Chapter

Solving General Ring Network Design Problems by Meta-Heuristics

Andreas Fink, Gabriele Schneidereit, Stefan Voß
Technical University of Braunschweig; Department of Business Administration, Information Systems and Information Management; Abt-Jerusalem-Str. 7, D-38106 Braunschweig, Germany; email: {a.fink, g.schneidereit, stefan.voss}@tu-bs.de

Key words: Ring networks, Meta-heuristics

Abstract: Ring network design problems have many important applications, especially in the field of telecommunications and vehicle routing. Those problems generally consist of constructing a ring network by selecting a node subset and corresponding direct links. Different requirements and objectives lead to various specific types of *NP*-hard ring network design problems reported in the literature, each with its own algorithms. We exploit the similarities in problems to produce a more general problem formulation and associated solution methods that apply to a broad range of problems. Computational results are reported for an implementation using a meta-heuristics framework with generic components for heuristic search.

1. INTRODUCTION

Recently, the design of information and communication infrastructure has become a major challenge both within companies and between widespread places, e.g., in major cities where metropolitan area networks are of interest. High-bandwidth fiber optic networks occupy an intermediate position between local area networks (LANs) and wide area networks. Among the various topologies available to the design of such networks, ring networks may be beneficial because they provide some protection against link failures (Morreale and Campbell 1990).

In this paper we consider a *General Ring Network Design Problem* (*GRNDP*), which may be described as follows (from the perspective of telecommunications): Given is a set of nodes representing objects that may be linked to a network (e.g., routing devices, companies, public places of interest with some means of public information provision). Any two nodes on the ring are able to communicate with each other so one gains a certain revenue. Moreover, there may be revenues for each node included in the ring. On the other hand, construction costs are incurred for the design of direct links. The basic objective is to maximize the sum of all revenues minus the construction costs while building a ring network. Possibly, this ring must meet additional requirements such as mandatory inclusion of a subset of required nodes or upper bounds for the number of nodes in the ring or the sum of the link costs (e.g., due to network reliability demands). There are several generalizations of the Traveling Salesman Problem (see below) that can be treated as ring network design problems.

Our main intent is as follows. We will integrate different types of problems, which have so far been treated separately in the research literature, into a generalized model. Since most of the problems subsumed under this model defy exact solution with reasonable computational effort (due to *NP*-hardness), the modeling of the problem is often an approximate one (concerning, e.g., the objective), and the data are generally imprecise, heuristics are the primary way to tackle these problems. Furthermore, even our generalized model does not include many practical situations; due to variations, e.g., with respect to the pursued objective or specific problem characteristics, ‘solving’ problems from practice often requires special purpose methods. Thus, we are faced with the difficulty of efficiently adapting and applying appropriate methods to real-world problems. Our aim is to make it easy to apply general yet effective heuristics to such problems. We focus on the application of meta-heuristics by means of HOTFRAME (Fink and Voß 1999), which provides a collection of adaptable reusable software components for heuristic search. That is, we show that the idea of a general heuristic solver linking various problem classes is reasonable.

In Section 2, we formally define the GRNDP and discuss certain relationships to other problems as they have appeared in the published literature. In Section 3, we discuss the application of meta-heuristics to the GRNDP. In addition to describing different construction methods, we present an appropriate neighborhood structure and describe the application of simulated annealing and various tabu search procedures. Computational experiments are presented in Section 4. These results demonstrate the possibility of effectively and easily applying generic and robust meta-heuristics with no calibration necessary. Finally, we draw some conclusions and give hints for further research.

2. PROBLEM DESCRIPTION

In the following we formally define the *General Ring Network Design Problem* (*GRNDP*), which is later shown to subsume various special types of problems previously discussed in the literature. Given a graph G with node or vertex set $V=\{1,...,n\}$ and edge set E with non-negative edge weights c_{ij} for all $(i,j) \in E$, the ring network design problem is defined by a vector $(G=(V,E), c, r, p, Q, a, b, h, \alpha, \beta, \gamma, \delta)$. A solution of a problem instance is a ring $R=(i_1,...,i_k)$ with corresponding links $((i_1,i_2), ..., (i_{k-1},i_k), (i_k,i_1))$. Whenever a ring contains a direct link between i and j, a cost c_{ij} is incurred. That is, only fixed construction costs are taken into account, and the sum of construction costs of R is given by $c(R)=\sum_{j=1}^{k-1} c_{j,j+1} + c_{k1}$.

For every pair of nodes i,j, $i<j$, a revenue r_{ij} is obtained if and only if i and j both belong to the ring. For each node i, r_{ii} represents a revenue for the inclusion of node i in the ring, and p_i represents a penalty for not including node i.

Additional requirements may restrict the set of feasible solutions. The set Q defines a subset of required nodes that must be included in the ring. The value a may set a lower bound for the revenues to be obtained. A budget b may limit the ring costs by $c(R) \leq b$. The number h may set an upper bound for the number of nodes in a ring.

The objective is to find a feasible ring R that maximizes the objective function. Parameters α, β, γ, and δ define the composition of the objective function, e.g., as the sum of the revenues by node pairs (α = 1), revenues by included nodes (β = 1), minus the construction costs (γ = 1), as well as penalties for nodes not included (δ = 1). The GRNDP, which is *NP*-hard as it reduces to several *NP*-hard problems (see below), is represented by the following mathematical model, which builds on ideas of Gendreau et al. (1995). Define $x_{ij}=1$, if edge (i,j) is included in the ring, and $x_{ij}=0$, otherwise. Furthermore, let $y_i=1$, if node i is included in the ring, and $y_i=0$, otherwise.

