

A Tabu Scatter Search Metaheuristic for the Arc Routing Problem

Peter Greistorfer ^{*,1}

*Karl-Franzens-Universität Graz*¹

Abstract

We consider a special routing problem which has a variety of practical applications. In a graph-theoretic context it is known as the Capacitated Chinese Postman Problem. Given an undirected network in which the demand is located on edges, the goal is to determine a least-cost schedule of routes. The route demands, which must not exceed the vehicle capacity, are serviced by a fleet of vehicles which is located at a single depot node. We present a metaheuristic based on a Tabu Search procedure that makes use of the Scatter Search paradigm. The computational results indicate that the algorithms proposed can keep up with other arc routing heuristics.

Key words: Chinese Postman Problem, Capacitated Arc Routing, Tabu Search, Scatter Search, solution combination method, hybrid metaheuristic

* Corresponding author:

Email address: peter.greistorfer@kfunigraz.ac.at (Peter Greistorfer).

URL: <http://www.kfunigraz.ac.at/ifwww/pg/home.htm> (Peter Greistorfer).

¹ Institut für Industrie und Fertigungswirtschaft, Universitätstraße 15/G2, 8010 Graz, Austria, Fax. ++43-316-3809555

1 Introduction

The purpose of this paper is the development of a heuristic solution strategy for a special logistical problem which is known as the *Capacitated Chinese Postman Problem* (CCPP). The CCPP can be described as follows. We assume an undirected graph, in which nodes represent junctions and edges represent streets with non-negative travel costs and positive demands. There is an unlimited number of homogeneous vehicles located at a single depot node. The goal is to find a schedule of routes each of which starts and ends at the depot node. This should be done in a way to minimize the total travel cost over all routes, which also includes the cost of deadheading, i.e. the traversing of already serviced edges in order to get to regions which have not been serviced yet. The schedule must satisfy the restrictions that every edge is serviced by exactly one vehicle and that no route demand exceeds a given vehicle capacity. The CCPP is known to be an instance of the general class of Capacitated Arc Routing Problems (CARPs), where a service need not be required for every edge, and it is the edge-oriented variant of the well-known node-oriented (standard) Vehicle Routing Problem in which demands are located at nodes.

The CCPP belongs to a class of network optimization problems which have high practical relevance in the areas of routing and scheduling. Indeed, there exist many applications for the CCPP and other problems within its generalization, the CARP. We shall point out some well-known examples: postal mail delivery, school bus routing, road sweeping, winter gritting or household refuse collection. But applications are not limited to the routing of creatures or goods. Interesting variants may also be found in industrial manufacturing, e.g. the routing of automatic machines that put conducting layers or components

on to a printed circuit board (e.g. see Ball & Magazine, 1988). The CCPP is NP-hard (Golden & Wong, 1981), so the interest focuses on the development and implementation of heuristics.

Besides the origins of arc routing, which go back to the famous "Königsberger Brückenproblem" of Leonard Euler (1736) (e.g. see Fleischner, 1990) and to the Chinese mathematician Kwan (1962), the work on corresponding heuristics became commonly known with Golden, DeArmon, & Baker (1983) (see also Christofides, 1973; Golden & Wong, 1981) and was extended by Pearn (1989) and Eglese & Murdock (1991). During the last few years capacitated arc routing and its extensions have again become popular with the rise of the group of metaheuristic methods, especially *Tabu Search* (TS), *Genetic Algorithms* (GAs) or *Simulated Annealing* (see Amberg, Domschke, & Voß, 2000; Hertz, Laporte, & Mittaz, 2000; Lacomme, Prins, & Ramdane-Chérif, 2001). Comprehensive overviews, which also include the examination of problem variants like the *Rural*, *Windy* or *Maximum Benefit Postman Problem*, can be found in Assad & Golden (1995) or in Eiselt, Gendreau, & Laporte (1995a,b). A book exclusively devoted to arc routing was edited by Dror (2000). Recent work treats the *Hierarchical Postman Problem* (Ghiani & Improta, 2000), the *Mixed Rural Postman Problem* (Corberán, Martí, & Romero, 2000), covers the inclusion of turn penalties (Clossey, Laporte, & Soriano, 2001; Lacomme et al., 2001) or deals with the development of improved bounds (Amberg & Voß, 2002).

From the solution-oriented side generally two streams of approximative methods are to be distinguished: methods which are based on Eulerian graphs (algorithms generally using matching components) or methods which directly work in the original graph. Regardless of this classification, they all have in

common the necessity to form routes by some heuristic rules. Here modern metaheuristic methods have proved to be successful. Again, these iterative procedures can be divided into (at least) two classes: Procedures which operate with neighbourhoods that generate one new solution at each iteration or procedures which simultaneously operate on more than one solution. This (at first sight) describes the difference between a TS implementation and a GA implementation. Recently a popular trend has manifested itself: the creation of so-called *hybrid* procedures, in which a number of *pure* metaheuristics, precisely their components, is combined. However, this trend would not have survived if there had not been any advantage in doing so. One advantage was that hybridization easily enables a single-solution method to additionally profit from the "general and common knowledge" of multiple solutions, till then, an advantage exclusively reserved for GA methods. According to Glover (1998a), these hybrids belong to the class of *solution combination methods* (SCMs). Within this framework GAs are accepted as a special instance of the so-called *Scatter Search* (SCS), which itself belongs to a generalized form called *Path Relinking* (PR).

PR became known as a tuning add-on to TS (Glover, 1989) which provides additional possibilities to exploit adaptive memory. By means of certain choice criteria to generate moves and, hence, neighbourhood solutions, a path between two good solutions is built with the aim to hopefully detect new solutions with higher quality. These might be superior to the current best solution of an optimization problem because they have attributes of a quality similar to that of the initializing *elite* solutions. The SCS paradigm, though originating far earlier (Glover, 1977), has not become sufficiently known. Following Glover, SCS is the first general method which is able to *combine* a number

of solutions. This is done by linear combinations, performed on a subset of solutions which are designated to be elite in terms of comprising attributes which are deemed to characterize a higher solution quality. The new and frequently infeasible offsprings are then transformed into *feasible* ones by means of some *generalized rounding* procedures. (It is not of primary importance, however, to obtain feasibility. The term *acceptable* would be more convenient since heuristics which usually build on newly combined solutions may also operate on infeasible solutions). As in every population method, afterwards the new solutions are evaluated and possibly inserted into the elite collection of starting points. This is iteratively performed until a certain termination criterion stops the process. The main task in every SCS approach is the balancing between a homogeneous population, while simultaneously maintaining a necessary degree of diversity. In contrast to classical GAs, the main advantage in SCS is the ability to introduce more useful information into a generation process, which is basically due to the phenotypical encoding in contrast to the genotypical encoding which is used in GAs. However, in addition, there is also the inherent disadvantage that the exploration of an SCS solution space opens more dimensions, or, in other words, that the quality of the solutions visited is generally complex and hard to evaluate. Nevertheless, this *evolutionary* framework outlined above gives a promising foundation for the design of a hybrid metaheuristic procedure which is based on a combination of TS with a population strategy. This work is also encouraged by the experience of Cung, Mautor, Michelon, & Tavares (1997), who studied an SCS/TS procedure in the context of the Quadratic Assignment Problem, and Greistorfer (1999), who combined a TS and a GA to solve a polygon scheduling problem.

