \dots, e_k$ ($k = 0, 1, \dots, m$). Let P be a graph property. A *P-critical spanning subgraph* of G is the first graph in the sequence $G_0, G_1, \dots, G_m$ having property P. P-critical spanning subgraphs of G for various properties P are called *short edge subgraphs* of G.

Consider the following spanning subgraphs of G:

H_1 containing $n-1$ shortest edges;

H_2 critical connected;

H_3 critical for property of having at most 2 vertices of degree 1;

H_4 critical for property of having all vertex degree at least 2;

H_5 critical 2-connected;

H_6 critical Hamiltonian.

Intuitively, one can expect that short edge subgraphs contain important information about the complexity of a TSP task. Generally speaking, graph theoretical invariants of short edge subgraphs can serve as complexity indices. Short edge subgraphs could be considered as weighted graphs in which case the weights could be used to define some subgraphs and than again some invariants of these subgraphs could be examined.

Note that the minimal spanning tree is not a short edge subgraph in general case.

P-critical spanning subgraphs could serve for forming complexity indices for TSP if the property P can be checked by a polynomial algorithm. Namely, we gradually add short edges and keep checking property P until it appears. Since, checking whether a graph is Hamiltonian is an NP-problem short edge subgraph H_6 seems to be not suitable for our purposes.

It is expected, in an intuitive sense, that every "measure of size" of short edges subgraphs H_1–H_5 can serve as complexity index of a particular instance of the TSP. It can simply be the number of edges in the considered subgraph. As matter of fact, every short edge subgraph can be interpreted as an approximation of the search space needed to find optimal solution of the TSP. Measure of size of that search space is obviously dependent on relaxation which is used in the branch and bound algorithm. In the variant with minimal spanning tree as the relaxation, it is reasonable to study the number of trees in short edges subgraphs as a complexity index.

Beside the number of edges and the number of trees the following invariants of short edges subgraphs are considered as complexity indices: diameter, the number of cuts points, the number of components and the number of vertices of degree 1 [**18**].

3.2. k-Best Solutions of TSP

In this section we describe an algorithm for finding k-best solutions of TSP (see Problem 5 in Section 2) following [**30**].

Our algorithm for finding best suboptimal solutions represents a modification of the branch and bound algorithm described in section 1. It consists of two phases. In Phase I our strategy is to branch the search tree until we find the best solution (global minimum). In the course of branching we split at each tree vertex the corresponding subspace of the search space into parts where the expected subgraph (obtained by algorithm α) is less non-feasible (Rule 1). The Phase II relies to the search tree from the former phase. We now use to branch at each tree vertex whose lower bound is minimal possible. To split the corresponding search space we have to choose between two different rules: if the solution of the relaxation problem is not feasible we branch as earlier (Rule 1); otherwise, we split the search space by destroying a feasible solution; e.g. by letting each of its edges to be forbidden in turn within subspaces (Rule 2). The termination of search is encountered after reaching one of criteria from section, or by exhausting the search space.

These algorithms have been implemented within the programming package TSP-SOLVER (see Section 3.6).

Depending on the TSP instance characterized by distance matrix (symmetric or asymmetric) and by tours (open or closed) we can have four types of TSP problems. With each possibility, we use the 3-optimal heuristic (see, for example, [**58**] for finding the initial upper bound and the following relaxations:

1. Minimal spanning tree (Prim's algorithm [**70**]) for a symmetric TSP with an open tour;
2. Minimal 1-tree (a modification of Prim's algorithm [**58**], p. 371) for a symmetric TSP with a closed tour;
3. Minimal rooted tree (for the algorithm see [**46**]) for an asymmetric TSP with an open tour;

4. Assignment problem (see [6]) for an asymmetric TSP with a closed tour;

Implemented programs have shown a satisfactory performances for the TSP instances up to 20 vertices.

For other approaches to the problem of finding k-best solution in various problems see [8], [31], [47], [57], [59], [65].

3.3. A Chained TSP

Let G be a digraph whose adjacency matrix has the form

$$A = \begin{Vmatrix} A_1 & B_1 & 0 & \dots & 0 \\ 0 & A_2 & B_2 & & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & & B_{m-1} \\ 0 & 0 & 0 & & A_m \end{Vmatrix},$$

where $A_1, A_2, \dots, A_m$ are square blocks of order n. Subgraphs of G with adjacency matrices $A_1, A_2, \dots, A_m$ are denoted by $G_1, G_2, \dots, G_m$ respectively. Digraph G is called a *chained* digraph. If arcs of G carry some weights the corresponding TSP is called a *chained* TSP.

We can form an auxiliary weighted digraph F with weights on both vertices and arcs in the following way. The vertex set of F is union of disjoints sets $X_1, X_2, \dots, X_m$. Elements (vertices of F) from X_i are in a biunique correspondence with Hamiltonian paths of the digraph G_i $(i = 1, 2, \dots, m)$. The weight of a vertex from X_i is the length of the corresponding Hamiltonian path in G_i. For any $i = 1, 2, \dots, m-1$ we have arcs goring from each vertex from X_i to each vertex of X_{i+1}. The weight of the arc between $x \in X_i$ and $y \in X_{i+1}$ is equal to the (p, q)-entry of B_i if p is the ending vertex of the Hamiltonian path in G_i which corresponds to x and q is the starting vertex of the Hamiltonian path in G_{i+1} which corresponds to y.

A chained TSP is reduced in [61], [62] to the problem of finding a shortest path in F which starts in one of vertices from X_1 and terminates in one of vertices of X_m where the length of path is the sum of weights of all vertices and arcs from the path. Some efficient procedures for finding such a path and k-best solution are given in [61], [62] as well.

A dynamic programming algorithm for the chained TSP is given in [15].

