Annals of Operations Research 63(1996)253–275 

# Dynamic tabu search strategies for the traveling purchaser problem

Stefan Voß

*Technische Universität Braunschweig,*
*Abteilung Allgemeine Betriebswirtschaftslehre,*
*Wirtschaftsinformatik und Informationsmanagement,*
*Abt-Jerusalem-Straße 7, D-38106 Braunschweig, Germany*

Tabu search is a metastrategy for guiding known heuristics to overcome local optimality with a large number of successful applications reported in the literature. In this paper we investigate two dynamic strategies, the reverse elimination method and the cancellation sequence method. The incorporation of strategic oscillation as well as a combination of these methods are developed. The impact of the different methods is shown with respect to the traveling purchaser problem, a generalization of the classical traveling salesman problem. The traveling purchaser problem is the problem of determining a tour of a purchaser buying several items in different shops by minimizing the total amount of travel and purchase costs. A comparison of the tabu search strategies with a simulated annealing approach is presented, too.

**Keywords**: Traveling purchaser problem, heuristics, tabu search.

## 1. Introduction

Due to the complexity of a great variety of combinatorial optimization problems, heuristic algorithms are especially relevant for dealing with these problems. Within the field of heuristics recent metastrategies for guiding, e.g., deterministic exchange procedures, have been proven to provide successful tools to overcome local optimality. Following this theme, we investigate the application of the *tabu search* metastrategy for solving a specific combinatorial optimization problem. For further successful applications of this kind of metaheuristic to a wide range of problems reported in the literature see e.g. the two-part survey of Glover [8,9], Glover and Laguna [10], Voß [26], and the contributions in Glover et al. [11].

One of the basic ingredients of tabu search is the *tabu list management*, which concerns updating a tabu list, i.e., deciding on how many and which moves (transitions from a feasible solution to a neighbourhood solution) have to be set tabu within any iteration of the search. In this paper we investigate two dynamic strategies for

managing tabu lists: the *cancellation sequence method* (CSM) and the *reverse elimination method* (REM). The impact of the different methods is shown with respect to the traveling purchaser problem, a generalization of the classical traveling salesman problem.

In section 2 we characterize the traveling purchaser problem and provide ideas for obtaining initial feasible solutions. Furthermore, previous improvement procedures are considered. Section 3 is devoted to the basic ideas of CSM and REM. A combination of these two methods as well as the inclusion of strategic oscillation is described in section 4. Coming back to the traveling purchaser problem in section 5 gives us a suitable field for numerical investigation of the methods under consideration. The paper concludes with some final remarks.

## 2. The traveling purchaser problem

There are numerous extensions and generalizations of the well-known Traveling Salesman Problem (TSP). One generalization of wide applicability is the Traveling Purchaser Problem (TPP) (see Ramesh [22], Golden et al. [12], Ong [18], Voß [24]).

### 2.1. PROBLEM STATEMENT AND APPLICATIONS

The problem can be stated as follows: We are given a set $V = \{1, 2,\ldots, m\}$ of $m$ markets, a depot (domicile or home-market) $s \in V$, and a set $I = \{1, 2,\ldots, n\}$ of $n$ items. For all possible connections between two markets $i, j \in V$, let $c_{ij}$ denote the cost of travel from $i$ to $j$. If item $k$ is available at market $i$, $d_{ik}$ denotes the cost of item $k$ at market $i$, otherwise $d_{ik}$ is set to infinity. The TPP is to find a tour starting and returning to the depot $s$ while visiting a subset of the $m$ markets to purchase each of the $n$ items at one of these markets, such that the total of travel and purchase costs is minimized.

To be clear of pathologies the following is assumed:

- Each item is available in at least one market.
- The traveller/purchaser may pass through a market any number of times without actually purchasing any item there. (This becomes relevant when graphs are considered for representing the set of markets which are not complete.)
- The traveller may purchase as many items as there are available at each market.

In addition, Golden et al. [12] claim that no items are available at the depot. This may be neglected by choosing suitable values $d_{sk}$, so that stock-keeping can be taken into account (but without regarding time-dependent stock-keeping costs, i.e., taking into account consideration of the availability of items at the domicile $s$ at a certain fixed cost per time unit).

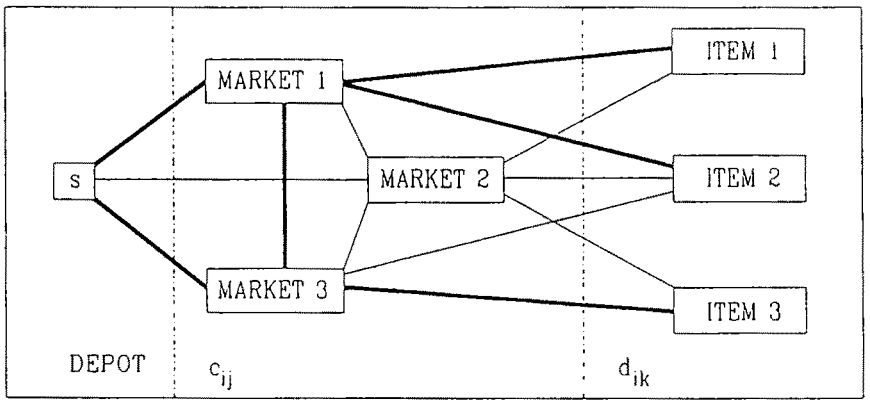


Figure 1. TPP example.

Figure 1 visualizes an example of the TPP with a tour through the depot $s$ and a subset of three additional markets shown in boldface. Each item is available in at least one of the markets the purchaser runs up to.

Applications of the TPP are quite frequent. Besides some obvious applications directly derived from the problem description, another interesting application of the TPP concerns the scheduling of $n$ jobs on an $m$-state machine. To change the state of the machine from state $i$ to state $j$ corresponds to a set-up time of $c_{ij}$, and to process job $k$ at state $i$ corresponds to processing time $d_{ik}$. The problem is to find a processing sequence starting and ending at a certain state $s$, so that each of the $n$ jobs is processed on a specified state of the machine while minimizing the makespan (or maximum completion time), i.e., the total amount of set-up and processing times. This problem is easily verified to be a TPP as market $i$ corresponds to machine state $i$ and item $k$ corresponds to job $k$, whereas the cost values coincide with the set-up and processing times, respectively.

Some further applications are given as the TSP is a reduction of the TPP. This is just the case when $m = n$ and each market carries only one item. This also shows that the TPP is an NP-hard problem since the TSP is one (see Garey and Johnson [7]), indicating that it seems that only problems of moderate size can be solved by an exact procedure.

Since in real-world problems visiting a market (and e.g. queueing up and waiting at the cash-desk) is consuming time that can be associated with some costs, we therefore introduce a fixed cost for visiting each of the markets. For each $i \in V$, let $f_i$ denote the fixed cost of visiting market $i$. Whenever one or more items are purchased at market $i$, one has to take $f_i$ into account. So we are given a traveling purchaser problem with fixed costs. When $f_i = 0$ for all markets $i$, the problem reduces to the TPP. (Furthermore, whenever we deal with complete directed graphs, the $f_i$-values need not be explicitly considered because they may be included within the arc

weights. Note that all procedures described below may equally be applied to directed as well as undirected graphs, despite the fact that our experiments are conducted on data with fixed costs on undirected graphs.)

A somewhat different application can be motivated from architectural aspects of certain ring networks, e.g. in high capacity telecommunications networks with reliability constraints. Here, markets may refer to distribution panels and items to devices or terminals that have to be spanned from the installed distribution panels in a star-like way. Note that there is a close relationship to various generalized Steiner problems in graphs (cf. Voß [25]).

