Developing Efficient Metaheuristics for Communication Network Problems by using Problem-specific Knowledge

Franz Rothlauf and Armin Heinzl

Working Paper 9/2004
August 2004

Working Papers in Information Systems

University of Mannheim
Department of Business Administration and Information Systems
D-68131 Mannheim/Germany
Phone +49 621 1811691, Fax +49 621 1811692
E-Mail: wifo1@uni-mannheim.de
Internet: `http://www.bwl.uni-mannheim.de/wifo1`

Developing Efficient Metaheuristics for Communication Network Problems by using Problem-specific Knowledge

Franz Rothlauf
Dept. of Business Administration and Information Systems
University of Mannheim
D-68131 Mannheim/Germany
rothlauf@uni-mannheim.de

Armin Heinzl
Dept. of Business Administration and Information Systems
University of Mannheim
D-68131 Mannheim/Germany
heinzl@uni-mannheim.de

August 24, 2004

Abstract

Metaheuristics, such as evolutionary algorithms or simulated annealing, are widely applicable heuristic optimization strategies that have shown encouraging results for a large number of difficult optimization problems. To show high performance, metaheuristics need to be adapted to the properties of the problem at hand. This paper illustrates how efficient metaheuristics can be developed for communication network problems by utilizing problem-specific knowledge for the design of a high-quality problem representation. The minimum communication spanning tree (MCST) problem finds a communication spanning tree that connects all nodes and satisfies their communication requirements for a minimum total cost. An investigation into the properties of the problem reveals that optimum solutions are similar to the minimum spanning tree (MST). Consequently, a problem-specific representation, the link biased (LB) encoding, is developed, which represents trees as a list of floats. The LB encoding makes use of the knowledge that optimum solutions are similar to the MST, and encodes trees that are similar to the MST with a higher probability. Experimental results for different types of metaheuristics show that metaheuristics using the LB-encoding efficiently solve existing MCST problem instances from the literature, as well as randomly generated MCST problems of different sizes and types.

1 Introduction

The problem of designing a communication network for a given set of requirements is relevant for communication providers, as well as for companies and

institutions who want to build up their own communication infrastructure by renting, or buying, communication capacities from communication providers. The problem considered in this paper involves selecting a spanning tree for the network on which all communication will be performed. Tree structures are especially important for the design of smaller networks like corporate networks, access networks, or local area networks. The problem of building up cost-minimal communication spanning trees was formalized by the minimum communication spanning tree (MCST) problem. The MCST problem (Hu, 1974) finds a spanning tree that connects all given nodes and satisfies their communication requirements for a minimum total cost. The number and positions of the network nodes are given a priori and the cost of the network is determined by the cost of the links. Like other constrained spanning tree problems, the OCST problem is $\mathcal{NP}$-hard (Garey & Johnson, 1979).

The purpose of this paper is to demonstrate how the MCST problem can efficiently be solved by using metaheuristics when problem-specific knowledge is considered for the design of an appropriate tree representation. Previous work (Rothlauf et al., 2003) has shown that optimal solutions for MCST problems are on average similar to the minimum spanning tree (MST) defined on the distance weights of the links. This problem-specific knowledge about the MCST problem can be utilized by combining metaheuristics like evolutionary algorithms (EA), or simulated annealing, with the link-biased (LB) encoding, which is a representation for trees. The LB encoding does not represent all possible trees with the same probability, but randomly chosen LB-encoded solutions encode trees that are similar to the MST with higher probability than randomly chosen trees. Performance evaluations are presented for a collection of existing problem instances from the literature as well as randomly generated MCST problem instances. The experimental results show that due to the over-representation of MST-like trees, metaheuristics using the link-biased encoding show higher performance than metaheuristics that use representations which encode all possible trees uniformly. The presented work is an example on how the efficiency of metaheuristics can be increased in a systematic way by making use of additional problem-specific knowledge regarding the problem to be solved.

The paper is structured as follows: The following section gives a short description of the MCST problem, reviews existing approximation algorithms, and gives an overview of earlier work which demonstrates that optimal solutions for MCST problems are similar to the MST. Section 3 demonstrates how metaheuristics and especially EAs can be adopted for solving the MCST problem. It describes the functionality and relevant design parameters of EAs and introduces the LB-encoding, which allows to represent trees similar to the MST with a higher probability. Section 4 presents experimental results for existing problem instances from the literature as well as for randomly created test problems. The paper ends with concluding remarks.

2 The Minimum Communication Spanning Tree Problem

The design of optimal communication networks that satisfy a given set of requirements has been studied extensively in the literature. Many different variants, with or without additional constraints, have been examined, and either exact optimization approaches or heuristics have been developed (for an overview compare Kershenbaum (1993), Cahn (1998), or Chang and Gavish (1993)). This section introduces the minimum communication spanning tree (MCST) problem, reviews existing methods for solving this problem, and discusses relevant properties of optimal solutions.