3.4. A Special Multi-TSP

We shall consider Problem 3 from Section 2, namely

Problem: *Given a weighted digraph (graph) G on $n = km$ (k, m integers) vertices, determine m disjoint $(k-1)$-paths with a minimal total length.*

Note that Problem 3 is quite different from Problem 2. Problem 2 is usually solved by a transformation which reduces it to the ordinary TSP [2]. This transformation cannot be applied successfully to Problem 3.

We describe in 3.4.1 some branch and bound algorithms for this problem following the report **[13]**. Note that in **[34]** our problem has been formulated in terms of integer programming.

Other relevant references are **[1]**, **[7]**, **[41]**, **[42]**, **[43]**, **[50]**, **[55]**, **[71]**, **[73]**.

In 3.4.2 we shall consider our problem on a digraph with limited bandwidth.

3.4.1. *General Case*

Consider first the case when the weight matrix of G is symmetric.

DEFINITION 3.1. A graph whose components are trees is called a forest. A forest with s edges is called an s-forest.

The solution of our problem is a collection of m disjoint $(k-1)$-paths, hence an s-forest with $s = m(k-1)$. Therefore we accept the problem of finding a minimal s-forest as the relaxation task in our branch and bound algorithm. We propose the following algorithm for finding a minimal s-forest.

ALGORITHM. *Starting from the 0-forest, chose for $i = 0, 1, \dots, s-1$ a shortest edge transforming an i-forest into an $(i+1)$-forest. The obtained s-forest is a minimal s-forest.*

The correctness of the algorithm is verified by the following lemma.

LEMMA 3.1. *Let I be a spanning i-forest $(i = 0, 1, \dots, s-1)$ in a graph G which is a subforest of a minimal s-forest. Let u be a shortest edge which transforms the i-forest I into an $(i+1)$-forest. Then there exist a minimal s-forest which contains the i-forest I and the edge u.*

Proof. Suppose at the contrary, that no minimal s-forest contains the i-forest I and the edge u. Consider a minimal s-forest S.

The graph $S + u$ is either an $(s+1)$-forest or contains a circuit C. $S + u$ contains at least one edge v $(v \neq u)$ which does not belong to I but belongs to C if C exists. If C does not contain such an edge v, all edges of C belongs to I and the edge u would have not transformed I into an $(i+1)$-forest.

Now the graph $S' = S + u - v$ is again an s-forest whose length $l(S')$ is not greater than the length $l(S)$ of S. The case $l(S') < l(S)$ would contradict the fact that S is a minimal s-forest. The other case $l(S') = l(S)$ means that S' is a minimal s-forest containing I and u, thus contradicting the assumption at the beginning of the proof.

This completes the proof. □

Since the 0-forest is contained in any minimal s-forest, we conclude using Lemma 3.1 that the Algorithm gives a minimal i-forest for any $i = 1, 2, \dots, s$. For $s = n - 1$ we get a minimal spanning tree. In this way the Algorithm is a generalization of the Kruskal's minimal spanning tree algorithm [**51**], [**58**].

Consider the solution of a relaxation task during the work of our branch and bound algorithm. If it is not a collection of m disjoint $(k-1)$-paths and its length is smaller than the current upper bound we apply the following branching rules to produce new subproblems:

1. we "destroy" a vertex of degree greater then 2, if it exists; otherwiseRca
   we "destroy" a path with a maximal number of edges. 2.

REMARK. If the branching rule consists only from 1, our branch and bound algorithm would solve m-TSP without limitations on the numberof cities visited by each salesman (Problem 2).

Now we consider m-TSP with an asymmetric weight matrix. We start with some definitions.

DEFINITION 3.2. A tree in which each edge is directed is called a directed tree.

DEFINITION 3.3. A directed tree is called a rooted directed tree if the following conditions are fulfilled:

1. There is a vertex x with no incoming arcs;
2. Exactly one arc terminates in each of other vertices.

Vertex x is called the root.

DEFINITION 3.4. A digraph whose week components are rooted oriented trees is called an arborescence. An arborescence with s arcs is called an s-arborescence.

In solving m-TSP with an asymmetric matrix we can use a branch and bound algorithm in which the relaxation task is the task of finding a minimal s-arborescence where $s = m(k-1)$. An algorithm for finding a minimal s-arborescence has been described in [**16**] but we do not reproduce it here.

REMARK. An algorithm for finding a minimal arborescence (without specification of the number of arcs in it) in a weighted digraph with possibly negative arc weights was developed by J. Edmonds [**40**], [**63**]. This algorithm cannot be directly applied to problem of finding a minimal s-arborescence. However, it was a starting point in developing our algorithm.

Branching rules are analogous to those ones in the symmetric case:

1. we "destroy" a vertex of out-degree greater than 1, if it exists; otherwise
2. we "destroy" a path with a maximal number of arcs.

Following [**12**] we describe the role of the assignment problem in solving m-TSP.

In order to apply the assignment problem as a relaxation task to m-TSP we perform a transformation to the original weighted digraph G.

We form a new weighted digraph H by adding m new vertices to digraph G. New vertices are joined by arcs to all vertices from G in both directions. Weights of all new arcs are mutually equal, otherwise arbitrary.

A solution of m-TSP in G can be extended to a linear factor in H consisting of m $(k+1)$-cycles each containing exactly one of m new vertices. Such cycles are called *feasible* and any other cycle is called *unfeasible*.

A branch and bound algorithm for m-TSP on G acts on digraph H. The relaxation task is the assignment problem and the corresponding algorithm from **[6]** can been used. Branching rules are formulated so that unfeasible cycles are "destroyed".

At the end of this section we mention same heuristics.

In **[34]** a modified furthest insertion heuristic, known for an ordinary TSP, has been proposed for m-TSP. Many heuristics for TSP could be extended to m-TSP. Possible extensions we classify within the following two groups of heuristics according to **[12]** and **[22]**.

1. Using any heuristic for the ordinary TSP we determine a circuit (or cycle) and split it by deleting m edges (arcs) into m paths in such a way that we obtain a shortest possible solution.
2. Using any heuristic for the ordinary TSP by which a path is gradually augmented we construct a $(k-1)$-path. Then we delete the vertices from the so constructed path and apply the above step to the remaining digraph until we construct m $(k-1)$-paths.