Maximize

$$\mu \sum_{\substack{i,j \in V \\ i<j}} r_{ij} \cdot y_i \cdot y_j + \mu \sum_{i \in V} r_{ii} \cdot y_i - \mu \sum_{(i,j) \in E} c_{ij} \cdot x_{ij} - \delta \sum_{i \in V} p_i \cdot (1 - y_i) \qquad (1)$$

subject to

$$\sum_{(i,k) \in E} x_{ik} + \sum_{(k,j) \in E} x_{kj} = 2 \cdot y_k \qquad \forall k \in V \qquad (2)$$

$$\frac{1}{2}\left(\sum_{\substack{(i,j)\in E\\ i\in S, j\notin S}} x_{ij} + \sum_{\substack{(i,j)\in E\\ i\notin S, j\in S}} x_{ij}\right) + (1-y_k) + (1-y_l) \geq 1$$

$$\forall S \subset V : 3 \leq |S| \leq n-3,\ k \in S,\ l \notin S \quad (3)$$

$$y_i = 1 \qquad \forall i \in Q \quad (4)$$

$$\sum_{\substack{i,j\in V\\ i\leq j}} r_{ij} \cdot y_i \cdot y_j \geq a \quad (5)$$

$$\sum_{(i,j)\in E} c_{ij} \cdot x_{ij} \leq b \quad (6)$$

$$\sum_{i\in V} y_i \leq h \quad (7)$$

$$y_i \in \{0,1\} \qquad \forall i \in V \quad (8)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \quad (9)$$

Constraints (2) ensure that each node in the solution is connected to exactly two other nodes. Constraints (3) are connectivity constraints, guaranteeing that all nodes of the solution are connected to each other (such that sub-cycles are prevented). Constraints (4) ensure that each required node is in the solution. Constraint (5) guarantees that the lower bound for the revenues is met. Inequality (6) prevents the given budget *b* from being exceeded. Constraint (7) guarantees that there are no more than *h* nodes in the solution. As this model is not linear and contains an exponential number of constraints, modifications are needed to apply standard methods of mathematical programming. A linearization can be obtained by the use of binary variables z_{ij} that indicate whether nodes *i* and *j* are both in the solution. The constraints (3) can be replaced by a polynomial number of constraints exploiting the idea of Miller, Tucker, and Zemlin for the prevention of sub-cycles (Miller et al. 1960). Corresponding transformations are described by Fink et al. (1998) and Gouveia and Pires (1998). While such models may be useful for computing optimal solutions to small problem instances or to compute upper bounds, one generally needs heuristic approaches to efficiently generate good solutions for larger problem instances. Furthermore, heuristic approaches allow an easy extension with respect to model variations such as non-linearities.

The GRNDP subsumes several types of problems. It reduces to the classical *Traveling Salesman Problem* (*TSP*) when, e.g., there are no revenues, *Q*=*V*, and there are no limiting restrictions (i.e., *a*, *b*, and *h* are not

specified). While it is reasonable to apply special methods to solve TSP instances, it may be appropriate to solve other types of problems that generally do not result in solutions representing Hamiltonian cycles by general heuristic approaches for the GRNDP. In the following we briefly refer to some of these problems, which are related to the GRNDP in Table 1.

The *Ring Network Design Problem* (*RNDP*) has been discussed by Gendreau et al. (1995) who concentrate on the development of efficient heuristics such as greedy construction, as well as greedy add and drop exchange based local search. The *Steiner Ring Network Design Problem* (*SRNDP*), which has been introduced by Laporte and Nobert (1983) and varies the problem to find a traveling salesman tour of minimal length including a given subset of the node set (Cornuejols et al. 1985, Ratliff and Rosenthal 1983), is distinguished by a subset of required nodes that must be included. Gouveia and Pires (1998) develop mathematical models for the *Steiner Ring Network Design Problem with Revenues* (*SRNDPR*) that generalizes the SRNDP with respect to revenues and additional restrictions. The *Selective Traveling Salesman Problem* (*STSP*) or *Orienteering Problem* (*OP*) (Laporte and Martello 1990, Fischetti et al. 1998[1]) is to maximize the revenues associated with the nodes included in the ring while there is an upper bound for $c(R)$. On the other hand, the *Prize Collecting TSP* (*PCTSP*) (Balas 1989) is to minimize link costs and penalties due to nodes not included in the ring while there is a lower bound for node revenues associated with the nodes included in the ring.

	Q	a	b	h	α	β	γ	δ	r
TSP	V	0	∞	$\|V\|$	0	0	1	0	$r_{ij}= 0 \quad \forall i,j$
RNDP	$\varnothing$	0	b	$\|V\|$	1	0	1	0	$r_{ii}= 0 \quad \forall i$
SRNDP	Q	0	∞	$\|V\|$	0	0	1	0	$r_{ij}= 0 \quad \forall i,j$
SRNDPR	Q	0	∞	h	1	0	1	0	$r_{ii}= 0 \quad \forall i$
STSP/OP	{1}	0	b	$\|V\|$	0	1	0	0	$r_{ij}= 0 \quad \forall i\neq j$
PCTSP	$\varnothing$	a	∞	$\|V\|$	0	0	1	1	$r_{ij}= 0 \quad \forall i\neq j$

Table 1 Survey of several problem types with respect to the GRNDP frame.

Although, as we have shown above, the GRNDP covers a great variety of combinatorial optimization problems with a ring-like structure there are a great number of other network design problems that have not yet been included within this framework. A generic network design model that emphasizes a great variety of vehicle routing and transportation problems as

[1] Differing from Fischetti et al. (1998), some authors (e.g., Chao et al. (1996)) treat the OP as to construct an (open) tour from a source node to a target node (instead of a ring).

well as specifically tailored problem specific solution methods is given by Magnanti and Wong (1984).

With respect to ring network design problems the GRNDP is perhaps the most general for existing problems from the literature. However, in various telecommunication applications an underlying network structure consists of a backbone as well as of tributary networks. While the backbone may constitute a set of major hub nodes used as, e.g., switching points for communication traffic (also called concentrators or access points), traffic may arise at any node of a network so that any of these nodes may be linked to a hub node through a 'secondary' tributary network. For an excellent survey on hub location problems see Klincewicz (1998). In the GRNDP we may model the ring that is to be constructed as a backbone while a tributary network is not explicitly considered (as, e.g., in the TPP below).

Corresponding modifications of the GRNDP arise from the problem of connecting LAN clients using the ring topology: Apart from the given n nodes of the basic problem, there are some secondary nodes that have to be connected to the ring. Consequently, additional costs have to be taken into account. This problem is related to the *Traveling Purchaser Problem* (*TPP*), which is to find a tour through a subset of m markets and the depot and to purchase each of n items such that the sum of travel costs and purchase costs is minimized. Here, the items correspond to the secondary nodes, and the markets to the nodes of the basic RNDP, respectively. The TPP is another generalization of the TSP with linkages not only to telecommunications, but also to production planning and marketing (Voß 1996). That paper discusses certain tabu search heuristics and a simulated annealing approach for the TPP based on add and drop exchanges.