Recalling the TPP as an extension of the TSP gives an obvious but impractical way of solving the problem exactly by applying exact algorithms for the TSP. If we solve a TSP with respect to any subset of the $m$ markets and choose that one which minimizes the total of travel, purchase, and fixed costs, we obtain an optimal solution for the TPP with fixed costs. Since this might not be the best approach for solving the problem exactly and even the problem is NP-hard, in the sequel we refer to heuristics for the TPP. Note that the only exact approach for the TPP we know from the open literature is a lexicographic search procedure given by Ramesh [22], who presents numerical results for problems with up to 12 markets and 10 items (without considering any fixed costs).

## 2.2. FINDING INITIAL FEASIBLE SOLUTIONS

Next we describe the basic ideas of the ADD-procedure and the DROP-procedure that were developed for location models e.g. in Jacobsen [13] and applied to the TPP in Voß [23]. Similar procedures are applied, e.g., for the multiconstraint zero-one knapsack problem in Dammeyer and Voß [4], for clustering problems by Osman and Christofides [20], and for the generalized assignment problem in Osman [19].

The following notation will be used:

$V = \{1,\ldots,m\}$ : set of markets the purchaser may visit,

$V_s := V - \{s\}$,

$V_f \subseteq V$ : set of so-called "forbidden" markets the purchaser does not visit,

$V_r \subseteq V$ : set of so-called "run up" markets the purchaser visits,

$V_u \subseteq V$ : set of so-called "undecided" markets for which it is not yet decided whether the purchaser runs up to them or not.

A partitioning of $V$ into $V_f$, $V_r$ and $V_u$ will be referred to as a *partial solution* iff $V_u$ is non-empty and all items are available at least at one of the markets in $V_r \cup V_u$. A partitioning of $V$ into $V_f$ and $V_r$ ($V_u = \emptyset$) is a *feasible solution* iff all items are available at least at one of the markets in $V_r$. A vector $x = (x_1,\ldots,x_m)$ of binary vari-

ables may be used to indicate which markets are included within a solution ($x_i = 1$) or excluded ($x_i = 0$, i.e. it is "forbidden" to purchase any items at market $i$). That is, any tour through the markets of $V_r$ runs up to exactly those markets $i$ with value $x_i = 1$ (with respect to a solution a market is said to be "run up to" iff it is included in the solution).

The *ADD-procedure* is a greedy heuristic. Initially, $V_r = \{s\}$, $V_u = V_s$, i.e., no market (despite the depot) is run up to, all markets are undecided and the objective function value is $Z = \infty$ (or the sum $\sum_k d_{sk}$, when items are available at the depot). In each step, the procedure *adds* to $V_r$ that remaining element of $V_u$ of the known partial solution which provides the largest reduction of $Z$, such that the new partitioning remains a partial solution or becomes a feasible solution if $V_u = \phi$. If no more reduction of $Z$ is possible, then define $V_f := V_u$, $V_u := \phi$, and the procedure terminates with a feasible solution. Given an initial cycle or (sub)tour $T$, adding a market may be formalized as follows: Determine a market $p \in V_u$ and (adjacent) markets $i, j \in V_r$ such that the following quantity (savings) is maximized over all possible choices of $p$, $i$, and $j$:

$$c_{ij} - (c_{ip} + c_{pj}) + \sum_{k=1}^{n} \max\{\min\{d_{hk} \mid h \in T\} - d_{pk}, 0\} - f_p. \tag{1}$$

The purchaser does not use edge $(i, j)$ any longer (i.e. savings of $c_{ij}$). Instead, he runs up to market $p$ through edges $(i, p)$ and $(p, j)$ (i.e. negative savings of $c_{ip} + f_p + c_{pj}$). Additional savings are obtained for each item that may be purchased at $p$ at a cheaper cost than at any other market of the previous tour $T$.

For ease of description within the subsequent consideration we refer to the process of adding a market to a solution as *ADD-step*. (Note that this notation is independent from the fact whether the solution is feasible or not.) That is, the described ADD-procedure consists of a number of at most $m - 1$ consecutive ADD-steps. Since in each ADD-step no revision of decisions taken in earlier steps with respect to run up markets is allowed, the ADD-procedure may be characterized as a construction method. This procedure has been proposed by Golden et al. [12], whereas modifications concerning a suitable choice of the first market to form an initial tour have been given in Ong [18] and Voß [24]. Here, we build an initial tour $T = (s, i, s)$, where $i \in V_u$ minimizes $c_{si} + \sum_k d_{ik} + f_i + c_{is}$.

Like ADD, the *DROP-procedure* is also a greedy heuristic. Generally speaking, it is an inversion of the ADD-procedure. Initially $V_f = \phi$ and $V_r = V$, i.e., each market is run up to, each item is purchased at the cheapest price and a cycle or tour through all the markets has to be generated. The value of the objective function $Z$ is the sum of the travel costs of the cycle, the cheapest purchase costs of each item and the fixed costs of all markets. Starting with this feasible solution, the procedure in each step *drops* that market which gives the largest reduction of $Z$. If no more reduction is possible, we terminate with a feasible solution. This procedure has been

described as a so-called tour reduction heuristic by Ong [18]. The initial tour is determined by applying some TSP heuristic (e.g. an insertion method by Karg and Thompson [14] and 3-opt by Lin and Kernighan [16]).

Corresponding to the above notation, we refer to the process of dropping a market from a solution as a *DROP-step*. Furthermore, unless explicitly stated, we assume for both, ADD- as well as DROP-steps, that they are performed in a greedy fashion, i.e., the best possible choice is made for the respective step. Note that for a given tour $T$, the effect of a DROP-step may be calculated in a similar way as the savings of an ADD-step in (1) above.

### 2.3. DETERMINISTIC EXCHANGE PROCEDURES

Next we describe deterministic exchange procedures based on the concept of ADD and DROP as they have been proposed by Voß [24]. Given a feasible solution, markets will be exchanged by using ADD- and DROP-steps in the following two approaches:

In the first approach (called IMP1) we start with a partitioning of $V$ into $V_f$ and $V_r$ which represents a feasible solution. With DROP we are searching for that market from $V_r$ whose exclusion gives the best improvement of the objective function value $Z$ or represents the smallest increase of $Z$, if no decrease is possible. With ADD we try to improve the obtained (not necessarily feasible) solution until no more reduction of $Z$ is possible (and the obtained solution by construction is feasible). The ADD-steps will be repeated until one market which leads to an increase of $Z$ would be added within the ADD-part (i.e., ADD is applied as long as improvements of $Z$ are possible). That is, one iteration of IMP1 consists of exactly one DROP-step followed by a number of consecutive ADD-steps. IMP1 is terminated after a complete iteration whenever any market, which had previously been dropped throughout the procedure, was added again in the ADD-part. (For the sake of completeness, let us note as an exception that the depot $s$ will not be dropped, even if no other market is run up to.)

Of course, the above approach may be modified by changing ADD and DROP within the described procedure, resulting in the second approach called IMP2. That is, each iteration of IMP2 consists of exactly one ADD-step followed by a number of DROP-steps.

For numerical results on these methods, the reader is referred to Voß [24]. Here, they will be applied for determining initial feasible solutions and serve as a basis for local search procedures. Based on IMP1 and IMP2 we have two possible *neighbourhood* definitions, where a move corresponds to an iteration of IMP1 or IMP2, respectively. That is, a move consists of exactly one DROP- or ADD-step combined with a number of ADD- or DROP-steps, respectively. Both approaches may be referred to as *combined* DROP- and ADD-procedures (or combined ADD- and DROP-procedures, respectively) and will be applied within local search methods to be described below.