The CCPP procedure to be introduced follows a 2-step approach. In the *first*

step the underlying graph is made Eulerian, which can be achieved in polynomial time by the use of a matching procedure. Then a starting solution is built by a descent algorithm which is based on the savings idea of Clarke and Wright (Greistorfer, 1995). The *second step*, the main part of this work, is devoted to the design of a metaheuristic improvement procedure based on the wish to take advantage of the individual benefits of a (basically) single-solution-oriented approach and a (highly) population-oriented approach. As just outlined, a basic TS works on single solutions by building neighbourhoods from which a best admissible candidate is passed to the next iteration. Advanced forms make use of population based strategies and maintain a pool of elite solutions. In doing so, an intensification phase of a TS may include the use of a frequency memory and may benefit from the periodical recovery of elite solutions or from elements of population techniques such as SCS or PR. In this paper we focus on the combination of a TS with an advanced SCM, a strategy which can be interpreted as a type of SCS.

Now we introduce the notation to facilitate the description of the algorithms. Let $G = (V, E)$ be an undirected connected graph, where $V = \{1, \dots, l\}$ is the node set and $E = \{(i, j) | i < j; i, j \in V\}$ is the set of (undirected) edges with cardinality $n = |E|$. Further, we assume a given cost and demand matrix and that the depot node is node 1. The vehicle capacity is given by W . A solution to the problem is a schedule S which comprises routes R_1, \dots, R_m , where m is a decision variable. The total cost of a schedule is $c(S)$. Route demands are denoted by $d(R_p)$ ($p = 1, \dots, m$) and a shortest path (SP) between two nodes is abbreviated with $SP(i, j)$.

In Section 2 we shall present a *matching based construction heuristic*. Section 3 will introduce the hybrid metaheuristic improvement procedure: the

Tabu Search framework, the *population components* and their interaction, which constitutes the *Tabu Scatter Search*. The computational results will be presented in Section 4. Finally, we shall conclude with some remarks and an outlook in Section 5.

2 Matching Based Construction Heuristic

The starting procedure is the *matching based construction heuristic* (MBCH). *Step 1* of the following description goes back to Edmonds & Johnson (1973). It builds a minimum cost Eulerian graph from which the solution of the relaxed (uncapacitated) problem can be traced easily. If an undirected graph is non-Eulerian, then it always has an even number of odd-degree nodes, say ω . Thus, the addition of $\frac{\omega}{2}$ paths which start at one odd-degree node and end at another odd-degree node transforms the graph into a Eulerian one because now all nodes are of even degree. Such a minimum cost partition of odd-degree nodes can be optimally determined by solving a Minimum Weighted Matching Problem (MWMP). Detailed hints for an efficient MWMP implementation are given in Lawler (1976) who describes a polynomial time bounded labeling algorithm which has the running time $O(\omega^3)$.

Steps 2 to 4 are responsible for merging initial routes into bigger ones, while trying to obtain a cost reduction. This is done by an adaption of the (node-oriented) savings approach of Clarke & Wright (1964). The description basically follows Greistorfer (1995):

Matching Based Construction Heuristic (MBCH)

Step 1: (One big route) Identify the odd-degree nodes in the problem

underlying graph G and compute the lengths of all SPs joining two odd-degree nodes. Solve a MWMP in a complete graph which is defined on those odd-degree nodes having edges with costs equal to the SPs lengths. Augment G by adding one SP for every pair of matched nodes and obtain the Eulerian (multi-) graph G^* .

Step 2: (Service cycles) Select an unserved edge $p = (i, j)$ from G^* .

According to the maximum load W construct a minimum cost *service cycle* (SC) SC_p by connecting i and j with an $SP(i, j)$ found in G^* and mark *selected* edges of SC_p as serviced. Remove all edges of SC_p from G^* . Repeat *Step 2* until all edges are serviced.

Step 3: (Initial routes) Link every SC_p by means of a minimum cost *depot cycle* (DC) DC_p to obtain a route R_p . Sort all routes by decreasing lengths of DCs to obtain a list S of preliminary routes.

Step 4: (Merging routes) In an *ordered search* merge an SC_i via a common node into any SC_j , where $j > i$, if $d(R_i) + d(R_j) \leq W$. Delete DC_i and set $d(R_j) := d(R_j) + d(R_i)$. Repeat *Step 4* until no further merging is possible.

Step 5: (End) S is a feasible schedule of the problem.

After the construction of a Eulerian graph in *Step 1*, the initial SCs are formed in *Step 2* by so-called shortest paths cycles. Service edges are *selected* according to a last-in-first-out-rule. An edge of SC_p is set serviced if it appears for the last time in the current Eulerian subgraph. *Ordered search* means that, starting from the top of the list, for a route R_i it is checked whether its SC

can be included in any route R_j which has a shorter DC. The corresponding saving in that case is the cost of the longer DC, DC_i , which is deleted. If no inclusion is possible due to the capacity restriction, then R_i is no longer a candidate to be merged and is therefore marked as fixed. A route which was not fixed and only dropped because it did not have a common node with any other route may be reconsidered in subsequent iterations.

3 Tabu Scatter Search Metaheuristic

Since its introduction in Glover (1986), TS has been proven to be one of the most efficient heuristic optimization techniques. The basic concept of simply making recent moves *tabu* was substantially extended by many authors who investigated the TS framework, e.g., for vehicle routing, scheduling or assignment problems. For a comprehensive literature overview see the bibliography by Osman & Laporte (1996). The present know-how regarding TS and its refinements as well as its connections to broader concepts, such as PR and SCS, are described in Glover & Laguna (1997); Glover (1998a) and Corne, Dorigo, & Glover (1999). One of the refinements discussed is the embedding of multi-solution aspects which originally did not appear to be typical of TS. Such an influence is performed by maintaining a set of high quality solutions which have been encountered during the search and by deriving specific information from this set to guide the search. A somewhat self-evident and therefore more obvious population aspect can be found in GAs, the success of which has directly been linked with the power of evolutionary populations for many years. So it is easy to understand that a number of TS/GA-hybrids has been propagated to highlight the profitable union of these established algorithmic

foundations (Thangiah, Osman, & Sun, 1994; Costa, 1995; Greistorfer, 1999). The success of these studies has been one reason why the present paper deals with a type of hybrid algorithm. Another motivation comes from the effective application of (more or less) hybrid SCS approaches, especially in the area of linear programming, which was reported in recent papers (Cung et al., 1997; Laguna, 1997; Glover, Løkketangen, & Woodruff, 2000). So it was a natural step to support a TS by an SCM add-on to explore the influence of a population strategy on the ability of generating high quality solutions in a framework typical of SCS.