In addition, it is possible to consider capacities associated with the nodes of the ring (e.g., limitations of the number of secondary nodes connected to a ring node or limitations regarding the amount of traffic through the nodes). This problem relates to the *Capacitated Minimum Spanning Tree Problem* (*CMST*; cf., e.g., Amberg et al. 1996). Another modification of the GRNDP (taking costs and protection as objectives and considering capacities) is investigated by Mocci and Primicerio (1997). Furthermore, it is possible to change the concept of having required nodes. As an example, one may assume that a number of subsets of the node set are given such that at least or exactly one node of each subset has to be included in the backbone (see, e.g., Laporte et al. 1996). Related is the determination of relevant clusters that then form the basis for building tributary networks (Laguna 1994). For a general discussion of revenues in telecommunications see, e.g., LeBlanc and Narasimhan (1994).

3. HEURISTIC APPROACHES

In Section 1, we proposed solving ring network design problems by means of HOTFRAME (Fink and Voß 1999), a meta-heuristics framework that provides a collection of reusable components for heuristic search. In the sequel we describe those methods that we will apply to ring network design problems. One should keep in mind that we intend to make it easy to apply general yet effective heuristics. However, the need for careful and time consuming tuning of most meta-heuristics constitutes a major drawback as general problem solvers. The problem of fixing the parameters involved may be called calibration. Our design philosophy is that there should be no need to perform a parameter tuning to get reasonable results. Thus, we primarily focus on auto-adaptive meta-heuristics that do not require calibration for the specific problem at hand. In Section 4, we will describe computational results – especially with respect to evaluating the competitiveness of our approach.

3.1 Construction of starting solutions

The application of local search heuristics operating on solution spaces restricted to feasible solutions may require the construction of initial feasible solutions. (Below, we also describe the incorporation of infeasible solutions into the solution space, which allows the direct application of improvement methods.) Here we consider three construction methods, which shall later be applied for RNDP instances. These methods are appropriate if one can find feasible solutions by successively adding nodes until some termination condition is met.

To find out whether a reasonably good starting solution is needed at all for the successful application of the improvement methods, first, we may construct a somewhat arbitrary ring by successively adding nodes according to the numbering of the nodes until this would break the budget restriction or exceed the maximum number of nodes limit. Second, we use a cheapest insertion heuristic (Chins) that works as follows: Start with a ring R with one randomly chosen node. Now successively enlarge this ring by choosing in each iteration $k=2,...,h$ the best combination (concerning the objective function value) under consideration of the remaining $n-k+1$ nodes and all $k-1$ insertion positions taking into account the budget limit.

Third, building on this cheapest insertion heuristic, we also use a corresponding pilot method (Duin and Voß 1999). The pilot method is a meta-heuristic that builds primarily on the idea of looking ahead for each possible local choice (by computing a so-called ‘pilot’), memorizing the best result, and performing the corresponding move. Here, we apply this strategy

by successively performing a cheapest insertion heuristic for all possible local steps (i.e., starting with all rings resulting from adding some not yet included node at some position to the current ring). Of course, this leads to an exponential time complexity. However, as the experimental results presented in Section 4 will show, it is reasonable to restrict the pilot process to a given *evaluation depth*. That is, the pilot method is performed until an incomplete solution with a given number of nodes is reached; this solution is completed by continuing with a conventional cheapest insertion heuristic.

3.2 Neighborhood structure

The neighborhood description assumes the ring R as a chain of nodes, each connected by an edge. With this, moves are composed of attributes defined by giving the edges that are deleted and inserted; this information is exploited by the tabu search methods described below.

We use a neighborhood that is defined as follows: The neighbors of a solution (ring) R are all solutions that may be reached from R by performing one of these moves:

– A node $v \in R$ may be shifted to another position in the ring.
– A node $v \in R$ may be excluded from the ring.
– A node $v \in V \setminus R$ may be inserted to the ring at some position.

The evaluation of potential moves can be defined as the increase of the objective function value. As the size of the neighborhood is of $O(n^2)$, it is crucial to perform the move evaluation in an incremental way (instead of recomputing the objective function of a modified ring) to reach a sufficiently efficient implementation. One may define the solution space to include only feasible solutions or it may be expanded to include infeasible solutions, which are evaluated as $-\infty$. To guide local search methods into promising regions of the solution space, it is essential to sensibly evaluate moves. The simple evaluation of a move by the implied change of the objective function value often does not provide enough information in this respect. For example, moves leading to infeasible neighbors might not be evaluated as $-\infty$ but by some kind of penalization that nevertheless takes into account possible changes of costs and revenues. If we are ‘deep’ in infeasible regions, and several moves are needed to reach feasibility, moves must be evaluated with respect to the tendency towards feasibility (which may be non-trivial as there can be conflicting constraints). Moreover, considering the STSP, where only revenues are part of the objective function while there is an upper bound for the sum of costs, potential changes of the revenues must be evaluated in relation to the corresponding cost changes.

Restricting to the RNDP, SRNDPR, and the STSP/OP, for which we describe computational results in Section 4, we compute the evaluation

$f(\mu(R_1,R_2))$ of a move μ from a solution (ring) R_1 to a solution R_2 as follows (larger move evaluations mean 'better' moves). Disregarding infeasibilities and assuming that increasing revenues are accompanied by decreasing costs and vice versa, move evaluations are computed as

$$f(\mu(R_1,R_2)) = \begin{cases} f(R_2) - f(R_1) & \text{if } \mu \neq 0 \\ \dfrac{f(R_2) - f(R_1)}{c(R_2) - c(R_1) + 1} & \text{if } \mu = 0 \text{ and } f(R_2) - f(R_1) \geq 0 \\ \dfrac{f(R_2) - f(R_1)}{c(R_2) - c(R_1) - 1} & \text{if } \mu = 0 \text{ and } f(R_2) - f(R_1) < 0 \end{cases}$$

Also considering moves involving infeasible solutions, move evaluations are adapted by adding or subtracting a large value that is considerably larger than any objective function difference between two feasible solutions when moving from an infeasible solution to a feasible one or moving from a feasible solution to an infeasible one, respectively. In case the neighborhood of the current solution only contains infeasible solutions, moves are evaluated by the reduction of violations of constraints.