3.4.2. *Bandwidth limited TSP*

Let G be a bandwidth-limited digraph (for the definition see Section 2) on N vertices. Following **[13]** and **[27]** an algorithm for solving Problem 3 on G will be described.

In **[64]**, **[72]** (see also **[45]**) a polynomial algorithm for symmetric TSP on bandwidth limited graph is described. It is based upon dynamic programming approach. Our algorithm extends the basic ideas of this algorithm for an asymmetric TSP and generalizes it for m $(m \geq 1)$ traveling salesmen.

Let vertices of digraph G be represented as in Fig. 1 for a fixed j $(\omega < j \leq N)$.

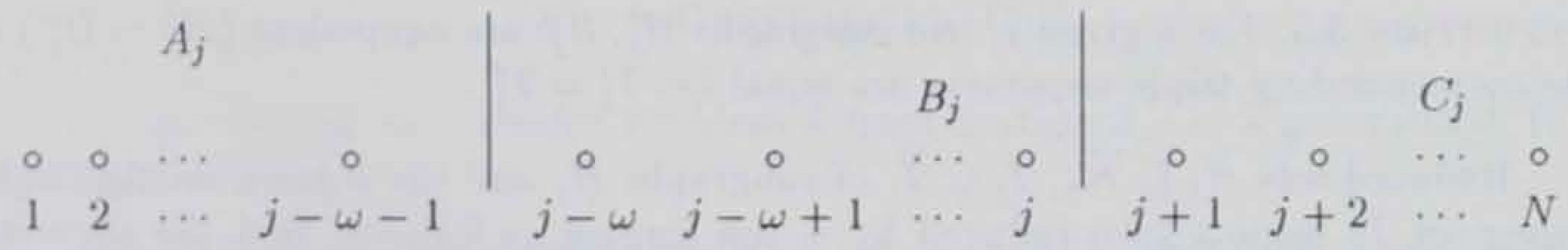

Fig. 1

Let us define the sets:

$$A_j = \{1, 2, \ldots, j-\omega-1\},$$
$$B_j = \{j-\omega, j-\omega+1, \ldots, j\},$$
$$C_j = \{j+1, j+2, \ldots, N\}.$$

The algorithm is based on the fact that, due to the bandwidth ω of the digraph G, there are no arcs from vertices of A_j to the vertices of C_j.

Consider m disjoint paths in digraph G. Let H be a digraph which has the same vertices as G and only arcs from paths mentioned. Let H_j be the subgraph of digraph H induced by the set $A_j \cup B_j$. Digraphs H and H_j contain four types of vertices with respect to the values of vertex indegrees d^- and outdegrees d^+:

1. isolated vertices: $d^- = 0$, $d^+ = 0$ (type 1),
2. starting vertices: $d^- = 0$, $d^+ = 1$ (type 2),
3. terminal vertices: $d^- = 1$, $d^+ = 0$ (type 3),
4. internal vertices: $d^- = 1$, $d^+ = 1$ (type 4).

Degrees of vertices in A_j are the same in H_j and H since in both G and H arcs between A_j and C_j do not exist.

Let us add vertex $j+1$ to H_j, so that the digraph H_{j+1} is obtained. Vertex $j+1$ is adjacent with 0, 1 or 2 vertices in B_j. Various possibilities of joining vertex $j+1$ and a vertex $b \in B_j$ that yield a legitimate subgraph H_{j+1}, depend not only on in-(out-)degree of vertex b, but also on paths in H_j containing vertices in B_j.

The set B_j contains $\omega + 1$ elements. Let x_i^j be the i-th vertex in B_j (x_i^j has an absolute position $j - \omega - 1 + i$ in G). For a vertex $x_i^j \in B_j$, let $u_i^j \in \{1,2,3,4\}$ be its vertex type. Let $t_i^j = (u_i^j, v_i^j, w_i^j)$ be an assigned triple of nonnegative integers defined as follows:

1. if x_i^j is isolated (internal) vertex, then $u_i^j = 1 (u_i^j = 4)$, and $v_i^j = w_i^j = 0$.
2. if x_i^j is starting (terminal) vertex, the $u_i^j = 2 (u_i^j = 3)$; v_i^j is the number of arcs in a path with one endvertex x_i^j, and w_i^j is position (ordinal number) of the other endvertex of the corresponding path. (If the other end of the path is in A_j, we have $w_i^j = 0$).

Triple sequence $T_j = (t_1^j, t_2^j, \ldots, t_{\omega+1}^j)$ describes the set B_j, hence the digraph H_j in the extent necessary for further analysis.

Let $\overline{\mathcal{H}}_j$ be the set of all subgraphs H_j and $\overline{\mathcal{T}}_j$ be the set if corresponding triple sequences T_j. Let us define an equivalence relation on the set $\overline{\mathcal{H}}_j$.

DEFINITION 3.5. For a given j, two subgraphs H_j', H_j'' are equivalent ($H_j' \sim H_j''$) if the corresponding triple sequences are equal i.e. $T_j' = T_j''$.

Reduced sets $H_j \subseteq \overline{\mathcal{H}}_j$, $\mathcal{T}_j \subseteq \overline{\mathcal{T}}_j$ of subgraphs H_j and the corresponding triple sequences T_j respectively, induced by $\sim$ are formed as follows: find the shortest subgraph $\mathcal{H}_j \in \overline{\mathcal{H}}_j$ in each equivalence class, include it in $\mathcal{H}_j$, and include at the same time its 'description' i.e. corresponding $T_j \in \mathcal{T}_j$.

The following *algorithm* $\mathcal{A}$ solves the given problem (M-TSP): (All subgraphs H_j are represented by corresponding triple sequences T_j).

1. step *Initialization*: Determine initial set $\mathcal{H}_{\omega+1}$ (see Section 4 in [**27**]).