## 2.4. SIMULATED ANNEALING

Simulated annealing is a metaheuristic based on iterative improvement principles and has its origins in the simulation of controlled cooling of material in a heat bath. Based on the algorithm of Metropolis et al. [17], at each iteration of the algorithm a new configuration (neighbourhood solution) is generated by some perturbation of the current one. If the new configuration gives an improvement, i.e. results in a decrease in energy, it is accepted. Otherwise it will be accepted with a certain time-dependent probability, based on some temperature, which goes to zero as the number of iterations increases. A general outline of a simulated annealing procedure for a minimization problem may be described as follows (see, e.g., Dowsland [6] for a recent survey on simulated annealing).

**Given** a feasible solution $x^*$ with objective function value $Z^*$, let $x := x^*$ with $Z(x) = Z^*$, choose an initial temperature $T > 0$ and a temperature reduction factor $\delta \in (0, 1)$.

**Iteration**:

**while** stopping criterion is not fulfilled **do begin**

(1) randomly select an admissible neighbourhood solution $x'$ of $x$ with objective function value $Z(x')$; define $\Delta E := Z(x') - Z(x)$

(2) decide on the acceptance of $x'$:
**if** $\Delta E \leq 0$ **then**
accept $x'$ and perform exchanges: $x := x'$, $Z(x) := Z(x')$,
**if** $Z(x) < Z^*$ **then** $Z^* := Z(x)$, $x^* := x$ **endif**
**else**
accept $x'$ with probability $\exp(-\Delta E/T)$: generate a uniform random number $\gamma \in (0, 1)$;
**if** $\gamma \leq \exp(-\Delta E/T)$ **then** perform exchanges: $x := x'$, $Z(x) := Z(x')$ **endif**
**endif**
**if** no significant change for a specified number of iterations **then** reduce $T$: $T := \delta \cdot T$ **endif**

**endwhile**

**Result**: $x^*$ is the best of all determined solutions, with objective function value $Z^*$.

Following the basic simulated annealing philosophy, the *cooling schedule* is defined by specifying parameter $T$. First an initial value of $T$ is pre-defined at a relatively high value so as to accept most changes in the beginning of the search, and then a scheme for reducing $T$ during the search should be defined. Often, this scheme is denoted as a temperature function together with a number of repetitions (*epoch*) for deciding how many changes for each temperature should be performed.

For an implementation of simulated annealing for the TPP we refer to Voß [24], who applies the above procedure with the following parameter setting. The initial

temperature is chosen to be a problem-dependent value, i.e. $5(n + m)$. This temperature is reduced by a temperature reduction factor of $\delta = 0.6$, and a number of 10 iterations defines an epoch. At the end of every epoch we check for an equilibrium. In the case that an equilibrium has been obtained (i.e., no significant change of the objective function value has occurred throughout all iterations of the most recent epoch; all these objective function values differ only by a sufficiently small nonnegative value, say 0.01, and no improvement of $Z^*$ has occurred) the temperature is reduced. Furthermore, the temperature is reduced at the latest after 20 epochs even if no equilibrium has been obtained. After every fifth iteration, a 3-optimal exchange procedure is applied to the actual tour (because in general finding an optimal tour without considering the items is already an NP-hard problem, and the choice of where to add or drop respective markets is approximate and need not be optimal). Two versions are adapted for comparison in this paper.

In SA1 each iteration consists of exactly one DROP-step and a number of ADD-steps. We first check all possible DROP-steps to find the most improving one deterministically. If no such step exists, a market being dropped is randomly chosen. Then, as long as ADD-steps are improving the solution, they are performed in a greedy fashion. If at a temperature greater than 0.1 (this value was chosen to be different from 0 as a means for a stopping criterion) only deteriorations are possible, a market to be added is randomly chosen. It is determined whether to accept this choice or not as is usual in simulated annealing (see (2) in the outline above). During the whole search we try to keep the six best solutions and after termination of the simulated annealing procedure we check for all these solutions whether they can be improved with IMP1.

The same proceeding applies to SA2, too, but with exchanging DROP- and ADD-steps. That is, whereas the rationale behind SA1 is the principle idea of IMP1 within the context of simulated annealing, for SA2 it is that of IMP2. Therefore, each iteration consists of exactly one ADD-step and a nonnegative number of DROP-steps, whereas only the acceptance of the last DROP-step of each iteration is checked by the simulated annealing process (see (2) in the outline above).

## 3. Tabu search: Dynamic tabu list management

In this section we investigate two dynamic strategies for managing tabu lists: the *cancellation sequence method* (CSM) and the *reverse elimination method* (REM). First some basic ideas of tabu search are described and then CSM and REM are treated.

A transition from a feasible solution to a transformed feasible solution is referred to as a move which may be described by a set of one or more *attributes*. These attributes (properly chosen) can become the foundation for creating an *attribute based memory*. Following a steepest descent/mildest ascent approach, a move may either result in a best possible improvement or a least deterioration of the objective function

value. Without additional control, however, such a process can cause a locally optimal solution to be re-visited immediately after moving to a neighbour, or in a future stage of the search process, respectively.

After providing a move definition with respect to the TPP, the basic ingredients of tabu search as well as the methods CSM and REM will be discussed subsequently.

### 3.1. MOVE DEFINITION

With respect to the TPP, a partitioning of the set $V$ into $V_f$ and $V_r$ may be described by means of a binary vector $x$ with $x_i = 1$ for all markets $i \in V_r$ and $x_i = 0$, otherwise. In this zero-one integer programming context, the attributes may be the set of all possible changes in value assignments for the binary variables. Then two attributes $e$ and $\bar{e}$, which denote that a certain binary variable is set to 1 or 0 (i.e., a market $e$ is added or dropped), may be called *complementary* to each other.

Following the DROP-ADD-approach described in section 2.3, a move from one solution to another consists of dropping one market and then successively adding more markets as long as they improve the objective function value. Correspondingly, we may consider the ADD-DROP-approach. Hence, a move may be represented by more than one attribute referred to as *multi-attribute move*. As such, a multi-attribute move may be completely decomposed into single-attribute moves (adding or dropping a market), the feasibility of a solution may be lost after performing each part of the move. However, this decomposition allows us to use the notation of a *successive multi-attribute move* (cf. Dammeyer and Voß [4]). This kind of move may be useful if an underlying mathematical formulation of a problem consists of inequality constraints or a number of problem-specific objects not known a priori (with respect to the TPP, the number of markets to be included is not known in advance). Another classification of multi-attribute moves distinguishes *static* and *dynamic moves* depending on whether the number of attributes in a move is constant for all iterations or not. With respect to the TPP, the above neighbourhood definition implies the notion of dynamic moves.