2.1 Problem Description

The minimum communication spanning tree problem (also known as optimal communication spanning tree problem or simple network design problem (Johnson, Lenstra, & Kan, 1978)) was introduced in Hu (1974). The problem is listed as [ND7] in Garey and Johnson (1979) and Crescenzi and Kann (2003). For the MCST problem, the number and positions of network nodes are given a priori and the cost of the network is determined by the cost of the links. A link's flow is the sum of the communication demands between all pairs of nodes communicating either directly, or indirectly, over the link. The goal is to find a tree that connects all given nodes and satisfies their communication requirements for a minimum total cost. The cost for each link is not fixed a priori but depends on its length and its capacity. A link's capacity must satisfy the flow over this link, which depends on the entire tree structure.

The MCST problem can formally be defined as follows. An undirected graph is denoted as $G = (V, E)$. $n = |V|$ denotes the number of nodes and $m = |E|$ denotes the number of edges of the graph. There are communication or transportation demands between the n different nodes. The demands are specified by an $n \times n$ *demand matrix* $R = (r_{ij})$, where r_{ij} is the amount of traffic required between location v_i and v_j. An $n \times n$ *distance matrix* $W = w_{ij}$ determines the distance weights associated with each pair of sites. A tree $T = (V, F)$ where $F \subseteq E$ and $|F| = |V| - 1$ is called a *spanning tree* of G if it connects all the nodes. The weight $w(T)$ of the spanning tree is the weighted sum over all pairs of vertices of the cost of the path between all pairs in T. In general,

$$w(T) = \sum_{i,j \in F} f\left(w_{ij}, b_{ij}\right),$$

where the $n \times n$ matrix $B = b_{ij}$ denotes the traffic flowing directly and indirectly between the nodes i and j. It is calculated according to the demand matrix R and the structure of T. T is the minimum communication spanning tree if $w(T) \leq w(T')$ for all other spanning trees T'.

For the MCST problem as proposed by Hu (1974) the cost of a link is calculated as the product of the distance weight w_{ij} times the overall traffic b_{ij} running over the link. Therefore, $f = w_{ij} b_{ij}$. The MCST problem becomes the *minimum spanning tree* (MST) problem if $f = w_{ij}$. Then, T is the minimum

spanning tree if $w(T) \leq w(T')$ for all other spanning trees T', where $w(T) = \sum_{i,j \in F} w_{ij}$.

Cayley's formula identified the number of spanning trees on n nodes as n^{n-2} (Cayley, 1889). Furthermore, there are n different stars on a graph of n nodes. The similarity between two spanning trees T_i and T_j can be measured using the distance $d_{ij} \in \{0, 1, \ldots, n-2\}$ which is defined as

$$d_{ij} = \frac{1}{2} \sum_{u,v \in V} |l^i_{uv} - l^j_{uv}|. \tag{1}$$

l^i_{uv} is 1 if a link from u to v exists in T_i and 0 if it does not exist in T_i. The number of links that two trees T_i and T_j have in common can be calculated as $n - 1 - d_{ij}$.

2.2 Solving the Minimum Communication Spanning Tree Problem

Like other constrained spanning tree problems, the MCST problem is $\mathcal{NP}$-hard (Garey & Johnson, 1979, p. 207). Further more, it was shown in Reshef (1999) that the problem is $\mathcal{MAX}$ $\mathcal{SNP}$-hard (Papadimitriou & Yannakakis, 1991) which means it cannot be solved using a polynomial-time approximation scheme, unless $\mathcal{P} = \mathcal{NP}$. Therefore, the MCST problem belongs to the class of optimization problems that behave like MAX-3SAT (Garey & Johnson, 1979).

Only for a few easy and restricted problem instances, algorithms have been developed that return optimal solutions. Hu (1974), who introduced the MCST problem, gave exact algorithms for two specific versions of the MCST problem. He showed that for the *complete unweighted graph* version, where $w_{ij} = 1$ for every i and j, the problem can be solved in polynomial time using the Gomory-Hu spanning tree algorithm (Gomory & Hu, 1961; Hu, 1974). Hu called this the *optimum requirement spanning tree problem*. In addition, he showed for the *uniform demand* version of the MCST where the communication demands r_{ij} between any two sites are equal that the optimal solution is a star if the distance weights w_{ij} satisfy a stronger version of the triangle inequality: for every $1 \leq i, j, k \leq n$ such that $w_{ij} \leq w_{ik} \leq w_{jk}$, we have $(w_{jk} - w_{ij})/w_{ik} \leq (n-2)/(2n-2)$. If both the communication demands r_{ij} and the distances weights w_{ij} between any two sites are equal, then the optimal solution is also a star. Later Johnson et al. (1978) showed that only the uniform demand version, where the w_{ij} satisfy this stronger version of the triangle equation, can be solved in polynomial time and that all other uniform demand versions of the MCST problem, where $w_{ij} \in \{1, \infty\}$ are $\mathcal{NP}$-hard. Wu et al. (1998) extended this work and showed that the uniform demand version where the weights w_{ij} satisfy the triangle inequality, is $\mathcal{NP}$-hard. For the uniform demand version, Wong (1980) presented a heuristic that finds a tree T which has a maximum cost of twice that of the optimal solution, $w(T) \leq 2w(T_{opt})$.