3.3 Improvement methods

Improvement procedures for the ring network design problem may be based on greedy local search, i.e., successively performing a move with a maximum evaluation at each step, considering the neighborhood described in the previous section. As such methods may lead to local optima of non-satisfying solution quality, we consider the application of meta-heuristic search methods (simulated annealing and various tabu search approaches) with the aim to guide the search to overcome local optimality.

3.3.1 Simulated annealing

Simulated annealing extends basic local search by allowing moves to inferior solutions (Kirkpatrick et al. 1983, Dowsland 1993). The basic algorithm of simulated annealing may be described as follows: Successively, a candidate move is randomly selected; this move is accepted if it leads to a solution with a better objective function value than the current solution, otherwise the move is accepted with a probability that depends on the deterioration Δ of the objective function value. The probability of acceptance is computed according to the Boltzmann function as $e^{-\Delta/T}$, using a temperature T as control parameter.

We follow the parameterization of this general simulated annealing procedure as described by Johnson et al. (1989). The value of T is initially high, which allows many inferior moves to be accepted, and is gradually reduced through multiplication by a parameter *coolingFactor* according to a geometric cooling schedule. At each temperature *sizeFactor* $\times$ $|N|$ move candidates are tested ($|N|$ denotes the current neighborhood size). The starting temperature is determined as follows: Given a parameter *initialAcceptanceFraction* and based on an abbreviated trial run, the starting temperature is set so that the fraction of accepted moves is approximately *initialAcceptanceFraction*. A further parameter, *frozenAcceptanceFraction*, is used to decide whether the annealing process is *frozen* and should be terminated. Every time a temperature is completed with less than *frozenAcceptanceFraction* of the candidate moves accepted, a counter is increased by one. This counter is reset every time a new best solution is found. The procedure is terminated when the counter reaches 5. We set these parameters to the values recommended by Johnson et al. (1989): *coolingFactor* = 0.95, *initialAcceptanceFraction* = 0.4, *frozenAcceptanceFraction* = 0.02, *sizeFactor* = 16. The suitability of these values has been confirmed by preliminary experiments.

3.3.2 Tabu search

The basic paradigm of tabu search is to use information about the search history to guide local search approaches to overcome local optimality (Glover and Laguna 1997). In general, this is done by a dynamic transformation of the local neighborhood. As for simulated annealing, this may lead to performing deteriorating moves when all improving moves of the current neighborhood are set tabu. A general description of a tabu search frame may be presented as shown in Algorithm 1 for a given starting solution *s* and a tabu criterion that is represented by the object *TabuMemory*. A neighbor, respectively a corresponding move, is called *admissible*, if it is not tabu or if an aspiration criterion is fulfilled. The only aspiration criterion used here is to allow all moves that lead to a neighbor with a better objective function value than encountered so far.

Algorithm 1 Generic tabu search heuristic.

```
TabuSearch( s, TabuMemory ):

Initialize TabuMemory
while ( stopping criterion not fulfilled )
    s' = BestAdmissibleNeighbor( s, Neighborhood, TabuMemory )
    TabuMemory.add( move( s, s' ) )
    s = s'
    TabuMemory.add( s )
    if ( escape triggered by TabuMemory )
        perform a diversifying move
```

Sometimes, it has proven to be beneficial to incorporate means to diversify the search into new regions of the search space. This requires an appropriate mechanism to detect situations when the search might be trapped in a certain area of the solution space.

In the following we briefly describe various tabu search methods that differ especially in the way in which the tabu criteria are defined, taking into consideration the information about the search history (performed moves, traversed solutions).

Static tabu search

The most commonly used tabu search method is to use a *recency-based* memory that stores moves, more exactly move attributes, of the recent past. The basic idea of such static tabu search approaches is to prohibit an appropriately defined inversion of performed moves for a given period. Here we use a tabu list with a fixed length l.

Considering a performed move, we store in dependence on the move type move attributes that represent the inserted edges in a static tabu list of fixed length. To obtain the current tabu status of a move to a neighbor, we must check whether the edges to be deleted are contained in the tabu list. As we apply non-homogeneous moves with a varying number of attributes,[2] there are different ways to define the tabu criterion: A move may be classified as tabu when at least one, two, or three of the attributes of this move are contained in the tabu list. For our purposes, all these criteria are implemented using a parameter *tabu threshold*. It defines the number of attributes of a move that have to be contained in the tabu list in order to

[2] When a node is shifted to another position in the ring this results in three deleted edges and three inserted edges. When a node is shifted out from (included to) the ring this leads to two (one) deleted and one (two) inserted edges.

regard the move as tabu (taking into account that a move may also lead to less than *tabu threshold* attributes).

Strict tabu search

Strict tabu search embodies the idea of preventing cycling to formerly traversed solutions. That is, the goal of strict tabu search is to provide necessity and sufficiency with respect to the idea of not revisiting any solution. Accordingly, a move is classified as tabu if and only if it leads to a neighbor that has already been visited during the previous part of the search.

There are two primary mechanisms to accomplish the tabu criterion: First, we may exploit logical interdependencies between the sequence of moves performed throughout the search process, as realized by the reverse elimination method (cf., e.g., Glover and Laguna 1997, Voß 1996). Second, we may store information about all solutions visited so far. This may be carried out either exactly or, for reasons of efficiency, approximately. For our purpose this is accomplished by using a hash function, that defines a non-injective transformation from the set of solutions to integer numbers (Woodruff and Zemel 1993). Given a vector $(w_1,...,w_n)$ of pseudo-random integers, we compute hash codes of a solution that is represented by a ring R as

$$|R| + \sum_{\text{edge}\,(i,j)\in R} (i+j)\cdot w_{\min\{i,j\}} \quad .$$

As the hash code of two different solutions may be the same whenever a so-called collision occurs, moves might be unnecessarily set tabu in some cases. However, as our experiments have shown, this random effect seldom affects the search negatively. Accordingly, we restrict the trajectory based memory to the use of hash codes. Each solution (i.e., each hash code) of the trajectory memory is attributed by the iteration when a corresponding solution was visited the last time and by the frequency indicating how often this solution has been visited.