### 3.2. TABU SEARCH: BASIC INGREDIENTS

To prevent a search from endlessly cycling between the same solutions, the attribute based memory of tabu search is structured at its first level to provide a *short-term memory function*, which may be visualized to operate as follows. Imagine that the attributes of all moves are stored in a so-called *running list* (RL), representing the trajectory of solutions encountered. Then, based on certain restrictions, a *tabu list* may be defined which implicitly keeps track of salient features of the moves by recording attributes complementary to those of RL. These attributes will be forbidden from being embodied in moves selected in at least one subsequent iteration because their inclusion might lead back to a previously visited solution. Thus, the tabu list restricts the search to a subset of admissible moves (consisting of admissible attributes or

combinations of attributes). The goal is to permit "good" moves in each iteration without re-visiting solutions already encountered. A general outline of a tabu search procedure based on applying such a short-term memory function may be described as follows (for solving a minimization problem like the TPP):

**Given** a feasible solution $x^*$ with objective function value $Z^*$, let $x := x^*$ with $Z(x) = Z^*$.

**Iteration**:

**while** stopping criterion is not fulfilled **do begin**

(1) select best admissible move that transforms $x$ into $x'$ with objective function value $Z(x')$ and add its attributes to the running list

(2) perform tabu list management: compute moves to be set tabu, i.e., update the tabu list

(3) perform exchanges: $x := x'$, $Z(x) = Z(x')$; **if** $Z(x) < Z^*$ **then** $Z^* := Z(x)$, $x^* := x$ **endif**

**endwhile**

**Result**: $x^*$ is the best of all determined solutions, with objective function value $Z^*$.

Evidently, the key to this procedure lies in the *tabu list management* which concerns updating the tabu list, i.e., deciding on how many and which moves have to be set tabu within any iteration of the search. There are in fact several basic ways for carrying out this management, generally involving a *recency based* record that can be individually maintained for different attributes. With respect to dynamic tabu list management strategies, the idea is to handle an attribute in a candidate list related to a sublist of RL first. Via certain criteria, these attributes can be definitely included in the tabu list if necessary, or excluded from the candidate list if possible. The use of different candidate list strategies becomes especially relevant in order to avoid extensive computational effort without sacrificing solution quality.

In our present development, we are going to focus primarily on the preceding short-term framework in order to give special attention to a class of tabu list management strategies that take into account the *dynamic interdependencies* among the memory attributes. These interdependencies, which rely on logical structure as well as on heuristic elements, have frequently been neglected in the tabu search literature. However, as can be deduced from different empirical studies (see, e.g., Domschke et al. [5] and Voß [27]), they can be highly important. Here we investigate CSM and REM as exponents of corresponding methods while incorporating the candidate list ideas implicitly.

### 3.3. THE CANCELLATION SEQUENCE METHOD

The primary goal of CSM as well as REM is to permit the reversion of any attribute (or move) but one between two solutions to prevent re-visiting the older

solution. To find those so-called *critical* attributes (or moves), CSM uses RL as a candidate list (sometimes referred to as *active tabu list*), that contains the complements of attributes that eventually may become tabu. Whenever an attribute of the last performed move finds its complement on RL this complement will be eliminated from RL. All attributes between the cancelled one and its recently added complement build a *cancellation sequence* (*C-sequence* for short) separating the actual solution from the one that has been left by the move that contains the cancelled attribute. Any attribute but one of a C-sequence is allowed to be cancelled by future moves. This condition is sufficient but not necessary to prevent re-visiting previously encountered solutions, as some aspects may have an influence on whether or not CSM works well.

An attribute becomes tabu if its complement is the only attribute of a C-sequence (the set of such attributes is denoted by TABU-C). Furthermore, in general an attribute becomes tabu for one iteration if its complement is the most recent attribute in RL (denoted by TABU-S). Note, however, that making a single attribute tabu prevents many moves which could lead to yet unvisited solutions. For implementational issues concerning CSM, the reader is referred to Dammeyer et al. [2]. An example for the proceeding of CSM is provided below.

Additional specifications help to improve CSM with respect to running times as well as solution quality. For instance, whenever a C-sequence includes a smaller one, the smaller sequence is said to *dominate* the larger. In such a case the larger C-sequence may be neglected, because any of its attributes will only become tabu if they are within the smaller sequence, too. In this respect, a value *last_start* may indicate the iteration number when the starting element of the most recent C-sequence was added to RL. This value may be helpful to evaluate dominance of occurring sequences (compare the example below). Furthermore, throughout the whole search a global *aspiration level criterion* may be defined as the best objective function value computed so far. That is, whenever a move including a tabu attribute would lead to a solution with a better value than previously encountered during the search, this attribute is released from its tabu status.

To clarify the concept of CSM we consider an example visualizing the various concepts discussed so far. With respect to the TPP we are dealing with successive multi-attribute moves. Therefore, we refer to steps instead of iterations, e.g., when defining a tabu status. To simplify matters of presentation when calculating objective function values, without loss of generality, we shall assume the knapsack problem instead of the TPP. Considering the set of binary variables, the definition of the DROP- and ADD-steps for the knapsack problem and the TPP coincide. For the knapsack problem no further ADD-steps are performed, whenever its constraint would be violated, whereas for the TPP this is the case when no further improvement of the objective function value is possible.

We shall choose the data for the knapsack problem, such that the following 11 moves ($\overline{1}\,2, \overline{3}\,1, \overline{2}, \overline{1}\,5, \overline{4}\,1, \overline{5}\,6, \overline{1}\,2, \overline{6}\,4\,3, \overline{2}, \overline{3}\,5, \overline{4}\,3\,2$) of a fictitious TPP instance totalling 22 steps will be performed:

Maximize $10x_1 + 20x_2 + 30x_3 + 35x_4 + 40x_5 + 50x_6$

subject to $6x_1 + 7x_2 + 8x_3 + 10x_4 + 10x_5 + 18x_6 \leq 25, \quad x_i \in \{0,1\} \quad \forall i \in \mathbb{N}_6.$

Figure 2 gives the treatment of this example with respect to CSM for a given starting solution $\boldsymbol{x} = (1, 0, 1, 1, 0, 0)$. A move is defined as complementing the entry of any binary variable. In addition, the corresponding parameters have to be chosen appropriately (e.g. the tabu list duration of a tabu attribute). We assume a static tabu