The development of exact optimization methods for the general, non-uniform demand version of the MCST problem showed less success. Some early work (Dionne & Florian, 1979; Lin, 1982; Gavish, 1983; Gavish & Altinkemer, 1990)

addressed the *general network design problem* and developed heuristics for finding optimal graphs G (not trees T) for given distance weights w_{ij} and demands r_{ij}. However, as the assumptions that are made for solving the general network design problem are incompatible with solving the MCST problem (compare Palmer (1994, p. 10ff)), these heuristics can not be applied to the MCST problem. Later, Peleg (1997) showed that the MCST problem is reducible to a problem called *minimum average stretch spanning tree* (MAST) problem. Therefore, both problems are equivalent to each other and approximation algorithms for the MAST problem can also be used for the MCST problem. In the MAST problem, which was introduced in Alon et al. (1995), a graph G and a distance matrix W is given and a spanning tree T has to be found that minimizes the average stretch of the edges (e.g. minimize $\frac{1}{n-1}\sum_{i,j\in E} t_{ij}^T/w_{ij}$, where t_{ij}^T is the sum of all the weights along the path between i and j in the spanning tree T). Alon et al. presented a randomized algorithm for the MAST problem that constructs a spanning tree such that the average cost of the tree is less than, or equal to, $\exp(O(\sqrt{\log n \log\log n}))$.

Other approximation algorithms for the MCST problem are based on the volume of communication $w(G) = \sum_{i,j\in E} r_{ij} t_{ij}^G$ in the complete graph G, where t_{ij}^G is the sum of all the weights along the shortest path between i and j in G. $w(G)$ represents a trivial lower bound for $w(T)$ because it considers the full original graph G and not only the links used for the tree T. Bartal (1996) and Wu et al. (1998) presented a randomized algorithm that constructs a spanning tree T with expected communication cost $w(T) = O(\log^2 n)w(G)$. This result has been improved by Bartal (1998) to an $O(\log n \log\log n)$ approximation. Around the same time, non-randomized, deterministic algorithms were developed that find a spanning tree with cost $w(T) = O(\log^2 n)w(G)$ (Peleg & Reshef, 1998; Reshef, 1999). Charikar et al. (1998) improved these results and presented a deterministic approximation algorithm that results in $w(T) = O(\log n \log\log n)w(G)$. When using Euclidean distances for the distance weights w_{ij}, Charikar et al. (1998) and Reshef (1999) presented deterministic approximation algorithms that output a spanning tree with cost $w(T) = O(\log n)w(G)$. Despite the progress in developing approximation algorithms for the MCST problem that are based on the volume of communication $w(G)$, Alon et al. (1995) showed that such approximation techniques cannot approximate the original MCST problem better than $\Omega(\log n)$. They demonstrated that there are graphs (and in particular the two-dimensional MCST problem) where the gap between $w(T)$ and $w(G)$ cannot be lower than $w(T) = \Omega(\log n)w(G)$. For more detailed information about approximation algorithms for the MCST problem, we refer to Reshef (1999).

When summarizing the development of optimization algorithms for the MCST problem, we conclude that no efficient algorithmic methods for solving the MCST problem are available. Some algorithms exist for simplified versions of the MCST problems (complete unweighted graph problem and uniform demand problems), but there are no efficient methods for standard MCST problems. Similarly, deterministic and randomized approximation algorithms for the MCST problem are available which are based on the volume of commu-

nication $w(G)$, but none of them is able to solve realistic MCST problems, or output optimal or near-optimal solutions ($w(T) \geq \Omega(\log n)w(G)$).

To overcome the problems with classical optimization approaches, and to develop more efficient optimization methods for the MCST problem, researchers have used metaheuristics like evolutionary algorithms, simulated annealing, tabu search, and other approaches. Such methods do not construct a tree according to an algorithmic method, but search through the search space consisting of all possible trees. It was recognized early on that the proper choice of a representation is crucial for the performance of metaheuristics. A representation determines how trees are encoded such that search operators can be applied. Commonly, trees are encoded as vectors or lists of strings where different strings encode different trees.