2. step: **For** $j = \omega + 1, \omega + 2, \ldots, N$ **do**:

   Add vertex $j + 1$ to every $H_j \in \mathcal{H}_j$ thus obtaining $H_{j+1} \in \overline{\mathcal{H}}_{j+1}$. (For various possibilities of adding vertex $j + 1$ that yield legitimate subgraphs H_{j+1} see Section 3 in [27]).

   Sort triple sequences $T_{j+1} \in \overline{\mathcal{T}}_{j+1}$ lexicographically. Sets of mutually equal triple sequences correspond to equivalence classes of subgraphs H_{j+1}. Keep the shortest subgraph H_{j+1} in each equivalence class, thus forming reduced sets $\mathcal{T}_{j+1}$ and $\mathcal{H}_{j+1}$.

3. step: In set $\mathcal{H}_N$ **find** the subgraph H_N of the shortest length. *Stop*: Digraph $H = H_N$ is an optimal solution.

It was proved in [27] that the described algorithm is polynomial of degree at most $\omega + 2$.

Note that standard branch and bound algorithms for TSP remain of an exponential complexity when applied to a bandwidth-limited graph [27].

However, a disadvantage of our polynomial algorithm when compared with the exponential branch and bound algorithm consists of the fact that it works a constant amount of time for all instances of our problem. Branch and bound algorithms are sensitive on the distribution of arc lengths so that there are problem instances which are quickly solved while for some "hard" instances an "exponential" time is necessary.

Experiments on computer with the above algorithm implemented have shown a bad performance. The number of generated triple sequences is enormous for quite modest values of N and ω. Big memory requirements have implied the usage of virtual memory and this contributed to higher execution times.

Therefore we have developed some procedures for reducing the number of generated triple sequences. The basic idea for these procedures consists in the following. Some subgraphs H_j can have so big length that they do not have any chance to be extended to an optimal solutions. Such subgraphs will not be extended any more when putting next vertex into consideration [27].

3.5. A Generalized TSP

In this section we consider Problem 4 from Section 2, i.e. a generalized TSP (GTSP).

Two branch and bound algorithms for solving GTSP have been almost simultaneously and independently published in [37] (see also [38]) and [68]. We shall shortly describe these two algorithms.

We note that an early attempt in solving the GTSP was based on a transformation by which GTSP is reduced to the TSP [74]. Namely, starting with given mp-digraph G a new digraph G' is constructed in which every Hamiltonian cycle corresponds to exactly one s-cycle and vice versa. More precisely, there exists a bijection between the set of all Hamiltonian cycles in G' and s-cycles in the mp-digraph G. Requirement that traveling salesman enters and leaves supervertex at the same vertex is also assured by the transformation.

This transformation increases the dimension of the problem sn times. Computational results on $VAX8800$ using this transformation and then solving the TSP (with one of the most efficient branch and bound algorithm for the asymmetric TSP [5]), show that the GTSP can be solved on small mp-digraphs with $s, n \leq 5$. The explanation is the following. The transformation is of such nature that the applied branch and bound algorithm for TSP (based on assignment problem as relaxation) forms two subcycles within each supervertex. Afterwards, subcycles "patching" gets into an exhaustive search.

Although computationally inferior this transformation helped to prove the expected fact that the GTSP is NP-complete [23].

In [37, 38] a branch and bound algorithm for the GTSP using minimal rooted directed tree as relaxation is presented (see Definition 3.3). This algorithm is referred to the "open" variant of GTSP i.e. finding minimum cost $(s-1)$-path which contains exactly one vertex from each supervertex.

Let $d^+(x)$ denote the out-degree of vertex x in a digraph. A path is a rooted directed tree with property that out-degree of all vertices is equal to 1, except for a single vertex t, called *terminal vertex* of the path, for which $d^+_G(t) = 0$.

Minimal rooted directed tree can be determined by Edmond's algorithm of complexity $O(n^2)$ [40]. In solving relaxation task the variant of Edmond's algorithm with a *fictive* equi-distanced vertex added to the input mp-digraph was used [46]. This variant enables to find a minimal rooted directed tree without prior specification of the root.

A characteristic of Edmond's algorithm is that in *forward* phase it finds a minimal rooted directed tree by successively including minimal entering arcs for each vertex i. In the case that a cycle is formed, all the vertices of the cycle are merged in a new vertex called *pseudo-vertex*. In the next iteration, the algorithm continues on the new digraph with reduced number of vertices: the formed pseudo-vertex and all the (remaining) vertices which are outside of the merged cycle. All the vertices in the new reduced digraph are treated equally.

At the initialization step of the algorithm this merging enables to transform supervertices into pseudo-vertices, and thus the dimension of digraph is reduced. So, task of finding a minimal rooted directed tree is reduced to a digraph with only s vertices.

In the second, *backward* phase, Edmond's algorithm is modified in such a way that expanding pseudo-vertices is done only to the level of supervertices (i.e. a pseudo-vertex which corresponds to a supervertex is not expanded).

For rooted trees two branching scheme are applied. The first one is applied when the rooted tree is not a path. In that case, branching rules implemented in any branch and bound algorithm for the standard TSP with rooted tree as relaxation can be used. The following branching rule is applied to the first supervertex I with

$d^+(I) = k$ ($k \geq 2$): successively including exactly one leaving arc while all other arcs were excluded from consideration. In this way $k + 1$ new subproblems in the search tree are generated. The last generated subproblem corresponds to the case when all leaving arcs from supervertex I are excluded.

In the case, when the rooted tree is a path, then supervertices (i.e. corresponding pseudo-vertices) are expanded in order to get into their structure, looking for a supervertex with discontinuity. If there is no discontinuity — in each supervertex arc enters and leaves the same vertex, this feasible solution is kept as candidate for the optimal one. In other case, three subproblems are generated by successively excluding exactly one arc which takes part in forming discontinuity. The last subproblem is obtained by excluding both arcs.