Reactive tabu search

Reactive tabu search aims at the automatic adaptation of the tabu list length of static tabu search (Battiti 1996). The basic idea is to increase the tabu list length when the tabu memory indicates that the search is revisiting formerly traversed solutions. The concrete algorithm applied here may be described as follows: We start with a tabu list length l of 2 and increase it to $\min\{\max\{l+2, l\times 1.2\},u\}$ every time a solution has been repeated, taking into account an appropriate upper bound u (to guarantee at least one admissible move). If there has been no repetition for some iterations, we decrease it appropriately to $\max\{\min\{l-2, l\times 0.8\},2\}$. To accomplish the detection of a

repetition of a solution, we apply a trajectory based memory using hash codes as described for strict tabu search.

As noted above, and especially for reactive tabu search, noticed by Battiti (1996), it may be appropriate to include means for diversifying moves whenever the tabu memory indicates that we may be trapped in a certain region of the search space. As a corresponding trigger mechanism, we use the combination of at least *two solutions each having been traversed three times*. The simple escape strategy used here is to perform randomly a number of moves (in dependence on the moving average of the number of iterations between solution repetitions).

4. COMPUTATIONAL RESULTS

We focus on the application of meta-heuristics by means of HOTFRAME (Heuristic OpTimization FRAMEwork, Fink and Voß 1999), which provides a collection of adaptable C++ classes for heuristic search and an architecture that defines the collaboration between classes. For the GRNDP we essentially had to implement classes representing the concepts *solution*, *move*, and *move attribute*. Then, by straightforward reuse of the heuristic classes provided by HOTFRAME – without any fine tuning – one is able to do a fair comparison of different heuristics by controlled and unbiased experiments. However, computation times resulting from the application of generic components may offer an opportunity to be reduced (see, e.g., Duin and Voß (1999) for corresponding ideas regarding the pilot method). Computation times are generally given as average CPU-time in seconds on a Pentium II/266.

In Section 4.1 we primarily compare different construction and improvement heuristics applied to the RNDP as there are no optimal solutions available. Then, we test the competitiveness of our approach by a comparison to (mostly provably optimal) results from the literature for the SRNDPR (Section 4.2) and the STSP/OP (Section 4.3); here we restrict to the straightforward application of strict and reactive tabu search, which are both parameterless, using a randomly chosen node as starting solution.

4.1 Ring Network Design Problem

In order to test the algorithms presented in Section 3 we have generated random instances of the RNDP according to the procedure described by Gendreau et al. (1995). For each instance, first coordinates of n points P_i were randomly generated in the $(0,1)\times(0,1)$ square according to a continuous uniform distribution. Then, each of the values c_{ij} was set to the Euclidean

distance between P_i and P_j. The budget value was set as $b=0.75\cdot\sqrt{n/2}$. For all pairs (i,j) with $i<j$ the revenue $r_{ij}=r_{ji}$ was randomly determined according to a continuous uniform distribution in $(0,18.4\cdot b/n^2)$. (See Gendreau et al. (1995) for more information.) We apply the heuristics described above for randomly generated data sets with 40, 80, and 120 nodes (ten problem instances each).[3] Solutions for such problem instances may be visualized as shown in Figure 1 for a problem instance with 120 nodes.

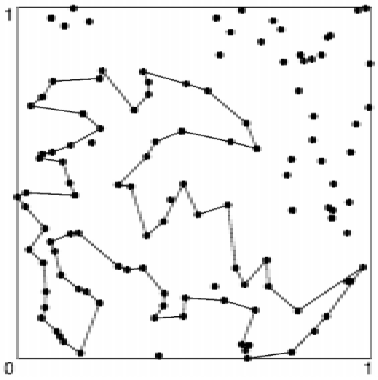

Figure 1 Problem instance (120 nodes) with ring.

In the following, we summarize some results of the application of different heuristics on the problems described above. All results given in the tables below are average values over ten problem instances for each class. For all columns of the following tables, the left value gives the average deviation from the best results obtained overall, and the right value shows the average CPU-time in seconds.

Table 2 shows the average deviation from the best results for different construction heuristics. The pilot heuristic obtained very good results, however, in connection with unreasonable computation times. Therefore, we also examined the effect of limiting the evaluation depth to 10 (Pilot-10). We note that the pilot heuristic with a bounded evaluation depth of 10 still provided very good results (still with large computation times). The development of the pilot process may be visualized as shown in Figure 2. The lines show the best objective function value obtained during the application of the pilot procedure for the problem instances with 80 nodes.

[3] Unfortunately, the computational results presented below are not comparable to the results obtained by Gendreau et al. (1995), as the problem instances used there were randomly generated and not kept on file. An implicit possibility for a comparison results from the correspondence of our cheapest insertion method with the C2 ring construction heuristic of Gendreau et al. (1995).

As can be seen, an evaluation depth of approximately 10 seems reasonable to speed up the computation, where in this case an evaluation depth of 22 would have led to the same results as the unbounded pilot procedure.

n	Chins		Pilot-10		Pilot	
40	45.02%	0.05	2.75%	14.15	2.75%	22.46
80	42.32%	0.29	7.19%	552.80	6.21%	3965.63
120	41.62%	0.97	5.09%	4419.53	3.17%	63094.22

Table 2 Average deviation from best results for construction heuristics.

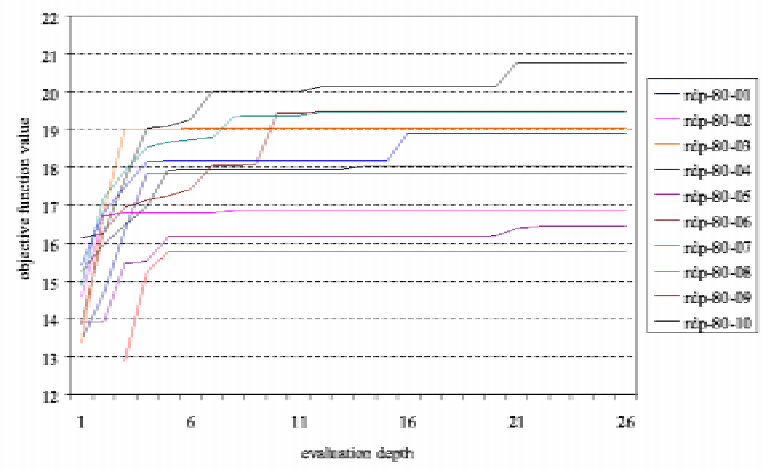

Figure 2 Development of best objective function value during pilot procedure for problem instances with 80 nodes.

n	Greedy Search s = id		Greedy Search s = Chins		Sim. Anneal. s = id		Sim. Anneal. s = Chins	
40	67.19%	0.06	30.37%	0.07	86.42%	1.86	32.02%	2.96
80	73.12%	0.34	32.27%	0.42	99.98%	5.81	34.47%	10.04
120	82.70%	0.85	30.15%	1.52	99.99%	11.39	31.05%	23.11

Table 3 Average deviation from best results for greedy search and simulated annealing.