One of the first metaheuristic approaches for the MCST problem was presented by Palmer (1994). When applying evolutionary algorithms (EA) to the MCST problem, he recognized that the design of a proper tree representation is crucial for the performance of metaheuristics. Palmer compared different types of problem representations for trees and developed a new representation, the *link and node biased* (LNB) encoding. EAs using the LNB encoding showed good results in comparison to a greedy star search heuristic (Palmer, 1994, chapter 5).

The *characteristic vector* (CV) encoding is a common approach for encoding graphs (Davis et al., 1993; Tang et al., 1997; Sinclair, 1995; Berry et al., 1997; Berry et al., 1999) and trees (Berry, Murtagh, & Sugden, 1994; Berry, Murtagh, & McMahon, 1995). It represents a tree resp. graph as a list of $n(n-1)/2$ binary values (compare section 3.4). The CV encoding shows good performance when used for encoding trees with a low number of nodes. However, with increasing problem size the performance of the CV encoding decreases and metaheuristics using this encoding show low performance (compare Rothlauf (2002, 6.4)).

Recently, several EAs using *direct representations* for trees have been presented, a direct representation for trees (Li, 2001), the *edge-set encoding* (Raidl & Julstrom, 2003), and the *NetDir* encoding (Rothlauf, 2002). When using direct representations for tree problems, there is no additional mapping from the original search space (consisting of trees) to a different search space, where the search operators are applied to, but problem-specific search operators are applied directly to trees. Therefore, metaheuristics search through the search space by directly modifying the structure of a tree. The performance results presented for direct representations are similar to weighted encodings, however it is difficult to design the search operators in such a way that the search space consisting of all possible trees is searched uniformly (Tzschoppe et al., 2004).

Weighted encodings, like the LNB encoding (Palmer, 1994), the weighted encoding (Raidl & Julstrom, 2000), the NetKey encoding (Rothlauf, Goldberg, & Heinzl, 2002), or variants of the LNB-encoding (Krishnamoorthy & Ernst, 2001) represent a tree using a list of continuous numbers (weights). The weights define an order of the edges of the corresponding tree and the represented tree is constructed from an ordered list of edges. Weighted encodings showed good performance when used for tree optimization problems. For more information about weighted encodings, we refer to section 3.3 and 3.4. Furthermore, Abuali

et al. (1995) introduced the *determinant factorization*. This representation is based on the in-degree matrix of the original graph and each factor represents a spanning tree if the determinant corresponding to that factor is equal to one. The results showed that this encoding shows similar performance to the LNB encoding.

Prüfer numbers have been introduced by (Prüfer, 1918) as a constructive proof of Cayleys theorem (Cayley, 1889) and have been subsequently used for different spanning tree problems like the MCST problem (Palmer, 1994; Palmer & Kershenbaum, 1994; Kim & Gen, 1999; Rothlauf, 2002), or the degree constraint spanning tree problem (Zhou & Gen, 1997; Krishnamoorthy et al., 1999). Prüfer numbers describe a one-to-one mapping between spanning trees on n nodes and strings of $n-2$ node labels. Prüfer numbers, like other one-to-one mappings between spanning trees and strings, lead to a low performance of metaheuristics (Rothlauf & Goldberg, 1999; Gottlieb et al., 2001; Julstrom, 2001) as small changes in the string, which encodes a tree, result on average in large changes in the encoded tree. Picciotto (1999) presented other one-to-one mappings between strings of length $n-2$ and trees like the *Blob Code*, the *Happy Code* and the *Dandelian Code* which have similar properties to Prüfer numbers.

Furthermore, Rothlauf et al. investigated how different tree representations influence the performance of EAs for the MCST problem (Rothlauf & Goldberg, 2000; Rothlauf, Goldberg, & Heinzl, 2002). A summary of this work, which also includes determinants for high-quality representations, can be found in Rothlauf (2002). Other work applying EAs to MCST and related problems (e.g. degree constraint MST problems) have been presented by Premkumar, Chu, and Chou (compare Premkumar et al. (2001) or Chu et al. (2000)).

In summary, due to the lack of efficient algorithmic methods, a large amount of work has applied metaheuristics like EAs to MCST problems. When using metaheuristics for tree problems, the choice of a proper tree representation is an important factors for success. Promising representations are either weighted representations or direct representations which both show good results. Other encodings like Prüfer numbers or the CV encoding result in problems for metaheuristics and are not well suited for solving the MCST problem (Palmer, 1994; Rothlauf, 2002).

2.3 Properties of the Minimum Communication Spanning Tree Problem

To develop efficient and problem-specific optimization methods for the MCST problem, it is necessary to examine the properties of the problem and to identify important and general properties. This subsection summarizes relevant properties of the MCST problem that can be used for the development of problem-specific metaheuristics.