We note that depth-search variant of the branch and bound is used, in order to obtain as quickly as possible a feasible solution.

A different approach, based on a *Lagrangian relaxation* for the GTSP, is described in [68]. Namely, GTSP is modeled as an integer program in which constraints are partitioned in two groups. The "complicated constraints" are dropped and brought into the problem's objective function. This results in obtaining a Lagrangian relaxation for the GTSP. (For a general framework of Lagrangian relaxation see [66].)

The presented approach is a sequence of three separate procedures: determining good lower bounds by solving Lagrangian relaxation, arc/vertex elimination and branch and bound enumeration. Instead of optimal solving the Lagrangian relaxation using a direct method, the first procedure approximates lower bounds using an iterative subgradient algorithm [46], [66]. The subgradient approach compared to direct methods uses little storage. This procedure also includes a heuristic for determining upper bounds. These bounds are than used in the second procedure to identify and remove many nonoptimal arcs and vertices. With the problem well-bounded and reduced in size, the third procedure uses implicit enumeration to guarantee an optimal solution.

Other relevant details of both branch and bound algorithms are described in [23]. Also, the efficiency of these two algorithms is compared. It seems that larger problems can be solved with the approach proposed in [68].

3.6. TSP-SOLVER A PROGRAMMING PACKAGE FOR TSP

A programming package for traveling salesman problem (TSP) has been implemented at University of Belgrade, Faculty of Electrical Engineering in 1989 and 1990. The package is called TSP-SOLVER.

Programming package TSP-SOLVER is implemented on a *VAX* computer under operating system *VMS* and using programming language *FORTRAN*. A Form Management System (FMS) has been used to create menus for the communication with the user. There are about 500 subroutines linked into several executable task units (subsystems). Subsystems communicate each to other through files with data (*M*-files) on particular TSP instances.

Subsystems of the system TSP-SOLVER are GENER, SALES, STATIS and EXTER.

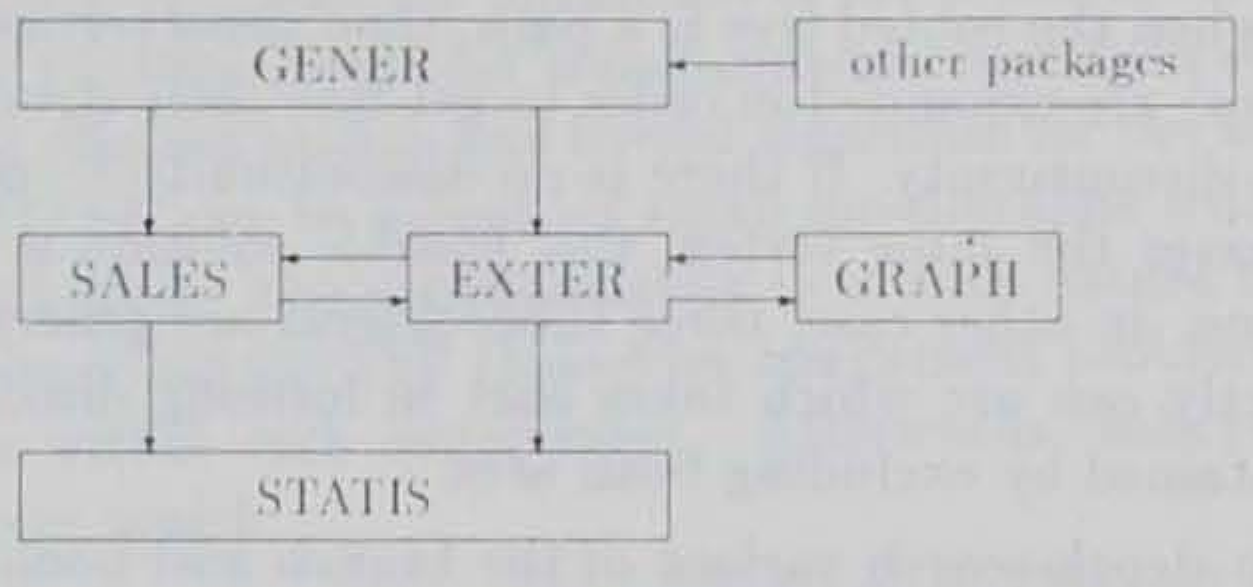

Fig. 2

GENER creates, modifies and verifies *M*-files. Creation of *M*-files makes use of random number generators. *M*-files created by other programs (outside TSP-SOLVER) are verified before treating by SALES.

SALES contains several algorithms and heuristics for TSP implemented. Input data are taken from one or more *M*-files. Output is directed again to *M*-files or/and other output devices.

STATIS performs statistical treatment of data in *M*-files including results of working of SALES.

EXTER is a subsystem to link the system TSP-SOLVER to the previously developed package GRAPH **[10]**, **[25]**, **[28]**, **[29]**. EXTER transforms data from TSP-SOLVER into formats used in GRAPH. Results of the working of GRAPH the subsystem EXTER can put in corresponding *M*-files. A user can also via subsystem EXTER inspect the actual work of SALES by interrupting a process, inspecting intermediary results, changing modes of the work, i.e. strategy of further work.

Information flow between subsystems is given in Fig. 2. All presented information exchanges are realized via *M*-files except for the linkage between EXTER and GRAPH where the medium is a special file used by GRAPH.

System TSP-SOLVER is in spirit similar to the system TRAVEL **[4]** but has much more algorithms and heuristics implemented. Experiments with TSP-SOLVER will be described in other papers. Here we describe subsystems in some detail.

A TSP is defined by a weighted graph represented by the corresponding distance matrix (DM). The following types of the distance matrix can be treated in package TSP-SOLVER.