Table 3 shows the results of greedy local search and simulated annealing with parameters set as described in Section 3.3.1. With s we denote the

applied starting solution: constructing the ring in accordance with the numbering of the nodes (s = id), the solution determined by the cheapest insertion heuristic (s = Chins). From these results, and further experiments not shown here, we conclude that reasonable starting solutions seem to be necessary to obtain high quality solutions. Due to the unsatisfactory solution quality obtained by greedy local search and simulated annealing, in the following we focus the presentation to the application of tabu search methods starting from the solutions obtained by the cheapest insertion method.

	TS-static (3,000)		TS-reactive (3,000)		TS-reactive (3,000)	
n	s = Chins		s = Chins		s = Pilot-10	
40	15.58%	11.29	6.31%	17.67	0.38%	31.45
80	20.06%	49.78	13.93%	50.95	4.54%	619.22
120	19.39%	129.21	18.88%	125.37	2.19%	4595.40

Table 4 Average deviation from best results for different tabu search methods.

Table 4 shows the results of static and reactive tabu search, applied for 3,000 iterations each. For static tabu search, the tabu list length l has been set to 0.3 times the number of the nodes (rounded down), the *tabu threshold* has been set to 2. The *tabu threshold* for reactive tabu search has been set to 1. The solution quality obtained is clearly better than those of greedy search or simulated annealing. However, when starting with the solutions obtained by the cheapest insertion heuristic, the results are not satisfying compared to the application of the pilot heuristic. The modest improvements obtained by reactive tabu search starting from the application of Pilot-10 may imply that there is a lack of search diversification. The dominating effect of the revenues may hinder a real restructuring of the ring during the search. This supposition has been confirmed by comparing solutions by means of the visualization as shown in Figure 1. Therefore, we examined the effect of the following diversification procedures: First, we increased the number of iterations to 100,000 in combination with a simple diversification procedure by performing seven random escape moves each 250 iterations. Second, we examined the effect of performing ten repetitions with 10,000 iterations each, starting from the rings constructed by the cheapest insertion heuristic with a randomly chosen first node. (As reactive tabu search already contains an escape strategy, we did not include the enforced escape moves for the latter test.) Both ideas led to a significant increase of solution quality (cf. Table 5). Assessing the results obtained while taking into account that reactive tabu search generally works without parameter tuning, we may

conclude that reactive tabu search has been the most appropriate improvement heuristic considered here.

	TS-static-esc, (100,000)		TS-reactive-esc, (100,000)		TS-static-esc, (10,000)		TS-reactive-esc, (10,000)	
n	s = Chins		s = Chins		s = Chins, 10 rep.		s = Chins, 10 rep.	
40	0.10%	373.26	0.00%	491.71	0.47%	374.83	0.02%	642.07
80	7.71%	1667.04	2.15%	1689.63	4.70%	1690.34	1.90%	1745.32
120	13.54%	4259.03	8.57%	4125.28	9.16%	4310.12	6.16%	4163.20

Table 5 Average deviation from best results for static and reactive tabu search (partly with additional escape moves).

4.2 Steiner Ring Network Design Problem with Revenues

We used the original problem instances described by Gouveia and Pires (1998). The data represent randomly generated instances with Euclidean distances and uniformly distributed revenues. Instances up to 50 nodes (sri20, sri30, sri40, sri50) have been solved to optimality by Gouveia and Pires (1998) by a mathematical programming approach. Furthermore, we used corresponding instances with more than 50 nodes (sri60, sri70, sri80), for which no optimal solutions are known.

Table 6 shows the results of the application of strict tabu search and reactive tabu search to instances with up to 50 nodes. All instances have been solved to optimality by both tabu search methods in a few seconds, whereas by Gouveia and Pires (1998) significantly more computation time is needed (up to several hours to prove optimality).

	Problem Instance					Strict Tabu Search			Reactive Tabu Search		
	n	|*E*|	|*Q*|	*h*	opt.	obj.	%dev.	time	obj.	%dev.	time
sri20	20	50	5	10	27471	27471	0.00%	0.4	27471	0.00%	0.6
sri30	30	75	4	8	14842	14842	0.00%	0.3	14842	0.00%	0.8
sri40	40	100	5	10	22156	22156	0.00%	0.2	22156	0.00%	0.2
sri50	50	125	6	12	35006	35006	0.00%	21.5	35006	0.00%	10.5

Table 6 Results for SRNDPR instances with $n \in \{20,30,40,50\}$.

Table 7 shows the results of the application of strict tabu search and reactive tabu search (for 100 and 1000 seconds, respectively) to instances with $n \in \{60, 70, 80\}$. As for these instances no optimal solutions are known the '%dev.' columns show the percentage deviation from the best results

obtained. One may conclude that there are no significant differences between strict and reactive tabu search concerning these instances. With respect to the effect on the solution quality of increased computation time, consider Figure 3.

Problem Instance						Strict Tabu Search				Reactive Tabu Search			
						100 s		1000s		100 s		1000 s	
	n	|*E*|	|*Q*|	*h*	best	obj.	%dev.	obj.	%dev.	obj.	%dev.	obj.	%dev.
sri60	60	150	8	15	54837	54816	0.04%	54837	0.00%	54580	0.47%	54823	0.03%
sri70	70	175	9	18	78679	78387	0.37%	78659	0.03%	78679	0.00%	78679	0.00%
sri80	80	200	10	20	99888	99733	0.16%	99888	0.00%	99772	0.12%	99772	0.12%
avg.							0.19%		0.01%		0.19%		0.05%

Table 7 Results for SRNDPR instances with $n \in \{60, 70, 80\}$.