A great benefit of metaheuristics, like EAs or local search approaches, is that the underlying search concepts are simple and general, and that they can be applied with little effort to a wide range of different problems. Many researchers believe that the main reason for the widespread use of metaheuristics

is that they can easily be adapted to new or modified problems. The basic principles are easy to understand, and practitioners without a profound scientific background are also able to apply metaheuristics to real-world problems. However, users often neglect that there is a trade-off between the number of problems that can be addressed by a specific optimization method and the solution quality that can be achieved for these problems (Goldberg, 1989, p.6). In general, the efficiency of a specific optimization method decreases with the number of problems that should be solved. At the extreme, if an optimization method is designed to solve all possible optimization problems that can be defined on a specific search space, this method does not perform on average better than random search (Wolpert & Macready, 1995). However, there is hope as optimization methods can perform better than random search if they focus only on a subset of all possible optimization problems. Therefore, efficient methods can be constructed by focusing only on a few specific problem types that should be solved. In order, to develop such efficient and problem-specific optimization methods, insights into specific properties of the problem must be available and utilized. The following paragraphs describe properties of the MCST problem that can subsequently be used for the design of problem-adapted metaheuristics.

When designing communication networks, the costs of a network strongly depend on the distance weights w_{ij} of the links used in the graph. Network structures that prefer links with low distance weights tend to have on average lower overall costs. When focusing on 2-dimensional grids (resulting in Euclidean distance weights w_{ij}), the weights of links near the gravity center of a graph are lower than the weights of links that are far away from the gravity center. Therefore, it is useful to run more traffic over the nodes near the gravity center of a tree than over nodes at the edge of this tree (Kershenbaum, 1993). Consequently, it is desirable to be able to characterize nodes as either interior (some traffic only transits) or leaf nodes (all traffic terminates). The more important a link is and the more transit traffic that crosses one of the two nodes, the higher is on average the degree of the nodes. Nodes near the gravity center tend to have higher degrees than nodes at the edge of the network. This observation was used by Palmer (1994) for the design of the LNB encoding. This tree encoding considers the relative importance of nodes in a tree and the more important a node is, the more traffic transits over it (Palmer & Kershenbaum, 1994). The LNB encoding is an illustrative example of how properties of good solutions for the MCST problem (more traffic is run over nodes near the gravity center of a tree) can been used for the design of a high-quality representation (LNB-encoding).

Another property of the MCST problem was observed by Rothlauf et al. (2003). This work performed a statistical analysis on the properties of optimal solutions T_{opt} for randomly generated MCST problems using both Euclidean (on a two-dimensional grid), and random distance weights w_{ij}. They compared the average distances $\mu(d_{mst,rand})$ of randomly created trees towards the MST to the average distances $\mu(d_{mst,opt})$ of the optimal solutions towards the MST. The MST, as well as the distances, are calculated as described in section 2.1. The results showed that the average distance between the optimal solution T_{opt} and the MST is significantly smaller than the average distance between a randomly

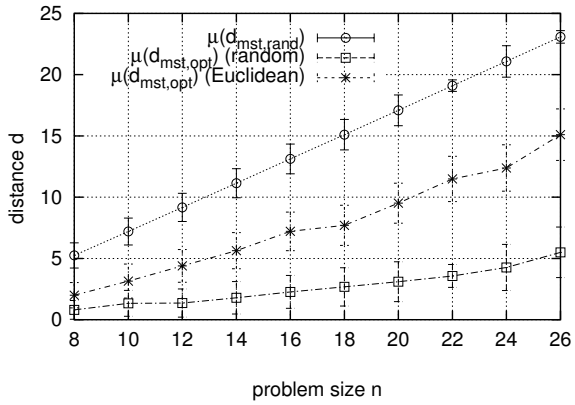


Figure 1: The plots show how $\mu(d_{mst,rand})$ and $\mu(d_{mst,opt})$ using random distances and Euclidean distances depend on the problem size n for randomly created MCST problems. As $\mu(d_{mst,opt}) < \mu(d_{mst,rand})$, optimal solutions are biased towards the MST.

created tree and the MST. Therefore, optimal solutions for the MCST problem are biased towards the MST.

Figure 1 summarizes the results of this work and plots $\mu(d_{mst,rand})$ and $\mu(d_{mst,opt})$ for Euclidean and random distance weights over the problem size (number of nodes) n. The error bars indicate the standard deviations for 100 randomly chosen problem instances of each size n. It can be seen that optimal solutions for MCST problems are strongly biased towards the MST. This means, the probability that trees similar to the MST are the optimal solution of an MCST problem is higher than the probability that randomly chosen trees are the optimal solution. Consequently, this property of the MCST problem should be used subsequently for the development of a problem-specific metaheuristic for the MCST problem (compare section 3.3).

3 Evolutionary Algorithms for the Minimum Communication Spanning Tree Problem

The following sections describe how EAs can be used for solving the MCST problem. Section 3.1 describes the basic principles of EAs and section 3.2 reviews important design parameters of EAs. As the proper choice of a representation is important for EA design, section 3.3 presents the LB encoding for trees. This representation allows EAs to consider that high-quality solutions of the MCST problem are similar to the MST. Finally, section 3.4 reviews some representations for trees which should serve as a benchmark for the LB encoding in section 4.