1° asymmetric DM corresponding to the complete directed weighted graph;

2° symmetric DM corresponding to the complete undirected weighted graph;

3° band DM where there exists a positive integer w ($w < n-1$) such that $DM(i,j)$ is finite and non-negative for $|i-j| \leq w$. All other entries of the

DM are equal to $+\infty$. The value w is called the bandwidth;

4° the weighted graph has a chain structure (chain-like DM). Such a graph consists of an ordered sequence of complete directed subgraphs in which each node of a subgraph is connected by an arc to each node of the immediately following subgraph;

5° DM is euclidean, i.e. satisfies the triangular inequalities $DM(i,j)+DM(j,k) \geq DM(i,k)$ for all i,j,k.

The following algorithms and heuristics have been implemented.

1° symmetric TSP: branch and bound algorithm by Volgenant and Jonker [**75**] and two others based on minimal spanning tree and 1-tree as relaxations; 3-optimal and nearest neighbor heuristics.

2° asymmetric TSP: branch and bound algorithm by Carpaneto and Toth [**5**] and two others based on minimal assignation [**6**] and oriented rooted tree [**46**] relaxations; 3-optimal and nearest neighbor heuristics.

3° branch and bound algorithms for a given number of best suboptimal solutions for cases 1° and 2° [**11**], [**12**].

4° multiple TSP: symmetric and asymmetric; branch and bound algorithms with minimal forests [**12**] and oriented rooted forests [**12**] as relaxations, a heuristic from [**22**].

5° asymmetric TSP with band distance matrix: one ore more salesmen, a polynomial algorithm [**13**].

6° chain-like TSP, asymmetric, a dynamic programming algorithm [**15**] and an algorithm using suboptimal solutions for subgraphs [**61**], [**62**].

Subsystem EXTER is an interface between TSP-SOLVER and GRAPH.

Some (non-weighted) graphs can be created in EXTER from TSP instances treated with TSP-SOLVER (e.g. minimal spanning trees, minimal assignation graphs and other subgraphs of "short" edges, etc). These graphs can be further treated by the system GRAPH, part ALGOR for graph theoretic algorithms (e.g. we can find vertex-degrees, diameter, eigenvalues etc.). These results can be sent to the corresponding M-files through EXTER and further elaborated by STATIS.

Interesting "short edge" graphs can be also created from intermediary distance matrices during the work of a branch and bound algorithm. Communication between EXTER and SALES is realized via a global sherable *common* area.

Facilities described in this section enable, among other things, the study of complexity indices for TSP [**14**], [**19**], [**20**], [**21**], [**32**] and were, in fact, motivated by such investigations.

REFERENCES

[**1**] A. I. Ali and J. L. Kennington, *The asymmetric m-traveling salesman problem: a duality based branch-and-bound algorithm*, Discrete Appl. Math. **13** (2–3) (1986), 259–276.

[2] M. Bellmore and S. Hong, *Transformation of the multisalesmen problem to the standard traveling salesman problem*, J. Assoc. Comput. Mach. **21** (1974), 500–504.

[3] R. V. Book and D. Du, *The existence and density of generalized complexity cores*, J. Assoc. Comput. Mach. **34** (3) (1987), 718–730.

[4] S. C. Boyd and G. C. W. R. Pulleyblank, *TRAVEL — an interactive traveling salesman problem package for IBM-personal computer*, Oper. Res. Letters **6** (3) (1987), 141–143.

[5] G. Carpaneto and P. Toth, *Some new branching and bounding criteria for the asymmetric traveling salesman problem*, Management Sci. **26** (1980), 736–743.

[6] G. Carpaneto and P. Toth, *Solution of the assignment problem*, ACM Trans. Math. Software **6** (1980), 104–111.

[7] N. Christofides and S. Eilon, *An algorithm for vehicle-dispatching problem*, Oper. Res. Quart. **20** (1969), 309–318.

[8] C. R. Chegireddy and H. W. Hamache, *Algorithms for finding k-best perfect matching*, Discrete Appl. Math. **18** (1987), 155–165.

[9] D. Cvetković, *A project for using computers in further development of graph theory*, in: *The Theory and Applications of Graphs* (Proc. 4th Internat. Conf. Theory and Appl. of Graphs, Kalamazoo 1980), (G. Chartrand, Y. Alavi, D. L. Goldsmith, L. Lesniak-Foster, and D. R. Lick, eds.), 285–286, John Wiley & Sons, New York – Chichester – Brisbane – Toronto – Singapore, 1981.

[10] D. Cvetković, *Further experiences in computer aided research in graph theory*, in: *Graphs, Hypergraphs and Applications*, Proc. Conf. Graph Theory held in Eyba, October 1984, 27–30, ed. H. Sachs, Teubner, Leipzig, 1985.

[11] D. Cvetković, *Finding k ($k > 1$) shortest traveling salesman tours in a complete digraph with an asymmetric weight matrix*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1985.

[12] D. Cvetković, *M traveling salesmen in a complete digraph with an asymmetric weight matrix*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1986, 1–30.

[13] D. Cvetković, *M traveling salesmen in a complete digraph with an asymmetric band weight matrix*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1986.

[14] D. Cvetković, *Adaptive solving of the traveling salesman problem*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1987, 1–48.

[15] D. Cvetković, *Connection between global and local solutions of the traveling salesmen problem*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1987, 1–11.

[16] D. Cvetković, *Finding a shortest rooted forest*, (Serbo-Croatian), unpublished report. Faculty of Electrical Engineering, Belgrade, 1987, 1–20.

[17] D. Cvetković, M. Čangalović, V. Dimitrijević, L. Kraus, M. Milosavljević, and S. Simić, *TSP-SOLVER — a programing package for the traveling salesman problem*, Univ. Beograd. Publ. Elektrotehn. Fak. Ser. Mat. **1** (1990), 41–47.

[18] D. Cvetković and V. Dimitrijević, *Short edge subgrpahs and complexity indices for the traveling salesman problem* (Serbo-Croatian), in: SYM-OP-IS '92, Beograd 13–16. Oct. 1992, ed. M. Vujošević, Beograd, 1992, 41–44.