In the next subsections we shall give a detailed step-by-step description of the individual components of the hybrid *Tabu Scatter Search* (TSCS) for the solving of the CCPP.

3.1 *The Tabu Search Framework*

Neighbourhood: To perform a transition from an old solution to a new one, a compound neighbourhood is used. It works on different move operators which are used to incorporate an algorithmic intensification and diversification strategy. Basically, the neighbourhood uses exchange and insertion moves of edge *sequences*. A sequence is a segment of edges which contains one or two edges to be serviced. In the latter case the two service edges (marking the start and the end of the sequence) are linked by an SP. Hence, the whole neighbourhood representation is about a permutation of all (service) edges contained in the underlying graph. This significantly facilitates the data structure, since every solution may be stored by means of a permutation of the numbers $1, \dots, n$ which represent edges that are linked by SPs. The route structure itself is

given by pointers to permutation positions which mark the end points of a route. Assume two sequences, *source* and *target*, and let sl , tl denote their lengths. Then $sl \cdot tl \neq 0$ describes an exchange move. In this case the source sequence is moved to the target position and vice versa. A zero-length of one of the both sequences defines an insertion of the other sequence at the "position" of the zero-length sequence. The move operators treated, A, B, \dots, H , are summarized in Table 1.

— **Insert Table 1** —

	$tl = 0$	$tl = 1$	$tl = 2$
$sl = 0$	—	A	B
$sl = 1$	C	D	E
$sl = 2$	F	G	H

Table 1

TS move operators

D , E , G and H are operators for exchange moves. The insertion moves of operators A , B (backward moves) and C , F (forward moves) are designed for the construction of new routes. New routes are initialized in the case that the capacity restriction would not be met in the source route (A , B moves) or in the target route (C , F moves). Conversely, these moves may cancel an existing route by taking its last service edge. For all operators it should be noted that source and target sequences may come from the identical route. This allows an improvement without substantially changing the assignment of edges to routes within a given schedule. If different routes are considered, then a potential exchange move is disregarded if it would violate the capacity restriction. A similar neighbourhood setting is the λ -interchange neighbourhood mechanism (e.g. see Osman & Wassan, 2002). Further information about the usefulness of compound neighbourhoods can be found, e.g., in Martin, Otto, & Felten (1992), Glover & Laguna (1997) and Mechti, Poujade, Roucairol, & Lemarié (1999).

For a given operator, which defines sl and tl , a move to a neighbouring solution can be described by a $move(s,p,t,q)$. Letters s , t denote the first edge in the source, target sequence of a route p and q , i.e. before executing the move, s belongs to R_p and t to R_q . For $sl, tl = 0$ the edge variables s, t have a dummy function. Figure 1 shows an illustrative neighbourhood example for a 2-route solution of a CCPP with 9 edges. Sequence lengths sl and tl are listed and the edges s and t are underlined. Source sequences are given by $\langle \underline{8} \rangle$ and $\langle \underline{5}, 1 \rangle$ whereas $\langle 6 \rangle$ and $\langle 3, \underline{9} \rangle$ are target sequences. In this example all source and target sequences come from different routes. It is assumed that all capacity restrictions can be met and therefore no additional route has to be constructed in the cases A , B , C and F .

— Insert Figure 1 —

Memory: A thorough tuning between intensification and diversification search phases ensures the best possible trade-off between the pursuit of good solutions which already exist and the finding of different and hopefully better solutions. In our TS implementation both phases are alternated. In doing so an (initial) intensification phase is achieved by a TS run using all move operators, while the attractiveness of each operator is noted by *long-term* frequency counts of improvements (on the current solution). Then, an diversification phase follows with the exclusive use of the most attractive operator which was noticed in the previous intensification phase. The idea behind this is based on the reasoning that the highest possible degree of freedom in the selection of an operator will make the procedure scan the vicinity of the current solution, which is supposed to contain a similar neighbour. If these choice possibilities are limited to a single operator, as it was observed, the course of the optimization tends to drift away in terms of cost, which is a diversification signal. Diversification increases all the more, the more it is triggered by an operator which has already sufficiently spent his positive effect in the previous intensification phase. Hence, the operator with the largest amount of frequency counts is chosen to diversify the search.

In every iteration the best *admissible* solution recognized in the neighbourhood is chosen to be the starting point for the next neighbourhood to be examined. Admissibility (i.e. the permission to be chosen due to the tabu restrictions and the aspiration criterion) is monitored by the *short-term* TS memory. Assume that the outcome of an iteration τ is the *move*(s,p,t,q). Then array *Rec*, which is used for the short-term memory, records the taking of the sequence(s) from its (their) route(s):

$$Rec(s,p) = \tau \quad \text{and/or} \quad Rec(t,q) = \tau .$$

Conjunction *or* stands for the insertion case. If, for instance, the *s*-sequence was *inserted* in R_q , i.e. $tl = 0$, then the non-taking of a (dummy) *t*-sequence from R_q would naturally not be stored. The recency information at an iteration $\tau' > \tau$ is used as follows. A planned $move(s,p,t,q)$, yielding S' , is *admissible*

a) if it is not short-term tabu, i.e. if

$$\tau' - Rec(s,q) > tSize \quad \text{and/or} \quad \tau' - Rec(t,p) > tSize ,$$

b) or if it is short-term tabu, but $c(S') < c(S_{best})$ (*aspiration-rule*).

The tabu list length, $tSize$, is the number of iterations during which the reversal of a move is not allowed. As suggested in many applications (see Laguna & Glover, 1993; Osman, 1993; Taillard, 1991) we decided on the use of a dynamic policy, where this control parameter is varied during the search. In addition, according to a set of experiments, it turned out to be useful to provide a sequence of numbers, rounded from $0.7n, 0.8n, \dots, 1.3n$, which $tSize$ follows by changing its value every $tSize$ iterations.

3.2 Population components

Initial Population: Every member of the initial pool is built by a random permutation of all edges which is led to a feasible solution by the use of two straightforward heuristic rules. Firstly, the assignment of edges to routes is obtained by splitting the permutation into a set of edge *clusters* with respect to the given capacity W . In doing so, service edges are determined according to a first-in-first-out-rule (fifo-rule) in the sequence in which they occur in the original random permutation. This constructs a feasible schedule of prelimi-

nary routes. Since this obviously tends to be insufficient with respect to the *sequence* of the edges, then, secondly, this provisional schedule may be improved by the following *greedy sequencing heuristic* (GSH). This approach is a variant of the so-called path scanning algorithms proposed by Golden et al. (1983).

Greedy Sequencing Heuristic (GSH)

Step 1: (Initialize) Set all edges unserviced and $i := 1$.