3.1 Evolutionary Algorithms

Evolutionary Algorithms (Rechenberg, 1973; Holland, 1975; Schwefel, 1975; Goldberg, 1989; Reeves, 1997) are nature-inspired, metaheuristic optimization

methods that imitate basic principles of natural evolution and genetics and apply genetic search operators like recombination, mutation, and selection to a sequence of alleles. The sequence of alleles is the equivalent of a chromosome in nature. A representation determines in which way possible solutions of the optimization problem are encoded as a sequence of alleles.

The application of EAs pose no significant prior restrictions on the optimization problem. To apply EAs, it is necessary to represent the problem solutions such that genetic operators can be applied, and to define a performance measurement that allows us to compare the quality of different candidate solutions. In EA terminology, the candidate solutions are named phenotypes, the individuals where the genetic operators are applied to are genotypes, and the mapping that assigns the phenotypes to the genotypes is denoted a representation or genotype-phenotype mapping. Genotypes can vary from binary strings to vectors of real numbers, to permutations, to prosecution rules, to schedules, up to program codes. When applying EAs, often different kinds of genotypes can be used to represent the same optimization problem. For example, when encoding trees (phenotypes), different genotypes like integer strings (Prüfer numbers), or vectors of real numbers (weighted encodings), can be used. Performance measurements for EAs can be based on a computer procedure, a simulation, an interaction with a human, or any combination of them. The purpose of the performance measurements is to distinguish between low-quality and high-quality solutions. In EA terminology, the performance resp. quality of a solution is named fitness.

In contrast to local search approaches like simulated annealing or tabu search, EAs maintain a population of solutions. A population consists of a number of individuals, and each individual represents one solution of the optimization problem. The initial population of candidate solutions (individuals) is usually generated randomly. Selection and variation operators are applied iteratively to a population. Selection operators compare the fitness of individuals in a population and remove – either deterministically or stochastically – low quality solutions. In parallel, additional copies of individuals with higher fitness are inserted. The heuristic search through the search space is performed by variation operators like crossover and mutation. In analogy to nature, recombination operators construct new solutions that are similar to the original solutions (parents) by randomly exchanging alleles between different individuals in a population. The application of mutation results in new solutions with similar properties.

Researchers have proposed many different variants of EAs in the literature. For illustrating the basic functionality of EAs we use the standard simple genetic algorithm (GA) used by Goldberg (1989). Simple GAs use recombination as the main search operator and mutation serves only as background noise. GAs are widely used and well understood (Goldberg, 2002). GAs use a constant population of size N, the genotypes consist of strings with fixed length l, and recombination operators are directly applied to the genotypes. The basic functionality of a simple GA is illustrated in table 1. After randomly creating and evaluating an initial population, the algorithm continuously creates new generations. New generations are created by recombining the selected highly

fit individuals and applying mutation to the offspring.

• initialize population
create initial population
evaluate individuals in initial population
• create new populations iteratively
select fit individuals for reproduction
generate offspring with recombination operator
mutate offspring
evaluate offspring

Table 1: Basic functionality of an EA

EAs have been applied to a large number of different optimization problems in many different application fields. To give a full survey over all EA applications is not possible due to the large amount of work that has been done in this field. For selected applications the reader is referred to Bäck, Fogel, and Michalewicz (1997). More examples for EA applications can be found in the literature or the main EA conferences.

3.2 Design Parameters of Evolutionary Algorithms

EAs can be classified according to the character of the main search operator used. It can be distinguished between approaches that use either recombination or mutation as the main search operator. A common representative of recombination-based EAs is the simple GA, which was already described in the previous paragraphs. A prominent example for mutation-based EA approaches are evolution strategies (ES) (Rechenberg, 1973; Schwefel, 1975; Schwefel, 1995). GAs and ES follow different optimization paradigms and make different assumptions about the character of the optimization problem they are applied to. Therefore, they are suited for different types of problems.

Mutation-based EA approaches like ES mainly rely on mutation and are common for continuous genotypes and parameter optimization problems. Although ES show the same basic functionality as GAs (compare table 1), recombination is seen only as a form of statistical error correction for wrong mutation steps (Beyer, 1995). Because ES use mutation (local search) as the main search operator, they perform well if the structure of the search space allows the population to move towards the high-quality solutions using iterated mutation steps. Therefore, the use of ES is only promising if similar solutions have similar properties and quality. Otherwise, if the fitnesses of similar genotypes are uncorrelated, no systematic local search is possible and ES perform like random search (Jones & Forrest, 1995). As ES have problems to overcome local optima in the search space, they use a population of individuals, which guarantees diversity, and they vary mutation step size to escape from local optima. As mutation is the main search operator, the most relevant design parameters are the length and the direction of the mutation step, the properties of the initial population, and the search strategy which determines how already found solutions influence the next search steps.