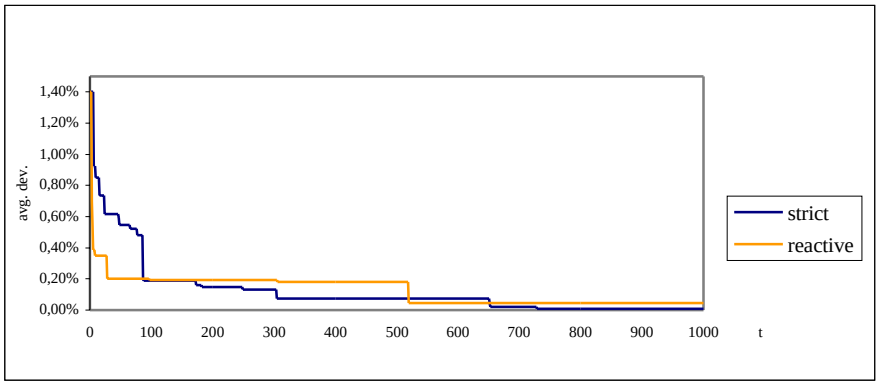

Figure 3 Deviation from best solutions found in dependence on increasing computation time (up to t = 1000s) for strict and reactive tabu search (average over sri60, sri70, and sri80).

4.3 Selective Traveling Salesman / Orienteering Problem

We compare to results reported by Fischetti et al. (1998), who have done an extensive examination of a branch-and-cut approach, which included several problem-specific enhancements. We consider two sets of problem instances (available from the TSPLIB (Reinelt 1991)). First, we use twelve problems, which have been solved to optimality by Fischetti et al. (1998). Originally, eil30, eil33, eil51, and eil76 are vehicle routing problem

instances. With respect to the STSP/OP, the customer demands are treated as node revenues. Moreover, the budget limit b is computed as $z \cdot H(I)$, with $z \in \{0.25, 0.5, 0.75\}$, and $H(I)$ representing the costs of the shortest Hamiltonian tour of problem instance I, which results in twelve problem instances. Second, we use two original TSP instances (ts225, pr226) from the TSPLIB, for which the revenues are defined, in accordance to Fischetti et al. (1998), for each node i by $r_{ii}=1$ (generation 1) or $r_{ii}=1+(7141 \cdot i+73) \bmod 100$ (generation 2). The budget limit b is computed as described above with $z=0.5$. The corresponding four instances are among the hardest problems considered by Fischetti et al. (1998).

Like for the SRNDPR we restrict to the straightforward application of strict and reactive tabu search. For the instances eil30, eil33, and eil51 the optimal solutions were obtained in a few seconds; eil76 led to deviations from the optimum smaller than 1%.[4] Table 9 shows the results of the application of strict tabu search and reactive tabu search (1000s) to the hard STSP/OP instances. The 'best/opt.' column gives the best results reported by Fischetti et al. (1998). For three of these instances optimal solutions are not known, so the best known objective function value is shown in parentheses. The '%dev.' columns give the percentage deviation from the best results obtained by Fischetti et al. (1998), which are provable near the optimum (according to the bounds reported by Fischetti et al. (1998), or represent the optimum in case of ts225-gen2). Here, strict tabu search is not competitive while reactive tabu search performs significantly better, however, with deviations up to 6%.[5] The superiority of reactive tabu search may be due to the incorporated escape mechanism that leads to a better search diversification.

[4] We should note that eil33-0.25 has a two-node-ring $R = (1,5,1)$ with an objective function value of 1200. Since Fischetti et al. (1998) restrict to rings with not less than three nodes, we also report the then best objective function value 800.

[5] The result obtained for pr226-gen2 by reactive tabu search is better than the best results reported by Fischetti et al. (1998). However, this may be inconsistent with respect to the bounds given there; so far, we were not able to definitely clarify this point.

Problem Instance				Strict Tabu Search			Reactive Tabu Search		
	n	*b*	opt.	obj.	%dev.	time	obj.	%dev.	time
eil30-0.25	30	96	2650	2650	0.00%	0.1	2650	0.00%	0.1
eil33-0.25	33	111	800	800	0.00%	0.1	800	0.00%	0.1
eil51-0.25	51	107	264	264	0.00%	2.8	264	0.00%	0.2
eil76-0.25	76	135	490	486	0.82%	1000.0	490	0.00%	239.7
eil30-0.50	30	191	7600	7600	0.00%	0.1	7600	0.00%	0.1
eil33-0.50	33	221	16220	16220	0.00%	252.8	16220	0.00%	54.7
eil51-0.50	51	213	508	508	0.00%	1.1	508	0.00%	1.4
eil76-0.50	76	269	907	900	0.77%	1000.0	904	0.33%	1000.0
eil30-0.75	30	286	11550	11550	0.00%	0.1	11550	0.00%	0.1
eil33-0.75	33	331	26380	26380	0.00%	133.4	26380	0.00%	9.4
eil51-0.75	51	320	690	690	0.00%	104.7	690	0.00%	3.7
eil76-0.75	76	404	1186	1184	0.17%	1000.0	1181	0.42%	1000.0
avg.					0.15%			0.06%	

Table 8 Results for STSP/OP instances for which optimal solutions are known.

Problem Instance				Strict Tabu Search			Reactive Tabu Search		
	n	*b*	best/opt.	obj.	%dev.	time	obj.	%dev.	time
ts225-gen1	225	63322	(125)	119	4.80%	1000.0	124	0.80%	1000.0
pr226-gen1	226	40185	(134)	112	16.42%	1000.0	126	5.97%	1000.0
ts225-gen2	225	63322	6834	6499	4.90%	1000.0	6686	2.17%	1000.0
pr226-gen2	226	40185	(6615)	5556	16.01%	1000.0	6688	-1.10%	1000.0

Table 9 Results for hard STSP/OP instances.

5. CONCLUSIONS

In this paper we have presented the application of meta-heuristics to ring network design problems. The best overall results, concerning solution quality, computation time, and ease of use have been achieved by the application of reactive tabu search. Comparisons with results from the literature have shown the appropriateness of applying general heuristic components, which are not adapted to the GRNDP but require only the implementation of the solution space and neighborhood structure. We were mostly able to obtain competitive results, yet our methods have been neither calibrated nor have we done any specializations with respect to the specific

problem type (RNDP, SRNDPR, STSP/OP). Of course, by using specialized methods one may generally obtain better results – at the cost of more effort in designing and implementing these methods.