Step 2: (Include an edge) Let j be the nearest node to i , where j is incident to at least one unserviced edge from the route which is currently being considered. Insert an $SP(i, j)$ and choose an arbitrary k to include an unserviced edge (j, k) , which is set serviced. If all edges of the current route are set serviced, then close it with an $SP(k, 1)$ and set $i := 1$, otherwise set $i := k$.

Step 3: (Extending routes/schedule) Iterate *Step 2* until all routes are sequenced.

Note that in *Step 2* not only edges may be newly sequenced but they may also be directed to achieve a cost reduction. However, as outlined, the edge *assignment* to routes is not affected by the GSH. It is solely the result of a low-level heuristic, namely the fifo-rule mentioned. The construction of high quality edge clusters is the purpose of the SCM operator.

SCM operator: Cung et al. (1997) propose an SCS approach for the Quadratic Assignment Problem (QAP). For the ease of explanation, let us assume that any QAP solution can be represented by a (quadratic) binary matrix $Q := (q_{ij})$. A trial point Q^P is then found by adding up the coefficient matrices Q_i^E

($i = 1, 2, \dots$) of some set of elite solutions. Of course this leads to an infeasible problem representation since certain cells of Q^P contain entries greater than 1 and/or some rows (columns) are not uniquely assigned. To transform Q^P into a feasible solution, say Q^F , a linear program (LP) is solved. It has the objective $\min \|Q^F - Q^P\|^2$, where $\|\cdot\|^2$ stands for the Euclidean norm. To obtain feasibility for the binary matrix Q^F , assignment constraints are added to ensure that there is exactly one decision variable with $q_{ij}^F = 1$ in each row and column, respectively. After resolving the terms in the objective, the non-constant part of it becomes $\min - \sum_i \sum_j q_{ij}^F q_{ij}^P$ under the supply and demand restrictions mentioned. If the sign is changed, this LP is a linear *Maximization - Assignment Problem*.

Now we shall demonstrate how we apply this idea to the CCPP, while we build upon the clustering sub-problem as the main aspect of a given CCPP solution. If the edges of E are identified with the numbers $j = 1, \dots, n$, then for a given solution S , the assignment of edges to routes is represented by an $m \times n$ matrix $A(S) := (a_{ij})$ with

$$a_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is serviced in } R_i \\ 0 & \text{else.} \end{cases}$$

Let $S_1^E, \dots, S_{n_c}^E$ be a set of n_c elite solutions which was chosen at random using a uniform distribution on the members of the pool. Then, the $m' \times n$ *assignment frequency matrix* $\Lambda := (\lambda_{ij})$, with m' is the maximum number of routes in any schedule S_i^E , $i = 1, \dots, n_c$, represents the (linear) combination of the elite solutions chosen and describes an infeasible trial point. Matrix Λ

is defined by

$$\Lambda = \sum_{i=1}^{n_c} A(S_i^E).$$

Like before, the Euclidean distance between the infeasible representation Λ and a combined and feasible assignment $A(S^F)$ can be minimized by a corresponding LP. In the non-quadratic coefficient matrix case this LP is a *Maximization - Transportation Problem* (Max-TPP) with the objective

$$\max \sum_{i=1}^{m'} \sum_{j=1}^{n'} a'_{ij} \lambda'_{ij},$$

where a dummy column $n' := n + 1$ has been added to $A(S^F)$ and Λ to obtain $A'(S^F) := (a'_{ij})$ and $\Lambda' := (\lambda'_{ij})$. This objective shows that the cost coefficients of the Max-TPP are the assignment frequencies of edges to routes in the set of schedules to be combined. Therefore the maximization favours assignments at positions which are often used and prevents the solution from including obvious unattractive assignments because they are rarely observed in the elite set. All decision variables a'_{ij} are non-negative and will be integers in any solution of the problem. Moreover, since every edge (column) must be serviced in a single route (row), the TPP demands d_j ($j = 1, \dots, n$) are equal to 1 and all corresponding decision variables are compelled to be zero or one. The number of edges which can be serviced in a route R_i constitutes its supply s_i ($i = 1, \dots, m'$). It is given by the approximation $\lceil W/d_{avg} \rceil$, where $\lceil \cdot \rceil$ denotes the round function and d_{avg} is the average edge demand. To balance the difference between the total supply of m' routes and the total demand of n (edges) the zero-cost column n' picks up the the oversupply $s_i \cdot m' - n$. Several tests revealed that this less restrictive setting works quite well in contrast to a supply function which depends on a direct relationship to the properties of

a combined route considered. On the other hand, this process may lead to a TPP solution whose CCPP correspondence has to be rejected since a capacity restriction is violated.

— **Insert Figure 2** —

In our implementation the Max-TPP is solved to optimality by means of the well-known primal modified distribution (MODI) method. Such a TPP solution, $A^*(S^F)$, gives an assignment of edges to routes, which is optimal in terms of minimizing the total distance to the assignments of the chosen elite schedules, which are combined in the infeasible representation Λ . Afterwards, the GSH, described in the previous subsection, is applied to S^F to derive a local optimal set of routes. Figure 2 shows an example of a CCPP with 6 edges each of which having unit demand. Vehicle capacity $W = 4$. It starts with the specification of three elite solutions, derives the assignment frequencies and includes demand, supply and dummy column values. An optimal Max-TPP solution is given in tableau form. From the latter one the preliminary new routes can be derived directly. Finally, the GSH produces a re-sequenced solution as it is supposed in the last table of Figure 2.

3.3 *Tabu Scatter Search Procedure*

Now we can present the whole TSCS. Its algorithmic backbone is the TS procedure described in Section 3.1 which is extended by a commonly shared pool of elite solutions. This pool is maintained by the TS, which inputs quality solutions and is used by the SCM to construct combined solutions. These solutions then serve as a basis for a new TS run. Here is a detailed algorithmic description:

Tabu Scatter Search (TSCS)

Step 1: (Initialization) Set TSCS parameters (for values see Section 4),
input the problem data and the solution obtained from the

MBCH of Section 2. Generate the initial TSCS pool, using random sequences, fifo-clustering and the GSH of Section 3.2. Initialize the TS memory.

Step 2: (TS phase) Iteratively make a) t_{int} TS iterations using all neighbourhood operators and b) subsequently t_{div} TS iterations with the best operator recognized in a) (see Section 3.1). Whenever a new current best solution is built or a *good* solution is found, make a pool input.

While performing the TS, execute an SCM phase, i.e. goto *Step 3*, every t_{scm} iterations. If the lower bound is reached or t_{max} iterations are made goto *Step 4*.

Step 3: (SCM phase) Randomly select $n_c \in \{2, \dots, n_{cmax}\}$ elite solutions from the TSCS pool. Derive Λ and solve the Max-TPP to obtain feasible edge clusters. Apply the GSH to transform these clusters into a final set of routes. Reset the TS memory, return to *Step 2* and continue the TS with the schedule found in the SCM phase.

Step 4: (End) Stop with the best solution found so far.