| step | move | solution | objective | RL | last_start | TABU-S | TABU-C | comment |
|---|---|---|---|---|---|---|---|---|
| 0 | | (1,0,1,1,0,0) | 75 | | | | | |
| 1 | $\bar{1}$ | (0,0,1,1,0,0) | 65 | $\bar{1}$ | 0 | 1 | | |
| 2 | 2 | (0,1,1,1,0,0) | 85 | $\bar{1}$ 2 | 0 | $\bar{2}$ | | |
| 3 | $\bar{3}$ | (0,1,0,1,0,0) | 55 | $\bar{1}$ 2 $\bar{3}$ | 0 | 3 | | |
| 4 | 1 | (1,1,0,1,0,0) | 65 | 2 $\bar{3}$ 1 | 2 | $\bar{1}$ | | |
| 5 | $\bar{2}$ | (1,0,0,1,0,0) | 45 | $\bar{3}$ 1 $\bar{2}$ | 3 | 2 | $3_1$ | |
| 6 | $\bar{1}$ | (0,0,0,1,0,0) | 35 | $\bar{3}$ $\bar{2}$ $\bar{1}$ | 5 | 1 | $3_2$, $2_1$ | |
| 7 | 5 | (0,0,0,1,1,0) | 75 | $\bar{3}$ $\bar{2}$ $\bar{1}$ 5 | 5 | $\bar{5}$ | $3_3$, $2_2$ | |
| 8 | $\bar{4}$ | (0,0,0,0,1,0) | 40 | $\bar{3}$ $\bar{2}$ $\bar{1}$ 5 $\bar{4}$ | 5 | 4 | $3_4$, $2_3$ | |
| 9 | 1 | (1,0,0,0,1,0) | 50 | $\bar{3}$ $\bar{2}$ 5 $\bar{4}$ 1 | 7 | $\bar{1}$ | $2_4$ | |
| 10 | $\bar{5}$ | (1,0,0,0,0,0) | 10 | $\bar{3}$ $\bar{2}$ $\bar{4}$ 1 $\bar{5}$ | 8 | 5 | $4_1$ | |
| 11 | 6 | (1,0,0,0,0,1) | 60 | $\bar{3}$ $\bar{2}$ $\bar{4}$ 1 $\bar{5}$ 6 | 8 | $\bar{6}$ | $4_2$ | |
| 12 | $\bar{1}$ | (0,0,0,0,0,1) | 50 | $\bar{3}$ $\bar{2}$ $\bar{4}$ $\bar{5}$ $\bar{6}$ $\bar{1}$ | 10 | 1 | $4_3$ | |
| 13 | 2 | (0,1,0,0,0,1) | 70 | $\bar{3}$ $\bar{4}$ $\bar{5}$ $\bar{6}$ $\bar{1}$ 2 | 10 | $\bar{2}$ | $4_4$ | dominated: $\bar{4}$ $\bar{5}$ $\bar{6}$ $\bar{1}$ |
| 14 | $\bar{6}$ | (0,1,0,0,0,0) | 20 | $\bar{3}$ $\bar{4}$ $\bar{5}$ $\bar{1}$ 2 $\bar{6}$ | 12 | 6 | $5_1$ | |
| 15 | 4 | (0,1,0,1,0,0) | 55 | $\bar{3}$ $\bar{5}$ $\bar{1}$ 2 $\bar{6}$ 4 | 12 | $\bar{4}$ | $5_2$ | dominated: $\bar{5}$ $\bar{1}$ 2 $\bar{6}$ |
| 16 | 3 | (0,1,1,1,0,0) | 85 | $\bar{5}$ $\bar{1}$ 2 $\bar{6}$ 4 3 | 12 | $\bar{3}$ | $5_3$ | dominated: $\bar{5}$ $\bar{1}$ 2 $\bar{6}$ 4 |
| 17 | $\bar{2}$ | (0,0,1,1,0,0) | 65 | $\bar{5}$ $\bar{1}$ $\bar{6}$ 4 3 $\bar{2}$ | 14 | 2 | $5_4$, $1_1$ | |
| 18 | $\bar{3}$ | (0,0,0,1,0,0) | 35 | $\bar{5}$ $\bar{1}$ $\bar{6}$ 4 $\bar{2}$ $\bar{3}$ | 17 | 3 | $1_2$, $2_1$ | |
| 19 | 5 | (0,0,0,1,1,0) | 75 | $\bar{1}$ $\bar{6}$ 4 $\bar{2}$ $\bar{3}$ 5 | 17 | $\bar{5}$ | $1_3$, $2_2$ | dominated: $\bar{1}$ $\bar{6}$ 4 $\bar{2}$ $\bar{3}$ |
| 20 | $\bar{4}$ | (0,0,0,0,1,0) | 40 | $\bar{1}$ $\bar{6}$ $\bar{2}$ $\bar{3}$ 5 $\bar{4}$ | 17 | 4 | $1_4$, $2_3$, $6_1$ | domination/last_start |
| 21 | 3 | (0,0,1,0,1,0) | 65 | $\bar{1}$ $\bar{6}$ $\bar{2}$ 5 $\bar{4}$ 3 | 19 | $\bar{3}$ | $2_4$, $6_2$ | |
| 22 | 2 | (0,1,1,0,1,0) | 90 | $\bar{1}$ $\bar{6}$ 5 $\bar{4}$ 3 2 | 19 | $\bar{2}$ | $6_3$ | aspiration criterion |

Figure 2. Example for CSM (C-sequences appear underlined).

list of length 1 (TABU-S) and a dynamic tabu list (TABU-C) with a maximum tabu duration of 4 for each tabu element (a subscript at a tabu element indicates its current tabu duration). C-sequences appear underlined. In the sequel we comment on some of the interesting aspects that may be observed.

Step 9 gives an example of too many tabu attributes. The tabu status for 3 and 2 is not necessary as they could have been chosen within the next two steps (i.e. steps 9 and 10) without repeating any previously encountered solution and resulting in the optimal solution. In step 15 the attribute 4 is no longer tabu, resulting in the same solution as in step 3. Nevertheless, we do not get stuck in a cycle as the tabu lists are different, and the search trajectory leads to new solutions lateron. Step 20 gives an example for the use of last_start. The performed attribute cancels its complement 4, resulting in a new C-sequence which is not directly dominated by any other sequence. However, the complement of one of the attributes, i.e. element 2, is tabu and represents a one-element sequence that has not been reversed due to its underlying tabu status (visualized in the value last_start). In step 22 the optimal solution is obtained utilizing a global aspiration level criterion.

### 3.4. THE REVERSE ELIMINATION METHOD

The conditions of CSM need not be necessary to prevent re-visiting previously encountered solutions. In general, however, necessity can be achieved by REM. Its idea is that any solution can only be re-visited in the next iteration if it is a neighbour of the current solution. Therefore, in each iteration the running list will be traced back to determine all moves which have to be set tabu since they would lead to an already explored solution. For this purpose, a *residual cancellation sequence* (RCS) is built up stepwise by tracing back the running list. In each step exactly one attribute is processed, from last to first. After initializing an empty RCS, only those attributes are added whose complements are not in the sequence. Otherwise their complements in the RCS are eliminated (i.e. cancelled). Then at each tracing step it is known which attributes have to be reversed in order to turn the current solution back into one examined earlier during the search. If the remaining attributes in the RCS can be reversed by exactly one move, then this move is tabu in the next iteration. For single-attribute moves, for instance, the length of an RCS must be one to enforce a tabu move.

For an example visualizing the proceeding of REM we refer to figure 3. Let a *position* denote an attribute's location in the running list. The figure gives an example for building residual cancellation sequences. We assume three moves following the combined DROP- and ADD-procedure, i.e. dropping market 4 and adding markets 7, 3, and 1, dropping 5 and adding 6 and 4, and exchanging 6 and 5. As we have defined dynamic moves for the TPP consisting of multiple attributes in any iteration, the number of traces to be performed is equal to the number of attributes of the corresponding move. Since we are dealing with successive multi-attribute moves, however, we deal with exactly one attribute at a time and the length of an RCS has to become equal to one to enforce a tabu move. Note that a tabu status only refers to the next step and not to a whole iteration.

Running list: $\overline{4}\ 7\ 3\ 1\ \overline{5}\ 6\ 4\ \overline{6}\ 5$ (latest attribute: 5)

| tracing step | position | residual cancellation sequence | length | tabu move |
|---|---|---|---|---|
| 1 | 9 | $5$ | 1 | $\overline{5}$ |
| 2 | 8 | $\overline{6}\ 5$ | 2 | - |
| 3 | 7 | $4\ \overline{6}\ 5$ | 3 | - |
| 4 | 6 | $4\ 5$ | 2 | - |
| 5 | 5 | $4$ | 1 | $\overline{4}$ |
| 6 | 4 | $1\ 4$ | 2 | - |
| 7 | 3 | $3\ 1\ 4$ | 3 | - |
| 8 | 2 | $7\ 3\ 1\ 4$ | 4 | - |
| 9 | 1 | $7\ 3\ 1$ | 3 | - |

Tabu list: $\overline{4}\ \overline{5}$

Figure 3. Example for RCS development.

Obviously, in general, the execution of REM represents a necessary and sufficient criterion to prevent re-visiting known solutions. With respect to the TPP, however, sufficiency of REM is not guaranteed due to the move definition described above. The tabu status only prevents re-visiting the same set of markets but does not refer to the sequence in which the markets are considered within a tour. Therefore, introducing a parameter limiting the size of RL seems to be helpful.

Since the computational effort of REM increases, if the number of iterations increases, ideas for reducing the number of computations have been developed (cf. Glover [9] and Dammeyer and Voß [4]). In principle, REM is not restricted in belonging into the short-term memory framework as presented in section 3.2. If enough storage is available this proceeding may go on without reservation. Usually, storage requirements have to be limited. In that sense a parameter *at_n* (the only one within REM) may be defined as the number of attributes that may be stored within the running list, i.e. the length of RL.