In recombination-based EA approaches (Holland, 1975; Goldberg, 1989) like the simple GA, recombination is the main search operator. Mutation is seen as background noise and results in minor modifications of the genotypes. GAs

mainly use binary strings as genotypes and recombination operators should create offspring that are similar to their parents by transferring meaningful sub-structures (building blocks) from the parents to the offspring (Goldberg, 1989). Common recombination operators are uniform and n-point crossover. When using uniform crossover, it is decided independently for every single allele of the offspring from which parent it inherits the value of the allele. For n-point crossover, n crossover points are randomly chosen in the string and two offspring are created from two parents by swapping the substrings. Because recombination-based EAs rely on the transfer of meaningful sub-structures from parent to offspring, problems that can be solved by such EAs must be quasi-decomposable that means it must be possible to create good solutions by combining good sub-structures (Goldberg, 1989). Otherwise, if a problem can not be decomposed, GAs show low performance (Goldberg, 2002).

The most relevant design parameters for GAs are the choice of the recombination operator and the used population size N. Recombination operators must ensure that meaningful sub-structures are not destroyed when creating the offspring from the parent (Thierens, 1995). Harik et al. (1999) presented a model that describes how the success probability P_n of GAs depends on the characteristics of the optimization problem and on the population size N. For GAs using binary strings and a proper crossover operator, P_n goes with $O(1-e^{-N})$. Therefore, with increasing population size the solution quality increases.

Other design criteria for EAs – GAs as well as ES – are the termination criteria, the selection operator, and the problem representation used. The termination criteria determines when the search should be stopped. For GAs using only crossover and no mutation, the run can be stopped after the population is fully converged (all individuals represent the same solution). Common selection operators are rank-based selection and fitness-proportional selection (compare Bäck et al. (1997, section C2)). The choice of the genotypes and the corresponding representation has a strong impact on EA performance (Liepins & Vose, 1990; Rothlauf, 2002). Users have to choose what kind of genotype they want to use for encoding the problem. Often there are different choices like binary strings, integers, or more complex genotypic structures. Furthermore, a representation must be defined that determines which phenotypes are represented by which genotypes.

3.3 A Proper Representation for the MCST Problem: The Link-biased Encoding

The *link-biased* (LB) encoding (Palmer, 1994; Rothlauf & Goldberg, 2003) is a weighted representation that encodes trees using real-valued alleles. When using this representation in combination with metaheuristics, problem-specific properties of the MCST problem (compare section 2.3) can be incorporated by adjusting the representation parameters properly.

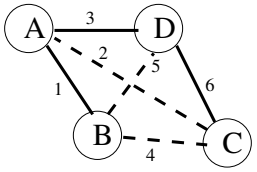


Figure 2: An example tree. The numbers indicate the number of a link.

3.3.1 Functionality

The general idea of the LB encoding is to represent a tree using a bias for every edge of the graph and to modify the distance weights w_{ij} of the underlying optimization problem according to these biases. A tree is constructed from the modified distance weights w'_{ij} by calculating the MST. Therefore, when using the LB encoding, a genotypic chromosome b holds biases for the links, and has length $n(n-1)/2$ for an n node network. When constructing the phenotype (the tree) from the genotype (the bias vector b containing the $n(n-1)/2$ biases), the genotypic biases are temporally added to the original distance weights w_{ij}. To get the represented tree, Prim's algorithm (Prim, 1957) is used to find the MST using the modified distance weights w'_{ij}. By running Prim's algorithm, links with low w'_{ij} will be used with high probability, whereas edges with high w'_{ij} will not exist in the tree. To finally get the tree's fitness, the encoded tree is evaluated by using the original distance weight matrix W and demand matrix R.

The weights b_k are floating values between zero and one. The original distance weights w_{ij} are modified by the elements of the bias vector b_k as

$$w'_{ij} = w_{ij} + P_1 b_k w_{max} \tag{2}$$

where w'_{ij} are the modified distance weights, w_{max} is the largest distance weight of a link ($w_{max} = \max(w_{ij})$), P_1 is the link-specific bias, and $k \in \{1, 2, \ldots, n(n-1)/2\}$ indicates the number of a link. The parameter P_1 controls the influence of the biases b_k and has a large impact on the structure of the tree. For $P_1 = 0$ the biases have no influence and only the MST calculated based on w_{ij} can be represented. The construction of the phenotype can be implemented with a Fibonacci heap and goes with $O(n^2)$. The structure of the represented tree depends not only on the bias values b_k, but also on the given distance weights w_{ij}. Therefore, the same link-biased individual can represent different trees if different distance weight matrices W are used.