After the initialization in *Step 1*, the TSCS begins with a TS run in *Step 2*. As described earlier, in doing so, we distinguish between an intensification and a diversification phase with regard to the search operators used. The durations of these phases are given by the input parameters t_{int} and t_{div} . During the TS phase a new best solution is always fed into the pool, whereas other ones have to pass two criteria. Firstly, these *good* solutions must lie within a specified range of percentage cost deviation from the current best solution and, secondly, to avoid duplications in the pool, a duplication check based on costs must turn out to be negative. Naturally, this straightforward setting allows

more pool transfers concerning the cost quality of a solution and it is more restrictive than methods based on hash functions and/or full duplication checks concerning the actual structure of a solution (compare Woodruff & Zemel, 1993; Glover, 1998b). However, a comparison to original experiments without cost duplication checks revealed that even such a simple strategy has a clear positive effect. A new member of the pool takes up the place of the current worst one (again measured by the cost criterion).

Every t_{scm} iterations the algorithm calls the SCM subroutine of *Step 3*, selecting a random number of solutions (at least two) from the pool according to a uniform distribution. After having built a new combined solution via the Max-TPP-approach, the clusters found are sequenced by the GSH. The new solution is returned to the TS to serve as an initial starting point which may be improved after resetting the tabu memory. Finally, the algorithm terminates in *Step 4* at the latest after a total of t_{max} iterations.

4 Computational experience

All algorithms were coded in PASCAL and were run on an IBM compatible PC, PII-300Mhz/AL440LX, under DOS 7.1. For the evaluation of the procedures introduced, three sets of data are used. There are 12 randomly generated non-planar instances and another 10 planar instances which were constructed by hand (grid-graphs). Both sets have Euclidean distances and were used as standard benchmark problems during the development of the algorithms. The third set contains 23 instances from DeArmon (1981) which are frequently used for the testing of CARP procedures.

The TSCS parameters found to be appropriate are the following. The maximum total number of iterations was set to $t_{max} = 10,000$ which requires a reasonable computation time. The intensification and diversification parameters t_{int} and t_{div} were initially set to 2,000 and 400 iterations, respectively. During a run t_{int} remains constantly set to 2000 (except the last intensification period), while t_{div} decreases to 300, 200, reaching 100 iterations for the last diversification segment. The size of the population was 20. Though higher values were tested, we could not find any improvement over this moderate value. On the contrary, it turned out to be of more interest to use the memory recovered for data instances of larger sizes. A non-duplicate-cost solution was categorized as good and fed into the pool if its cost were within a 20% interval of the currently best-known solution, an approach that sufficiently ensures the diversity of the population. The SCM procedure was called every $t_{scm} = 200$ iterations with the maximum number of solutions to be combined $n_{cmax} = 5$. The operator sequence for intensification was *DCAFBGEH*, which gives some priority to operators using smaller sequences in exchange moves or operators using insertion moves. Finally, we introduced TS *restarts* from a schedule randomly chosen from the pool if a non-improving period of 3,000 iterations had been passed.

We shall start our discussion with the results for the random graphs. These are shown in Table 2. The first four columns give some characteristics of the problem instance: the name, the cardinality of the edge set, $|E|$, the total demand O and the vehicle capacity W . Column (5) shows the cost of the MBCH, the starting solution procedure of Section 2, and the number of routes, $|S|$, in the schedule found. The next three columns, (6) to (8), refer to a TS algorithm (TSA) which was presented in Greistorfer (1995). Columns (6) to

(7) give again cost and size of the solution followed by the *schedule efficiency* (*SE*) number, which is the ratio of the length of the lower bound *LB* in column (12) to the cost of the schedule. *LB* is computed as $\max\{c(G^*), NSLB\}$, where $c(G^*)$ is the length of the Eulerian graph and *NSLB* depicts the so-called node scanning lower bound as it was introduced in Assad, Pearn, & Golden (1987). In column (8) t^* shows the iteration number in which the best solution of column(6) was found. In these early experiments the total number of iterations for all TSA runs was set between 2,000 and 12,000 iterations depending on the problem size. The columns in (9) give the TSCS results which also were computed starting from the solution in (5). The CPU times for determining the best solution and the overall time in (13) and (14) are given in seconds.

— **Insert Table 2** —

(1)	(2)	(3)	(4)	(5)		(6) (7) (8)				(9)		(10) (11) (12)		(13)	(14)	
				MBCH		TSA				TSCS						
	$ E $	O	W	$c(S)$	$ S $	$c(S)$	$ S $	SE	t^*	$c(S)$	$ S $	SE	t^*	LB	sec_{t^*}	sec
GrzR1	18	77	25	4871	4	4657	4	0.95	3005	4657	4	0.95	33	4443	0.1	35.4
GrzR2	22	107	25	6047	5	5708	5	0.93	2692	5708	5	0.93	37	5329	0.2	53.2
GrzR3	45	260	35	11282	9	10775	8	0.96	6470	<u>10733</u>	8	0.96	1229	10353	28.9	235.5
GrzR4	33	153	35	8923	5	8337	5	0.97	9142	8337	5	0.97	50	8083	0.6	130.6
GrzR5	31	147	25	8479	7	7723	7	0.84	1739	<u>7508</u>	6	0.86	7728	6487	83.0	107.4
GrzR6	58	298	35	14247	10	13770	9	0.94	1949	<u>13483</u>	9	0.96	5585	13007	227.6	406.5
GrzR7	62	324	30	16165	14	15454	12	0.93	7969	<u>15315</u>	11	0.94	7959	14436	360.8	453.1
GrzR8	31	183	40	13388	6	12224	5	0.77	479	<u>12013</u>	5	0.78	9070	9416	102.9	113.3
GrzR9	82	435	40	20622	14	20269	12	0.89	2989	<u>19258</u>	12	0.94	6938	18008	568.0	820.1
GrzR10	9	36	20	2969	2	2619	2	1.00	49	2619	2	1.00	14	2619	0.0	0.0
GrzR11	12	52	20	3388	3	3040	3	1.00	146	3040	3	1.00	105	3040	0.2	0.2
GrzR12	15	67	20	4595	4	4065	4	0.94	9	4065	4	0.94	75	3819	0.2	23.0
average seconds										114.4						198.2
number of best						6				12						
number of improvements										6						
improvements in %										50.0						

Table 2
Results for random graphs

Though the MBCH+TSA appears to produce acceptable results, as can be seen in column (7), the combined method MBCH+TSCS improved half of them. These cases are underlined in (9). Bold print indicates a proved optimal solution. At the bottom of the table one can additionally find the average running times and the number of the best known solutions which is twice as high as in the old TS and comprises all instances for the MBCH+TSCS.