[19] D. Cvetković, V. Dimitrijević, and M. Milosavljević, *A spectral traveling salesman problem complexity index* (Serbo-Croatian), in: *Zbornik radova XI Bosansko-Hercegovačkog simpozijuma iz informatike, Knjiga 2*, 276–1 – 276–5, Sarajevo-Jahorina '87, 1987.

[20] D. Cvetković, V. Dimitrijević, and M. Milosavljević, *Traveling salesman problem complexity indices based on minimal spanning trees*, in: *Graph Theory*, Proc. Eighth Yugoslav Seminar on Graph theory, Novi Sad, April 17–18, 1987, (R. Tošić, D. Acketa, V. Petrović, and R. Doroslovački, eds.), 43–51, Univ. Novi Sad, Institute of Mathematics, 1988.

[21] D. Cvetković, V. Dimitrijević, and M. Milosavljević, *A linear factor complexity index for the exact solution of the traveling salesman problem* (Serbo-Croatian), in: *Zbornik radova XIV*

konferencije SYM-OP-IS '87, Herceg Novi 1987, 95–99, Institut za ekonomiku industrije, Beograd, 1987.

[22] D. Cvetković, V. Dimitrijević, and M. Milosavljević, *A class of algorithm for suboptimal solving of the m traveling salesmen problem* (Serbo-Croatian), in: *Zbornik radova XIV konferencije SYM-OP-IS '87*, Herceg Novi 1987, 101–106, Institut za ekonomiku industrije, Beograd, 1987.

[23] D. Cvetković, V. Dimitrijević, and M. Milosavljević, *Non-standard traveling salesman problems*, (in preparation).

[24] D. Cvetković, M. Doob, and H. Sachs, *Spectra of graphs — Theory and application*. Deutscher Verlag der Wissenschaften - Berlin, Academic Press - New York, 1980.

[25] D. Cvetković and L. Kraus, *GRAPH an expert system for the classification and extension of the knowledge in the field of graph theory*, User's manual. Faculty of Electrical Engineering, Belgrade, 1983.

[26] D. Cvetković, L. Kraus, and S. Simić, *Discussing graph theory with a computer I, Implementation of graph theoretic algorithms*, Univ. Beograd, Publ. Elektrotehn. Fak. Ser. Mat. Fiz. No. **716** - No. **734**, 100–104, 1981.

[27] D. Cvetković, M. Milosavljević, and V. Dimitrijević, *An algorithm for M asymmetric traveling salesman problem on a bandwidth-limited graph*, YUJOR **1** (1) (1991), 15–25.

[28] D. Cvetković and I. Pevac, *Man-machine theorem proving in graph theory*, Artificial Intelligence **35** (1988), 1–23.

[29] D. Cvetković, Z. Radosavljević and S. Simić, *Using expert programming system in investigation in graph theory* (Serbo-Croatian), in: *Zbornik radova XVI konferencije SYM-OP-IS '89*, Herceg Novi, 1989, 165–168, Institut za ekonomiku industrije, Beograd, 1989.

[30] D. Cvetković and S. Simić, *Best suboptimal solutions in combinatorial optimization problems* (Serbo-Croatian), in: *Zbornik radova XVIII konferencije SYM-OP-IS '91*, Herceg Novi, 8–11.X.1991, 103–105; ed. R. Stanojević and J. Vuleta, Herceg Novi, 1991.

[31] U. Derigs, *Some basic exchange properties in combinatorial optimization and their application to constructing the k-best solutions*, Report 83295. Universität Bonn, 1983.

[32] V. Dimitrijević, *Adaptive Decision in a Class of Pattern Recognition Systems* (Serbo-Croatian). Master's Thesis, Faculty of Electrical Engineering, Belgrade, 1988.

[33] V. Dimitrijević, L. Buturović, and M. Milosavljević, *On the dependence of the complexity estimation for the traveling salesman problem from the problem dimension* (Serbo-Croatian), in: *Zbornik radova XIII simpozijuma o informacionim tehnologijama*, 273–1 – 273–6, Sarajevo '89, 1989.

[34] V. Dimitrijević and M. Milosavljević, *M-traveling salesmen problem with constraint on the number of visited cities* (Serbo-Croatian), in: *Zbornik radova XIII konferencije SYM-OP-IS '86*, Herceg Novi, 1986, 721–726, Institut za ekonomiku industrije, Beograd, 1986.

[35] V. Dimitrijević and M. Milosavljević, *An algorithm for adaptive solving of the traveling salesman problem* (Serbo-Croatian), in: *Zbornik radova XXXIII konferencije ETAN*, Sarajevo, XII.187–XII.194, 1988.

[36] V. Dimitrijević and M. Milosavljević, *A complexity index for exact solving of the traveling salesman problem based on the estimation of the duality gap* (Serbo-Croatian), in: *Zbornik radova XXII simpozijuma o informacionim tehnologijama*, 1–2, Sarajevo '88, 1988.

[37] V. Dimitrijević, M. Marković, and M. Milosavljević, *A traveling salesman problem on the multipartite digraph* (Serbo-Croatian), in: *Zbornik radova XV simpozijuma o informacionim tehnologijama*, 202–1 – 202–8, Sarajevo '91, 1991.

[38] V. Dimitrijević, M. Marković, and M. Milosavljević, *The optimal solving of the traveling salesman problem on the multipartite digraph* (Serbo-Croatian), in: *Zbornik radova XVIII konferencije SYM-OP-IS '91*, Herceg Novi, 8–11.X.1991, 106–109, ed. R. Stanojević and J. Vuleta, Herceg Novi, 1991.

[39] D. Du and R. V. Book, *On inefficient special cases of NP-complete problems*, Theoret. Comput. Sci. **63** (3) (1989), 239–252.

[40] J. Edmonds, *Optimum branchings*, in: *Mathematics of Decision Sciences, Lectures in Appl. Math.*, (G. Dantzig and A. Veinott, eds.), 346–361, 1968.