For applications and comparisons of a static tabu search approach, CSM, and REM as well as specific simulated annealing implementations, see Dammeyer and Voß [3,4] and Domschke et al. [5].

## 4. Some advanced issues

In the previous section we have described the basic ingredients of two tabu search methods that explicitly observe logical interdependencies encountered throughout the search. In this section a combination of these two methods, the inclusion of search intensification and search diversification, as well as strategic oscillation are considered.

### 4.1. SEARCH INTENSIFICATION AND SEARCH DIVERSIFICATION

Recalling the use of the running list, the basic idea of tabu search is to keep sufficient information of the search within some memory. Many applications of tabu search introduce memory structures based on *frequency*, and the coordination of these memory elements is made to vary as the above short-term memory component becomes integrated with longer term components. The purpose of this integration is to provide a balance between two types of globally interacting strategies, called *intensification strategies* and *diversification strategies*. For details of such considerations, see for example the survey by Glover and Laguna [10].

Considering the recent search history a general idea for reducing the computational effort within tabu search is that of *search intensification* using a so-called frequency based *short-term memory* (STM). Its basic idea is to observe the attributes of all performed moves or a subset of them. Those attributes, that have not been part of any solution generated during a given number of iterations, are eliminated from further consideration. This results in a concentration of the search in the neighbourhood of frequently used solutions. Consequently the computational effort will decrease, eventually with a loss of accuracy.

Correspondingly, a *search diversification* strategy may be based on some frequency based *long-term memory* (LTM) to penalize often selected assignments. Then the neighbourhood search can be led into not yet explored regions where the tabu list operation is restarted (resulting in an increased computation time). The basic intention of search diversification is to explore regions of the search space that previously have not been investigated in-depth, whereas search intensification tries to search within a limited space, e.g. in a promising region.

Both memory notions as well as combinations (L + STM) refer to some kind of frequency measure, where its success depends on the range of the running list (whenever it is carried along with the search). However, they do not refer to the quality of solutions as well as the attraction around local optima. Therefore, more evolved ideas may be considered in different settings. Since the main idea of human problem solving might be the foundation for the frequency based memory, this process appears to rely on combinations of different types of memory functions also incorporating a time-dependent measure of frequency (a so-called recognition value). According to Glover and Laguna [10], such considerations may lead to even more intelligent strategies for capturing successful and advantageous ingredients of tabu search on a broader level. They, as well as Voß [26,27], classify additional concepts of intensification and diversification not only depending on frequency based memory, too.

A careful observation of CSM and REM reveals that the latter method has more difficulties to thoroughly evaluate a larger solution space at different places than CSM. This is due to the fact that CSM might advise a tabu status to more attributes than necessary, while REM provides sufficiency as well as necessity. Following this argument, the following combination of both methods (CSM + REM) becomes obvious.

Whenever CSM terminates according to some stopping criterion, the best found solution may be the starting solution for REM, which should search in the near vicinity for even better solutions that might have been overlooked by CSM. That is, both algorithms are combined in a hierarchical way with the output of CSM being the input of REM. Whilst the parameter setting of CSM is forwarded immediately to REM (but with an empty running list) with most specifications, for LTM the previous frequency memory is ignored.

### 4.2. STRATEGIC OSCILLATION

*Strategic oscillation* (SO) or *tabu tunneling* describes approaches that allow for infeasibilities throughout the search. These infeasibilities may become necessary when considering problems with restrictive equality constraints and focusing on successive multi-attribute moves. They can also become a helpful tool by investigating not yet visited regions that are not reachable by the corresponding neighbourhood definition. In that sense, they may have an important role within the interplay of search intensification and search diversification. If the move definition refers to successive multi-attribute moves, like in the combined DROP-ADD-approach for the TPP, then this definition allows for intermediate infeasibilities for a certain number of attributes and returns to feasibility when performing the whole move. For instance, when dropping a market it may occur that items are no longer available at the remaining markets. That is, we already included a somehow intrinsic nature of SO into our DROP-ADD neighbourhood definition with respect to the TPP. (An example may be visualized with respect to figure 1 above. If market 1 is dropped, resulting in a tour $T = (s - \text{market } 3 - s)$, item 1 is no longer available in any market of $T$. After adding market 2, we again obtain a feasible solution where all items are available.)

Here, we go even further in that we propose to combine CSM and REM with the following proceeding. Assume the above definition of ADD and DROP, then we apply successive multi-attribute moves. For a certain number of steps, say 5, we drop exactly one market (as long as there are enough markets in the solution). Then we perform as many ADD-steps as they give improvements of the objective function value. (With respect to CSM, these ADD-steps shall be performed while maintaining the RL but with an empty tabu list for the first ADD-step. For REM the tabu attributes shall be calculated as usual.) The oscillation proposed by this proceeding simply allows for dropping markets although the remaining markets do not carry all items that have to be purchased.

For further references on strategic oscillation, the reader is referred to, e.g., Dammeyer and Voß [3], Glover and Laguna [10] and Kelly et al. [15].

## 5. Numerical results for the traveling purchaser problem

In this section we report our numerical results for the tabu search strategies described in the previous sections when applied to the TPP. Before presenting the

results in detail, we discuss the underlying data and the way initial feasible solutions have been obtained.

Numerical results for the TPP have been computed for three graphs known from the literature for different routing problems with 10, 31 (cf. Clarke and Wright [1]) and 52 (cf. Paessens and Weuthen [21]) markets. For each graph we varied the number of items eight times depending on the number of markets (2, 4,...,16 items for the 10-market example, 6, 12, 19, 25, 31, 37, 43, and 50 for the 31-market example, and 10, 21, 31, 42, 52, 62, 73, and 83 items for the 52-market example). The fixed costs were randomly generated from a uniform distribution within [0, 100]. To simulate different structures of markets to be visited we assumed a certain probability for each item to be available at any market. Four different settings, i.e. values 0.25, 0.4, 0.6, and 0.75 were assumed for this probability. The item costs were randomly generated from a uniform distribution. The availability of each item at each market was checked by a random number and, depending on that result, the item cost was either set to a prohibitively high value or randomly chosen within [0, 1000]. To take another structure into consideration, the item costs were also chosen within [0, 100] to get a proportion of 1 between item costs and fixed costs in contrast to a proportion of 10. With these proportions we have two structures with different chances of consideration for markets located at a distance far apart from the depot against nearby ones. To sum up we have 192 test problems (3 graphs, 8 numbers of items, 4 availabilities, and 2 settings for item costs).

For each problem, six initial feasible (or partial) solutions have been computed, resulting in an overall testbed of 1152 starting solutions over all 192 test problems:

(1) The first solution is a partial one consisting only of the depot.

(2) The second solution is obtained by applying the ADD-procedure.

(3) The third solution is obtained by applying the ADD-procedure, too. The difference is that each time after having performed five ADD-steps and at the end of the procedure, the resulting tour is re-optimized by applying 3-opt.

(4) For the fourth solution, again the ADD-procedure is applied in a modified way, until all markets are included into the solution.

(5) The fifth solution is obtained by applying the DROP-procedure or tour reduction method, respectively.

(6) The sixth solution is obtained by applying the DROP-procedure, too. The difference is that each time after having performed five DROP-steps and at the end of the procedure the resulting tour is re-optimized by applying 3-opt.