Table 3 presents the results for the constructed graphs. The structure of the heading is the same as in Table 2. The new algorithm outdoes the old one in 60% of all cases. On the other hand, it is obvious that random graphs are easier to solve than constructed graphs which are planar. An explanation can be seen in the fact that the solution space of the latter one is more difficult to explore since improvements can only be made at the expense of far-reaching schedule reorganizations. However, if an improvement is found, then it is generally larger than for random graphs. This can be seen by comparing the (relative) differences between the $c(S)$ -values of (9) and (5) in both tables. Note that a $t^* = 0$, as for instances GrzC1 and GrzC8, indicates that the overall best solution found was already a member of the initial pool, which could not be improved afterwards. Due to the small problem sizes it may be assumed that these solutions are optimal.

— **Insert Table 3** —

(1)	(2)	(3)	(4)	(5)		(6) (7) (8)				(9)		(10) (11) (12)		(13)	(14)	
				MBCH		TSA				TSCS						
	$ E $	O	W	$c(S)$	$ S $	$c(S)$	$ S $	SE	t^*	$c(S)$	$ S $	SE	t^*	LB	sec_{t^*}	sec
GrzC1	12	52	20	2256	4	1792	3	0.88	484	1792	3	0.88	0	1570	0.0	14.8
GrzC2	47	235	50	76429	6	73917	5	0.93	704	<u>73679</u>	5	0.93	3971	68863	110.5	274.6
GrzC3	66	387	35	70262	15	63662	12	0.63	3321	<u>57462</u>	12	0.70	8754	39954	447.8	511.7
GrzC4	74	380	40	69559	13	60359	10	0.66	1183	<u>57680</u>	10	0.69	4290	40059	287.8	670.5
GrzC5	60	357	50	46200	8	44600	8	0.66	5895	<u>42400</u>	8	0.70	7067	29500	315.5	443.9
GrzC6	33	153	20	40400	10	34400	8	0.57	3023	<u>33200</u>	8	0.59	7274	19600	86.6	118.9
GrzC7	22	107	30	21800	6	16200	4	0.66	3895	<u>15600</u>	4	0.69	307	10700	1.7	54.4
GrzC8	11	46	25	4430	2	3982	2	0.89	95	3982	2	0.89	0	3534	0.0	13.1
GrzC9	16	71	25	75873	5	49720	3	0.75	747	49720	3	0.75	41	37111	0.1	28.4
GrzC10	19	87	25	92783	6	65565	4	0.78	1584	65565	4	0.78	833	51370	3.3	40.0
average seconds										125.4						217.0
number of best						4				10						
number of improvements										6						
improvements in %										60.0						

Table 3
Results for constructed graphs

All results indicate that it is possible to find good and improved solutions for our own data sets. To evaluate the performance of the TSCS in a direct comparison with another TS algorithm from literature, we refer to the work of Hertz et al. (2000) and their CARPET algorithm. We do not directly compare our results with Lacomme et al. (2001), who present a GA/local search combination which uses a restart strategy, because the authors do not provide specific running times. As mentioned above, we used the well-known DeArmon instances, which are also used by Hertz et al. (2000). The results are shown in Table 4. Again, the main problem parameters are given in columns (1) to (4) and the starting solutions of the MBCH are shown in (5). After the TSCS-columns (6) to (10), the results of the CARPET algorithm are given in (11) and (12). The running times of CARPET refer to an SGI system (Silicon Graphics Indigo-2, R10000-195MHz/IP28), and are also given in seconds. The last two columns, (13) and (14), contain the best known solutions and the lower bounds given in Hertz et al. (2000) and Amberg & Voß (2002).

— **Insert Table 4** —

(1)	(2)	(3)	(4)	(5)		(6)		(7) (8) (9) (10)			(11) (12)		(13)	(14)	
				MBCH		TSCS						CARPET			
	$ E $	O	W	$c(S)$	$ S $	$c(S)$	$ S $	SE	t^*	sec_t^*	sec	$c(S)$	sec	Best	LB
DeA1	22	22	5	342	5	316	5	1.00	240	1.7	1.7	316	17.1	316 ^{b)}	316
DeA2	26	26	5	345	6	339	6	1.00	1338	13.2	13.2	339	28.0	339	339
DeA3	22	22	5	289	5	275	5	1.00	146	1.0	1.0	275	0.4	275 ^{c)}	275
DeA4	19	19	5	318	5	287	4	1.00	634	3.3	3.3	287	0.5	287 ^{c)}	287
DeA5	26	26	5	424	7	377	6	1.00	1978	19.3	19.3	377	30.3	377	377
DeA6	22	22	5	352	7	298	5	1.00	1818	12.7	12.7	298	4.6	298	298
DeA7	22	22	5	364	5	325	5	1.00	11	0.1	0.1	325	0.0	325 ^{b)}	325
DeA10	46	249	27	392	12	<u>348</u>	11	0.99	5337	166.3	310.9	352	330.6	348	344
DeA11	51	258	27	367	11	321	10	0.94	2979	116.0	388.8	<u>317</u>	292.2	303 ^{d)}	303
DeA12	25	37	10	289	4	275	4	1.00	1745	16.3	16.3	275	8.4	275 ^{c)}	275
DeA13	45	225	50	409	6	395	5	1.00	659	21.6	21.6	395	12.4	395	395
DeA14	23	212	35	601	9	458	7	0.98	155	1.0	70.0	458	111.8	458	450
DeA15	28	245	41	556	8	544	7	0.99	418	4.7	111.3	544	13.1	540 ^{d)}	536
DeA16	21	89	21	106	6	100	5	1.00	51	0.3	0.3	100	2.6	100 ^{c)}	100
DeA17	21	112	37	60	4	58	4	1.00	33	0.3	0.3	58	0.0	58 ^{a)}	58
DeA18	28	116	24	129	6	127	5	1.00	1429	16.5	16.5	127	9.2	127 ^{c)}	127
DeA19	28	168	41	93	5	91	6	1.00	9	0.1	0.1	91	0.0	91 ^{a)}	91
DeA20	36	153	37	172	6	164	5	1.00	262	5.4	5.4	164	1.5	164 ^{c)}	164
DeA21	11	66	27	63	3	55	3	1.00	0	0.0	0.0	55	1.1	55 ^{c)}	55
DeA22	22	107	27	123	4	121	4	1.00	9786	67.9	67.9	121	51.5	121	121
DeA23	33	154	27	160	7	156	6	1.00	4266	68.7	68.7	156	6.1	156 ^{c)}	156
DeA24	44	205	27	204	9	200	8	1.00	1378	40.1	40.1	200	18.3	200 ^{c)}	200
DeA25	55	266	27	237	12	235	11	0.99	36	1.7	458.4	235	186.3	233 ^{c)}	233
average deviation in %						0.33						0.32			
worst deviation in %						5.94						4.62			
average seconds												25.1		70.8	
number of optima						18						18			
number of best						20						19			

Running times of the TSCS in (9) and (10) scaled with respect to the CARPET-PC.

Best solutions in (13) firstly reported

a) by Christofides (1973),

b) by Golden et al. (1983),

c) by Pearn (1989),

d) by Lacomme et al. (2001) and

the rest by Hertz et al. (2000); (11) and (13) of DeA10 obtained with different CARPET versions.