[41] G. N. Frederickson, M. S. Hecht, and C. E. Kim, *Approximation algorithms for some routing problems*, SIAM. J. Comput. **7**, 178–193, 1978.

[42] A. M. Frieze, *An extension of Christofides heuristic in the k-person traveling salesman problem*, Discrete Appl. Math. **6** (1) (1983), 79–83.

[43] A. Garcia-Diaz, *A heuristic circulation-network approach to solve the multi-traveling salesman problem*, Networks **15** (4) (1985), 455–467.

[44] M. R. Garey and D. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*. San Francisco, W. H. Freeman Company Publishers, 1979.

[45] P. C. Gilmore, E. L. Lawler, and D. B. Shmoys, *Well-solved special cases*, in: *The traveling salesman problem*, (E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, eds.), 87–143, John Wiley & Sons, Chichester – New York – Brisbane – Toronto – Singapore, 1985.

[46] M. Gondran and M. Minoux, *Graphs and Algorithms*. John Wiley & Sons, Chichester – New York – Brisbane – Toronto – Singapore, 1984.

[47] H. W. Hamacher and M. Queyranne, *k best solutions to combinatorial optimization problems*, Annals of Oper. Res. **4**, 123–143.

[48] A. L. Henry-Labordere, *The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem*, RIRO **B-2** (1969), 43–49.

[49] C. Hoede, *Hard graphs for the maximum clique problem*, Discrete Math. **72** (1988), 175–179.

[50] S. Hong and M. W. Padberg, *A note on the symmetric multiple traveling salesman problem with fixed charges*, Oper. Res. **28** (1977), 871–874.

[51] J. B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proc. Amer. Math. Soc. **7** (1965), 48–50.

[52] G. Laporte, *The traveling salesman problem: An overview of exact and approximate algorithms*, European Journal of Operational Research **59** (1992), 231–247.

[53] G. Laporte, *The vehicle routing problem: An overview of exact and approximate algorithms*, European Journal of Operational Research **59** (1992), 345–358.

[54] G. Laporte, H. Mercure, and Y. Nobert, *Generalized traveling salesman problem through n sets of nodes: the asymmetrical case*, Discrete Appl. Math. **18** (1987), 185–197.

[55] G. Laporte and Y. Nobert, *A cutting plane algorithm for the m-traveling salesman problem*, J. Oper. Res. Soc. **31** (1980), 1017–1024.

[56] G. Laporte and Y. Nobert, *Generalized traveling salesman problem through n sets of nodes: an integer programming approach*, INFOR **21** (1983), 60–75.

[57] E. L. Lawler, *A procedure for computing k best solutions to discrete optimization problems and its application to the shortest path problem*, Management Sci. **18** (1972), 401–405.

[58] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, eds., *The Traveling Salesman Problem*. John Wiley & Sons, Chichester – New York – Brisbane – Toronto – Singapore, 1985.

[59] M. Leclerc and F. Rendl, *k-best constrained bases of a matroid*, ZOR **34** (2) (1990), 79–89.

[60] M. Milosavljević and V. Dimitrijević, *A complexity analysis concerning the exact solution of the traveling salesman problem* (Serbo-Croatian), in: *Zbornik radova X Bosansko-Hercegovačkog simpozijuma iz informatike*, 271–1 – 271–6, Sarajevo-Jahorina '86, 1986.

[61] M. Milosavljević and M. Obradović, *An algorithm for finding the shortest paths on a class of large dimensional graphs* (Serbo-Croatian), Naučno-tehnički pregled **35** (6) (1985), 3–6.

[62] M. Milosavljević and M. Obradović, *An algorithm for finding the k shortest paths on a class of large dimensional graphs* (Serbo-Croatian), in: *Zbornik radova X Bosansko-Hercegovačkog simpozijuma iz informatike*, 204–1 – 204–7, Sarajevo-Jahorina '86, 1986.

[63] E. Minieka, *Optimization algorithms for networks and graphs*. Marcel Dekker, Inc., New York – Basel, 1978.

[64] B. Monien and I. H. Sudborough, *Bandwidth constrained NP-complete problems*, in: *Proc. 13th Annual ACM Symp. Theory of Computing* (1981), 207–217.

[65] K. M. Murty, *An algorithm for ranking all the assignments in order of increasing cost*, Oper. Res. **16** (1968), 682–687.

[66] G. L. Nemhauser and A. L. Wolsey, *Integer and combinatorial optimization*. John Wiley & Sons, New York – Chichester – Brisbane – Toronto – Singapore, 1988.

[67] C. E. Noon and J. C. Bean, *An efficient transformation of the generalized traveling salesman problem*, Working Paper, Department of Management. University of Tennessee, Knoxville, 1989.

[68] C. E. Noon and J. C. Bean, *A Lagrangian based approach for the asymmetric generalized traveling salesman problem*, Operations Research **39** (4) (1991), 623–632.

[69] C. H. Papadimitrou and K. Steiglitz, *Combinatorial Optimization: Algorithm and Complexity*. Prentice-Hall, New Jersay, 1982.

[70] R. C. Prim, *Shortest connection networks and some generalizations*, Bell System Tech. J. **36** (1957), 1389–1401.

[71] M. R. Rao, *A note on multiple traveling salesman problem*, Oper. Res. **28** (1980), 628–632.

[72] H. D. Ratlif and A. S. Rosenthal, *Order picking in a rectangular warehouse: a solvable case of traveling salesman problem*, Oper. Res. **31** (1983), 507–521.

[73] R. Russell, *An effective heuristic for the m-tour TSP with some conditions*, Oper. Res. **25** (1977), 517–524.

[74] Z. Šarić, private communication 1990.

[75] T. Volgenant and R. Jonker, *Branch and bound algorithm for the symmetric traveling salesman problem based on 1-tree relaxation*, European Journal of Operational Research **9** (1982), 83–89.