The procedures that we are investigating are the deterministic exchange procedures IMP1 and IMP2 of section 2.3, the simulated annealing versions SA1 and SA2 that are superimposed on the DROP-ADD- (i.e. the basic move definition within IMP1) and the ADD-DROP-approach (i.e. IMP2), respectively. For the tabu search

metastrategy, both possibilities have been tested, too. That is, the methods CSM and REM, the combined CSM + REM, and CSM as well as REM with SO are combined with both exchange possibilities. Here, due to the better quality of the obtained results, we restrict ourselves to the presentation of the DROP-ADD-approach as the foundation for performing exchanging moves (the ADD-DROP-approach did not give better results).

Considering the general framework of tabu search as presented above, we need to describe a termination criterion. In tabu search we refer to an *epoch* as a certain number of iterations, similar to that used in simulated annealing (see section 2.4). For each of the six initial solutions, each algorithm terminates whenever there is no improvement of the best feasible solution (obtained by this algorithm for this starting solution) within all iterations of a certain number of successive epochs, say two. The length of an epoch starting from an initialized value of 10 is increased over time (i.e. after each epoch) by an increase factor which was chosen to be equal to 1.1. In addition, an overall termination criterion might be applied (for our examples, however, two different choices of $10n$ and of $50n$ for this criterion affected neither the termination of tabu search nor the solution quality).

For all tabu search methods, moves are defined as successive multi-attribute moves as described above. Within CSM, a tabu attribute stays tabu for the next eight steps. For REM we choose $at\_n = 4n$. Combining REM and CSM is performed in a very simple way with the rationale being some kind of search intensification, i.e., the best solution obtained through CSM is taken as starting solution for the application of REM.

The different memory functions work as follows. For LTM we calculate at most nine new starting solutions. That is, each time the respective tabu search procedure (CSM, REM, etc.) is applied in the usual way until termination, whereas all markets that have not been in any of the solutions encountered, are retained. Then, each time a new starting solution is generated, it includes the home depot together with all markets retained since the last restart. If two restart solutions are identical when following this proceeding, the algorithm is stopped. For STM, after each epoch, those markets that have not been included in any solution of the previous epoch are definitely excluded. For L + STM we have implemented the following modification. Whenever the corresponding method with STM is stopped, a new starting solution is obtained by choosing from those elements that have previously been eliminated according to STM, and the method with STM is restarted. This procedure is applied no more than ten times (i.e. the initial run and up to nine restarts) as long as a new starting solution can be found.

Table 1 summarizes our numerical investigations. It gives some relevant characteristics for all different algorithms. For each of the 192 test cases its overall best solution (with objective function value "best") obtained by any of the algorithms is taken as a reference solution. The first two columns show the percentage of best solutions found referring to the 192 problem instances and referring to all combinations

Table 1

Numerical results for the TPP.

| Method | Best solutions ref. 192 | Best solutions ref. 1152 | Average deviation ref. 192 | Average deviation ref. 1152 | Av. impr. ref. 1152 | Iter (Σ) | CPU time ref. 6 (Σ) |
|---|---|---|---|---|---|---|---|
| IMP1 | 69.79 | 49.40 | 0.570 | 1.713 | 7.818 | 6 | 10.44 |
| IMP2 | 70.83 | 51.04 | 0.696 | 1.600 | 7.866 | 3 | 2.11 |
| SA1 | 83.34 | 66.49 | 0.333 | 1.044 | 8.341 | 208 | 159.86 |
| SA2 | 88.02 | 72.66 | 0.228 | 0.773 | 8.532 | 181 | 69.25 |
| CSM | 96.88 | 88.11 | 0.042 | 0.150 | 8.995 | 116 | 92.58 |
| CSM / STM | 95.83 | 84.98 | 0.078 | 0.271 | 8.885 | 111 | 57.65 |
| CSM / LTM | 99.48 | 96.18 | 0.041 | 0.064 | 9.069 | 636 | 595.03 |
| CSM / L+STM | 98.96 | 96.27 | 0.041 | 0.076 | 9.058 | 881 | 617.37 |
| REM | 95.31 | 84.29 | 0.038 | 0.274 | 8.884 | 113 | 96.02 |
| REM / STM | 93.23 | 80.12 | 0.070 | 0.466 | 8.748 | 105 | 56.15 |
| REM / LTM | 98.44 | 92.88 | 0.013 | 0.099 | 9.040 | 518 | 528.31 |
| REM / L+STM | 97.40 | 92.88 | 0.023 | 0.092 | 9.053 | 706 | 537.71 |
| CSM+REM | 97.92 | 91.23 | 0.001 | 0.120 | 9.017 | 179 | 140.48 |
| CSM+REM / STM | 97.40 | 88.28 | 0.075 | 0.232 | 8.917 | 174 | 89.51 |
| CSM+REM / LTM | 100 | 97.05 | 0.000 | 0.045 | 9.080 | 698 | 639.52 |
| CSM+REM / L+STM | 98.96 | 96.53 | 0.041 | 0.070 | 9.061 | 942 | 645.87 |
| CSM-SO | 96.88 | 87.15 | 0.067 | 0.159 | 8.992 | 158 | 154.46 |
| CSM-SO / STM | 96.35 | 86.46 | 0.224 | 0.224 | 8.939 | 158 | 116.48 |
| CSM-SO / LTM | 99.48 | 95.23 | 0.000 | 0.030 | 9.099 | 698 | 909.52 |
| CSM-SO / L+STM | 100 | 96.88 | 0.000 | 0.039 | 9.091 | 1029 | 1091.78 |
| REM-SO | 98.96 | 91.32 | 0.041 | 0.136 | 9.006 | 122 | 153.45 |
| REM-SO / STM | 97.92 | 89.06 | 0.033 | 0.199 | 8.960 | 118 | 103.92 |
| REM-SO / LTM | 99.48 | 97.05 | 0.000 | 0.053 | 9.079 | 577 | 905.54 |
| REM-SO / L+STM | 100 | 97.92 | 0.000 | 0.017 | 9.100 | 697 | 832.21 |

of instances and initial runs. That is, the first column for each algorithm gives the percentual coincidence of its minimal objective function value over the six solutions with the respective values of best. In the second column the respective percentual coincidence referring to all $6 \times 192$ solutions is given. Correspondingly, the average deviation from the overall best known objective function values is given with respect to the best found solution out of the six instances over the 192 test problems and referring to all 1152 instances. Furthermore, we have added a column "av. impr." that gives the average improvement over the respective starting solutions in percent. The average number of moves (column "iter") gives the average number of iterations to find the best solution for each problem, i.e., with respect to the best found solution over all 192 test problems, however, summed over all six starting solutions. All programs are implemented in PASCAL and the CPU times refer to a 386 personal computer (with-

out coprocessor, 33 MHz). Each entry is the average value over all problems indicating the running time necessary for all six starting solutions and the corresponding search. (For clarification, to obtain average numbers for single runs the entries in the latter two columns have to be divided by six.)

The results show that simulated annealing is outperformed by the different tabu search strategies. CSM shows a slightly better behaviour with respect to the solution quality compared to REM. However, the more elaborate tabu list management within REM does not imply higher CPU times. Whereas the consideration of LTM takes large CPU times, the solution quality is enhanced. L + STM, however, does not give the results one would like to see. Opposite to the results for the multiconstraint zero-one knapsack problem (cf. Dammeyer and Voß [3]) we obtained improvements by applying the simple combination of CSM and REM. Strategic oscillation (SO) takes a lot of time but improves the solutions again with some advantages for REM-SO against CSM-SO.

Having investigated a number of 1152 reference solutions for our tests, we still have to admit the probably occurring simplicity of the instances due to the fact that almost 50% of all best solutions have already been obtained by IMP1 and IMP2, respectively. Nevertheless, the ranking within the methods gives enough ideas for recognizing their behaviour with respect to the TPP. This is especially relevant for the results with respect to both simulated annealing implementations when comparing them with any of the tabu search methods. Furthermore, we investigated some obvious modifications of the parameter settings which led to almost the same picture as the results shown in table 1. We report some of our findings with respect to the 31-market example with item costs ranging between 0 and 1000.