Lower bounds in (14) as given in Hertz et al. (2000) except DeA14 improved by Amberg & Voß (2002).

Table 4
Results for the DeArmon data set

Instances 8 and 9 are not contained in Table 4 since they are known to have inconsistent data. Like before, all optimal results for both algorithms in columns (6) and (11) are printed in bold type. In order to compare the computing times in an accurate way, our results are scaled as if they had been run on the SGI system. The scaling was performed using benchmarks provided by SPEC (2001). Our computing times were multiplied by the ratio $SPECint95(PII)/SPECint95(R10000)$, which is 11.6/8.88, indicating that the Indigo-2 is the slower computer.

As can be seen in Table 4, with one exception (DeA11) the MBCH+TSCS combination fully obtained the very good solutions of CARPET. The number of optimal solutions is the same in both approaches, whereas our method finds one more best known solution than CARPET (DeA10). The average deviation from the best known solution (heuristic solution value over the best known solution value) is almost the same, while the worst deviation is 1.32% higher (instance DeA11) in our procedure. Here it must be noted that all results reflect the unique standard parameter setting as given above and that a non-standard TSCS found several improvements including a $c(S) = 319$ for DeA11. On the other hand, our procedure clearly obtains its best results (on the average) faster than the total running times of CARPET indicate. It has to be admitted, however, that our scaled total running times are longer because all non-optimal instances are treated over a pre-defined and constant period of iterations whereas CARPET stops after a number of non-improving iterations. A second reason is that the TSCS neighbourhood requires considerably more computational effort to search the neighbourhood as the problem size increases. Lacomme et al. (2001) report several improvements on the DeArmon results of Hertz et al. (2000), where both are using a single parameter setting.

There are three improvements over CARPET (DeA10, DeA11, DeA15), while generating two new best known solutions (DeA11, DeA15) and one new optimum (DeA11). The current best known result for DeA10 can only be reached by the TSCS. Regarding the running times, however, it cannot be exactly specified to what the best performance of the hybrid GA relates in the isolated case, since the authors only give a single *average CPU time*. Deriving an average *SPECint95(PIII)* for 500Mhz processors from SPEC's database (the motherboard is not specified), gives 21.15 which allows a direct comparison on the basis of the Hertz-PC. This results in an *average CPU time* of $21.15 \cdot 21.0/8.88 = 50.0$ seconds for the hybrid GA compared to the *average total running times* of CARPET and TSCS which are 49.0 and 70.8 seconds, respectively, while the TSCS needs on the average 25.1 seconds to derive its *best* results.

All in all, the hybrid procedure introduced (MBCH+TSCS) has the power to significantly improve not only our starting solution (MBCH) but also the previous optimization framework (MBCH+TSA). As the comparison shows, it is also able to keep up with well-performing algorithms from literature.

5 Summary and Outlook

In this paper we have presented a heuristic framework for solving the well-known Capacitated Chinese Postman Problem, which has many real-world applications. Important areas to point out are the routing of vehicles, finding good schedules for postmen, but also applications in industrial engineering. The framework designed is strongly related to the ideas of Tabu Search and Scatter Search.

A starting solution for the problem is provided by a matching based construction heuristic. It merges routes, which were built as cycles in a Eulerian graph, to iteratively achieve greedy-like cost reductions until the algorithm terminates in a local optimum. These solutions are generally fast to construct. However, they clearly tend to be improvable by some more advanced techniques. The main focus in building an improvement procedure was placed on the task to combine two established methods to gain from individual benefits. The basic routine is a Tabu Search procedure with a compound neighbourhood that is able to react to the course of the optimization. Supported by attribute based short-term tabu memory and an aspiration criterion, this Tabu Search meta-heuristic also maintains a pool of elite solutions. They are used to generate combined solutions to be further improved by the Tabu Search. The solution combination operator works on the basis of assignment counts in the set of elite solutions to be combined. These counts estimate the desirable assignment of a given edge to a specific route. We propose a transportation problem formulation which finds a total Euclidean distance minimizing solution with respect to the solutions from which it was combined. Simultaneously preliminary feasible routes are built which are subsequently put in order by a path scanning type sequencing heuristic.

The computational experience with several data sets, like non-planar random graphs or planar grid-graphs and instances from literature, were presented and discussed. The merits of explicitly adding population components to a Tabu Search are obvious and understandable. The results appear to be promising, both in terms of quality and running time, and they show that the algorithm designed is competitive with respect to benchmarks from literature.

Future work will concentrate on theoretic research and algorithmic improve-

ments. Firstly, there is room for improving the bounds which should enable a better evaluation of heuristic developments (see Amberg & Voß, 2002). Secondly, it would be worthwhile to find linear programming representations that can transform the solution combination problem to higher levels which, besides the clustering problem, also deal with the sequencing problem. In doing so, Hamming distances could be a means of introducing more sensitive evaluation functions for the definition of being part of the elite and could help to quantify qualitative attributes which are generally hard to catch. Concerning the pool management, it would be interesting to test transfer procedures which are closer to the proposals of the template paper of Glover (1998b). Finally, from the algorithmic point of view, advanced data structures for an efficient search of the neighbourhood, e.g. following the implementation ideas in Wassan & Osman (2002), could make the proposed algorithms very competitive and able to solve large instances more efficiently. Moreover, the modelling of additional side constraints, like a mixed fleet or time windows, could be a challenge with interesting perspectives.

References

- Amberg, A., Domschke, W., & Voß, S., 2000. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European J. Opl Res.* 124 (2), 360–376.
- Amberg, A., & Voß, S., 2002. A hierarchical relaxations lower bound for the capacitated arc routing problem. In: Sprague, R. H. (Ed.), *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, IEEE, Piscataway. Vol. DTIST02. pp. 1–10.
- Assad, A. A., & Golden, B. L., 1995. *Arc routing methods and applications*.

- In: Ball, M. O., Magnanti, T. L., Monma, C. L., & Nemhauser, G. L. (Eds.), Network Routing. Vol. 8 of Handbooks in Operations Research and Management Science. North-Holland, Amsterdam et al., pp. 375–483.
- Assad, A. A., Pearn, W. L., & Golden, B. L., 1987. The capacitated Chinese postman problem: Lower bounds and solvable cases. *Am. J. Math. Mgt. Sci.* 7(1-2), 63–88.
- Ball, M. O., & Magazine, M. J., 1988. Sequencing of insertions in printed circuit board assembly. *Ops Res.* 36 (2, Mar.-Apr.), 192–201.
- Christofides, N., 1973. The optimum traversal of a graph. *Omega* 1, 719–732.
- Clarke, G., & Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Ops Res.* 12, 568–581.
- Clossey, J., Laporte, G., & Soriano, P., 2001. Solving arc routing problems with turn penalties. *J. Opl Res. Soc.* 52 (4), 433–439.
- Corberán, A., Martí, R., & Romero, A., 2000. Heuristics for the mixed rural postman problem. *Comput. & Ops Res.* 27 (2), 183–203.
- Corne, D., Dorigo, M., & Glover, F., 1999. *New ideas in optimization*. McGraw-Hill, London.
- Costa, D., 1995. An evolutionary tabu search algorithm and the NHL scheduling problem. *INFOR* 33 (3), 161–178.
- Cung, V.-D., Mautor, T., Michelon, P., & Tavares, A., Apr. 1997. A scatter search based approach for the quadratic assignment problem. In: Bäck, T., Michalewicz, Z., & Yao, X. (Eds.), *Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference*. Indianapolis, USA, pp. 165–170.
- DeArmon, J., 1981. A comparison of heuristics for the capacitated Chinese postman problem. Master's thesis, University of Maryland, College Park MD.