From a theoretical point of view, there is no hope for a 'perfect' general purpose solver that is better than any other method, because one can show that methods must be adapted to the problem at hand to yield superior results (Culberson 1998, Wolpert and Macready 1997). However, HOTFRAME enables one to perform such adjustments by using adaptation points as described above. In this respect, future research should more thoroughly examine the general design of solution spaces and neighborhoods, and its effects to the effectiveness of local search and corresponding meta-heuristics.

The real-world application of the approach discussed in this paper may require an incremental adoption path (Fink et al. 1999). However, the use of object-oriented software technology supports a corresponding process. In this way, our work exemplifies that integrated exploitation of knowledge from the field of computer science and operations research can support decision making in practice.

Acknowledgements

We thank Luis Gouveia for having kindly provided us with the problem instances used by Gouveia and Pires (1998).

REFERENCES

Amberg, A., W. Domschke, and S. Voß (1996) "Capacitated minimum spanning trees: Algorithms using intelligent search," Combinatorial Optimization: Theory and Practice, vol. 1, pp. 9–39.

Balas, E. (1989) "The prize collecting traveling salesman problem," Networks, vol. 19, pp. 621–636.

Battiti, R. (1996) "Reactive search: Toward self-tuning heuristics," in V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, "Modern Heuristic Search Methods," Wiley, Chichester, pp. 61–83.

Chao, I.-M., B.L. Golden, and E.A. Wasil (1996) "A fast and effective heuristic for the orienteering problem," European Journal of Operational Research, vol. 88, pp. 475–489.

Cornuejols, G., J. Fonlupt, and D. Naddef (1985) "The traveling salesman problem on a graph and some related integer polyhedra," Mathematical Programming, vol. 33, pp. 1–27.

Culberson, J.C. (1998) “On the futility of blind search: An algorithmic view of ‘no free lunch’,” Evolutionary Computation, vol. 6, pp. 109–127.

Dowsland, K.A. (1993) “Simulated annealing,” in C. Reeves, editor, “Modern Heuristic Techniques for Combinatorial Problems,” Halsted, Blackwell, pp. 20–69.

Duin, C. and S. Voß (1999) “The pilot method,” Networks, to appear.

Fink, A., G. Schneidereit, and S. Voß (1998) “Ring network design for metropolitan area networks,” Technical Report, Technical University of Braunschweig, 1998.

Fink, A. and S. Voß (1999) “Generic metaheuristics application to industrial engineering problems,” Computers & Industrial Engineering, to appear.

Fink, A., S. Voß, and D.L. Woodruff (1999) “An adoption path for intelligent heuristic search componentware,” in E. Rolland and N.S. Umanath, editors, Proceedings of the 4th INFORMS Conference on Information Systems and Technology, INFORMS, Linthicum, pp. 153–168.

Fischetti, M., J.J. Salazar González, and P. Toth (1998) “Solving the orienteering problem through branch-and-cut,” INFORMS Journal on Computing, vol. 10, pp. 133–148.

Gendreau, M., M. Labbé, and G. Laporte (1995) “Efficient heuristics for the design of ring networks,” Telecommunication Systems, vol. 4, pp. 177–188.

Glover, F. and M. Laguna (1997) “Tabu Search,” Kluwer, Boston.

Gouveia, L. and J.M. Pires (1998) “Models for a Steiner ring network design problem with revenues,” Technical Report, Faculdade de Ciencias da Universidade de Lisboa.

Johnson, D.S., C.R. Aragon, L.A. McGeoch, and C. Schevon (1989) “Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning,” Operations Research, vol. 37, pp. 865–892.

Kirkpatrick, S., C.D. Gelatt Jr., and M.P. Vecchi (1983) “Optimization by simulated annealing,” Science, vol. 220, pp. 671–680.

Klincewicz, J.G. (1998) “Hub location in backbone/tributary network design: a review,” Location Science, vol. 6, pp. 307–335.

Laguna, M. (1994) “Clustering for the design of SONET rings in interoffice telecommunications,” Management Science, vol. 40, pp. 1533–1541.

Laporte, G., A. Asef-Vaziri, and C. Sriskandarajah (1996) “Some applications of the generalized traveling salesman problem,” Journal of the Operational Research Society, vol. 47, pp. 1461–1467.

Laporte, G. and S. Martello (1990) “The selective traveling salesman problem,” Discrete Applied Mathematics, vol. 26, pp. 193–207.

Laporte, G. and Y. Nobert (1983) “Finding the shortest cycle through k specified nodes,” Congressus Numerantium, vol. 48, pp. 155–167.

LeBlanc, L.J. and S. Narasimhan (1994) “Topological design of metropolitan area networks,” Computer Networks and ISDN Systems, vol. 26, pp. 1235–1248.

Magnanti, T.L. and R.T. Wong (1984) “Network design and transportation planning: Models and algorithms,” Transportation Science, vol. 18, pp. 1–55.

Miller, C.E., A.W. Tucker, and R.A. Zemlin (1960) “Integer programming formulation of traveling salesman problems,” Journal of the Association of Computing Machinery, vol. 7, pp. 326–329.

Mocci, U. and L. Primicerio (1997) “Ring network design: An MCDM approach,” in G. Fandel and T. Gal, editors, “Multiple Criteria Decision Making,” Springer, Berlin Heidelberg, pp 491–500.

Morreale, P.A. and G.M. Campbell (1990) “Metropolitan-area networks,” IEEE Spectrum, vol. 27, no. 5, pp. 40–42.

Ratliff, D.M. and A.S. Rosenthal (1983) “Order picking in a rectangular warehouse: a solvable case of the travelling salesman problem,” Operations Research, vol. 31, pp. 507–521.

Reinelt, G. (1991) “TSPLIB – A traveling salesman problem library,” ORSA Journal on Computing, vol. 3, pp. 376–384.

Voß, S. (1996) “Dynamic tabu search strategies for the traveling purchaser problem,” Annals of Operations Research, vol. 63, pp. 253–275.

Wolpert, D.H. and W.G. Macready (1997) “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Computation, vol. 1, pp. 67–82.

Woodruff, D.L. and E. Zemel (1993) “Hashing vectors for tabu search,” Annals of Operations Research, vol. 41, pp. 123–138.