Varying the tabu duration with respect to CSM within 7 and 14 had nearly no influence on the solution quality, but the CPU times were effected by a range of 20% when applying the strategic oscillation option and by at most 10% without SO. Slightly larger values of the increase factor caused CPU times to be increased up to 30% with only slight improvements in the solution quality (which seems to be obvious due to the already good solutions obtained with the basic parameter setting). Modifying the stopping criterion, such that the algorithms terminate whenever there is no improvement within all iterations of a single epoch, gave a considerable reduction of the CPU times (about 30–40%) but with a considerably worse solution quality, especially for those methods incorporating strategic oscillation.

With respect to the numerical results some additional observations may lead to obvious questions that have been investigated through additional tests not shown within table 1. First, the deterministic exchange procedure IMP2 slightly outperforms IMP1 and the same dominance holds between SA2 and SA1. However, for the tabu search methods the opposite holds and the rationale for this behaviour is not quite clear. Provided that we have a fixed time limit, the question is whether this picture changes. As remarked above, we tried a fixed problem-dependent time limit allowing for at least $10n$ iterations for each method without getting any different results. When

the time limit is reduced the dominance of the tabu search methods over simulated annealing referring to the best of the six solutions encountered becomes more clear (ref. 192), although with respect to the columns ref. 1152 this is not the case. Additional tests for simulated annealing with slightly modified cooling schedules allowing enlarged CPU times comparable to those of LTM without SO, however, did not change this picture.

For larger graphs with up to 60 markets and up to 120 items, the picture with respect to the best solutions becomes clearer in that IMP1 and IMP2 turn out to produce weaker solutions. However, the relation between the simulated annealing and the tabu search approaches does not change. Therefore, in a final test we tried to incorporate additional restrictions with respect to the TPP and it turned out that the tabu search methods seem to have more serious difficulties in finding good solutions when capacity constraints come into play. A preliminary test in this respect gives some advantages for simulated annealing whenever the number of items allowed for being purchased at any market is limited. This, however, needs to be studied more thoroughly to provide some more evolved reasoning.

## 6. Conclusions

In this paper we have investigated tabu list management strategies that take account of dynamic interdependencies among memory attributes. These interdependencies, which rely on logical structure as well as on heuristic elements, have frequently been neglected in the tabu search literature. Even Glover and Laguna [10] state in their recent survey on tabu search that "dynamic rules for determining tabu tenure are among the aspects of tabu search that deserve more study". Therefore, the considerations of different ideas and concepts within tabu search presented in this paper mainly refer to the surroundings of these interdependencies.

Different studies reveal that the methods considered here may also improve on the best known solutions for well-known benchmark problems in different settings (see, e.g., the results in Voß [27] for the quadratic assignment problem). Further investigations incorporating additional features of tabu search might be of interest (see, e.g., some of the additional proposals of Glover and Laguna [10]).

Numerical investigations have been performed with respect to the traveling purchaser problem. This problem has been encountered when observing certain manufacturing processes, i.e., the scheduling of a number of jobs on a multi-state machine (an oven where different temperature options may arise for the jobs), and in telecommunications. Further research with respect to the TPP might be the development of a more evolved exact (e.g. branch and bound) algorithm as the one presented by Ramesh [22]. Logical tests and preprocessing techniques might be helpful in both settings, i.e., when considering approximate as well as exact solution procedures. Furthermore, a more thorough investigation of additional capacity constraints as mentioned above needs to be carried out.

## Acknowledgement

The author gratefully appreciates the helpful and constructive comments of Ibrahim H. Osman and Frank Dammeyer that helped improve the presentation of this paper.

## References

[1] G. Clarke and J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12(1964)568–581.

[2] F. Dammeyer, P. Forst and S. Voß, On the cancellation sequence method of tabu search, ORSA Journal on Computing 3(1991)262–265.

[3] F. Dammeyer and S. Voß, Application of tabu search strategies for solving multiconstraint zero-one knapsack problems, Working Paper, TH Darmstadt (1991).

[4] F. Dammeyer and S. Voß, Dynamic tabu list management using the reverse elimination method, Annals of Operations Research 41(1993)31–46.

[5] W. Domschke, P. Forst and S. Voß, Tabu search techniques for the quadratic semi-assignment problem, in: *New Directions for Operations Research in Manufacturing*, ed. G. Fandel, T. Gulledge and A. Jones (Springer, Berlin, 1992) pp. 389–405.

[6] K.A. Dowsland, Simulated annealing, in: *Modern Heuristic Techniques for Combinatorial Problems*, ed. C.R. Reeves (Blackwell, Oxford, 1993) pp. 20–69.

[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).

[8] F. Glover, Tabu search - part I, ORSA Journal on Computing 1(1989)190–206.

[9] F. Glover, Tabu search - part II, ORSA Journal on Computing 2(1990)4–32.

[10] F. Glover and M. Laguna, Tabu search, in: *Modern Heuristic Techniques for Combinatorial Problems*, ed. C.R. Reeves (Blackwell, Oxford, 1993) pp. 70–150.

[11] F. Glover, M. Laguna, E. Taillard and D. de Werra (eds.), *Tabu Search*, Annals of Operations Research 41 (Baltzer, Basel, 1993).

[12] B. Golden, L. Levy and R. Dahl, Two generalizations of the traveling salesman problem, Omega 9(1981)439–441.

[13] S.K. Jacobsen, Heuristics for the capacitated plant location model, European Journal of Operational Research 12(1983)253–261.

[14] R.K. Karg and G.L. Thompson, A heuristic approach to solving traveling salesman problems, Management Science 10(1964)225–248.

[15] J.P. Kelly, B.L. Golden and A.A. Assad, Large-scale controlled rounding using tabu search with strategic oscillation, Annals of Operations Research 41(1993)69–84.

[16] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operations Research 21(1973)498–516.

[17] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, Equation of state calculation by fast computing machines, Journal of Chemical Physics 21(1953)1087–1091.

[18] H.L. Ong, Approximate algorithms for the travelling purchaser problem, Operations Research Letters 1(1982)201–205.

[19] I.H. Osman, Heuristics for the generalised assignment problem: Simulated annealing and tabu search approaches, OR Spektrum 17(1995)211–225.

[20] I.H. Osman and N. Christofides, Capacitated clustering problems by hybrid simulated annealing and tabu search, International Transactions in Operational Research 1(1994)317–336.

[21] H. Paessens and H.K. Weuthen, Tourenplanung in städtischen Straßennetzen mit einem heuristischen Verfahren, in: *Tourenplanung bei der Abfallbeseitigung*, ed. H.H. Hahn (Schmidt, Bielefeld, 1977) pp. 101–130.

[22] T. Ramesh, Travelling purchaser problem, Opsearch 18(1981)78–91.
[23] S. Voß, ADD- and DROP-procedures for the travelling purchaser problem, Methods of Operations Research 53(1986)317–318.
[24] S. Voß, The traveling purchaser problem with fixed costs, Working Paper, TH Darmstadt (1989).
[25] S. Voß, *Steiner-Probleme in Graphen* (Hain, Frankfurt am Main, 1990).
[26] S. Voß, Intelligent search, Manuscript, TH Darmstadt (1993).
[27] S. Voß, Solving quadratic assignment problems using the reverse elimination method, in: *The Impact of Emerging Technologies on Computer Science and Operations Research*, ed. S.G. Nash and A. Sofer (Kluwer, Dordrecht, 1995) pp. 281–296.