- Dror, M. (Ed.), 2000. Arc routing: Theory, solutions and applications. Kluwer Academic Publishers, Boston.
- Edmonds, J., & Johnson, E. L., 1973. Matching, Euler tours and the Chinese postman. *Math. Prog.* 5, 88–124.
- Eglese, R. W., & Murdock, H., 1991. Routeing road sweepers in a rural area. *J. Opl Res. Soc.* 42 (4), 281–288.
- Eiselt, H. A., Gendreau, M., & Laporte, G., 1995a. Arc routing problems, part I: The Chinese Postman Problem. *Ops Res.* 43 (2, Mar-Apr), 231–242.
- Eiselt, H. A., Gendreau, M., & Laporte, G., 1995b. Arc routing problems, part II: The Rural Postman Problem. *Ops Res.* 43 (3, May-Jun), 399–414.
- Fleischner, H., 1990. Eulerian graphs and related topics. Part 1, Volume 1. Vol. 45 of *Ann. Disc. Maths.* North-Holland, Amsterdam et al.
- Ghiani, G., & Improta, G., 2000. An algorithm for the hierarchical Chinese postman problem. *Ops Res. Letters* 26 (1), 27–32.
- Glover, F., 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences* 8 (1), 156–166.
- Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Comput. & Ops Res.* 13 (5), 533–549.
- Glover, F., 1989. Tabu search – part I. *ORSA J. on Computing* 1 (3), 190–206.
- Glover, F., 1998a. Genetic algorithms, evolutionary algorithms and scatter search: Changing tides and untapped potentials. Tech. rep., Graduate School of Business, University of Colorado, Boulder, long version of *INFORMS CSTS Newsletter*, 19(1), 1998.
- Glover, F., 1998b. A template for scatter search and path relinking. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., & Snyers, D. (Eds.), *Artificial evolution, AE'97*. Vol. 1363 of *Lecture Notes in Computer Science*. Springer, Heidelberg, pp. 3–51.

- Glover, F., & Laguna, M., 1997. Tabu search. Kluwer Academic Publishers, Boston.
- Glover, F., Løkketangen, A., & Woodruff, D. L., 2000. Scatter search to generate diverse MIP solutions. In: Laguna, M., & Velarde, J. L. G. (Eds.), Computing tools for modeling optimization and simulation. Kluwer Academic Publishers, Boston, pp. 299–320.
- Golden, B. L., DeArmon, J. S., & Baker, E. K., 1983. Computational experiments with algorithms for a class of routing problems. *Comput. & Ops Res.* 10 (1), 47–59.
- Golden, B. L., & Wong, R. T., 1981. Capacitated arc routing problems. *Networks* 11 (3), 305–315.
- Greistorfer, P., 1995. Computational experiments with heuristics for a capacitated arc routing problem. In: Derigs, U., Bachem, A., & Drexl, A. (Eds.), Operations research proceedings 1994. Springer-Verlag, Berlin et al., pp. 185–190.
- Greistorfer, P., 1999. Hybrid genetic tabu search for a cyclic scheduling problem. In: Voß, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.), Meta-Heuristics: Advances and trends in local search paradigms for optimization. Kluwer Academic Publishers, Boston, pp. 213–229.
- Hertz, A., Laporte, G., & Mittaz, M., 2000. A tabu search heuristic for the capacitated arc routing problem. *Ops Res.* 48 (1, Jan-Feb), 129–135.
- Kwan, M.-K., 1962. Graphic programming using odd or even points. *Chinese Mathematics* 1, 273–277.
- Lacomme, P., Prins, C., & Ramdane-Chérif, W., 2001. A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers, E. J. W., Gottlieb, J., Lanzi, P. L., Smith, R. E., Cagnoni, S., Hart, E., Raidl, G. R., & Tijink, H. (Eds.), Applications of evolutionary computing, EvoWorkshops

2001. Vol. 2037 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, pp. 473–483.
- Laguna, M., 1997. Optimization of complex systems with OptQuest. http://www.decisioneering.com/articles/article_index.html.
- Laguna, M., & Glover, F., 1993. Bandwidth packing: A tabu search approach. *Mgt. Sci.* 39 (4), 492–500.
- Lawler, E. L., 1976. Combinatorial optimization: Networks and matroids. Holt, Rinehart & Winston, New York et al.
- Martin, O., Otto, S. W., & Felten, E. W., 1992. Large-step Markov chains for the TSP incorporating local search heuristics. *Ops Res. Letters* 11, 219–224.
- Mechti, R., Poujade, S., Roucairol, C., & Lemarié, B., 1999. Global and local moves in tabu search: A real-life mail collecting application. In: Voß, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.), *Meta-Heuristics: Advances and trends in local search paradigms for optimization*. Kluwer Academic Publishers, Boston, pp. 155–174.
- Osman, I. H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Ops Res.* 41, 421–451.
- Osman, I. H., & Laporte, G., 1996. Metaheuristics: A bibliography. *Ann. Ops Res.* 63, 513–628.
- Osman, I. H., & Wassan, N. A., 2002. A Reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. Tech. rep., School of Business, American University of Beirut, forthcoming in *J. of Scheduling*.
- Pearn, W. L., 1989. Approximate solutions for the capacitated arc routing problem. *Comput. & Ops Res.* 16 (6), 589–600.
- SPEC, 2001. SPEC. Standard performance evaluation corporation. <http://www.spec.org>.
- Taillard, E. D., 1991. Robust taboo search for the quadratic assignment prob-

- lem. *Parallel Comput.* 17, 443–455.
- Thangiah, S. R., Osman, I. H., & Sun, T., 1994. Hybrid genetic algorithm, simulated annealing and tabu search methods for the vehicle routing problem with time windows. Tech. rep., Dept. of Computer Science. Slippery Rock University.
- Wassan, N. A., & Osman, I. H., 2002. Tabu search variants for the mix fleet vehicle routing problem. Tech. rep., School of Business, American University of Beirut, forthcoming in *J. Opl Res. Soc.*
- Woodruff, D. L., & Zemel, E., 1993. Hashing vectors for tabu search. *Ann. Ops Res.* 41, 123–137.