We illustrate the construction of a tree from a bias vector b_k for a small example. For representing a tree with $n = 4$ nodes the genotype has length $l = n(n-1)/2 = 6$. For the example we want to use the link-biased individual $b = \{0.1, 0.6, 0.2, 0.1, 0.9, 0.3\}$. With $P_1 = 1$ and using the distance weights $w = \{10, 30, 20, 40, 10, 20\}$ we can calculate the modified cost according to equation 2 as $w' = \{14, 54, 28, 44, 46, 32\}$. Notice that $w_{max} = 40$. The represented tree that is calculated as the MST using the modified link costs w' is shown in figure 2. The six possible edges are labeled from 1 to 6 and the tree consists of the edges between A and B (link 1 with $w'_{AB} = 14$), A and D (link 3 with

$w'_{AD} = 28$), and C and D (link 6 with $w'_{CD} = 32$).

The LB encoding is a specialized version of the more general link-and-node-biased (LNB) encoding, which was originally proposed by Palmer (1994). In contrast to the LB encoding, the LNB encoding uses additional biases for each node. Therefore, the length of an LNB-encoded individual is $n(n-1)/2 + n$. Abuali, Wainwright, and Schoenefeld (1995) compared the LNB encoding to some other representations for the probabilistic MST problem and in some cases found the best solutions by using the LNB encoding. Later, Raidl and Julstrom (2000) proposed a variant of this encoding and observed solutions superior to those of several other representations for the degree-constrained MST problem. For the same type of problem, Krishnamoorthy and Ernst (2001) proposed another version of the LNB encoding. Gaube and Rothlauf (2001) investigated the properties of the LNB encoding and showed that due to the additional node biases not all possible trees can be encoded, and trees that are similar to a star are encoded with higher probability. Therefore, the LNB encoding with additional node-biases is only useful if the optimal solutions are similar to stars.

3.3.2 Properties of the LB encoding

A representation is denoted to be *redundant* if the number of possible genotypes exceeds the number of different phenotypes. The LB encoding describes how a finite number of different trees (n^{n-2}) can be represented by genotypes consisting of $n(n-1)/2$ real values. Therefore, the LB representation is a redundant representation as each tree (phenotype) can be represented by an infinite number of different genotypes. It was shown in Rothlauf and Goldberg (2003) that for large values of the link-specific bias ($P_1 \to \infty$), the LB encoding becomes *uniformly redundant*. This means, every possible tree is represented by the same number of different genotypes. With decreasing P_1, the LB encoding becomes *non-uniformly redundant* and solutions similar to the MST are over-represented. Then, a randomly chosen LB-encoded individual represents trees that are similarly to the MST with higher probability. At the extreme, if $P_1 = 0$, only one tree – the MST regarding the distance weights w_{ij} – can be represented as the link biases b_k have no impact on w'_{ij}.

Figure 3 summarizes the results from Rothlauf and Goldberg (2003) and illustrates how the LB encoding over-represents trees that are similar to the MST. The plots show the probability P_r that a link of a randomly generated LB-encoded tree is part of the MST over the link-specific bias P_1. Results are presented for different tree sizes n (16 and 28 nodes) and for different values of P_1. The plots show the mean and the standard deviation of P_r for the LB encoding as well as for a non-redundant representation (for example Prüfer numbers) that represents all possible trees with the same probability. It can be seen that for large values of P_1 ($P_1 > 100$), all n^{n-2} possible trees are represented uniformly that means no particular links are over-represented by the LB encoding. With decreasing P_1, randomly created LB-encoded trees contain links that are also used in the MST with higher probability. For small values of P_1, all edges of a randomly created individual are with high probability P_r also part of the MST. For $P_1 \to 0$, only the MST can be encoded.

This means that if P_1 is too small, not all possible trees can be represented. Therefore, it has to be examined how large P_1 must be chosen to allow the LB encoding to represent all possible trees. Prim's algorithm, which is used to create the tree from the w'_{ij}, uses a sorted list of edges (which is sorted according to w'_{ij}) and inserts iteratively edges with lowest w'_{ij} into the tree. Because the LB encoding modifies the original distance weights w_{ij} (equation 2), it also modifies this ordered list of edges and thus allows the encoding of trees that are different from the MST. If $P_1 \geq 1$ (and $b_k = 1$), $P_1 b_k \max(w_{ij})$ can be equal or greater than the highest original distance weight $\max(w_{ij})$. Then, the encoding can completely change the ordered list of edges as for an edge from node l to m the original $w_{lm} = \min(w_{ij})$ can be changed to $w'_{lm} = \min(w_{ij}) + P_1 \max(w_{ij}) > \max(w_{ij})$ if $P_1 > 1$. As a result, it is possible to encode all n^{n-2} different trees using the LB encoding if the link-specific bias is set to $P_1 \geq 1$.

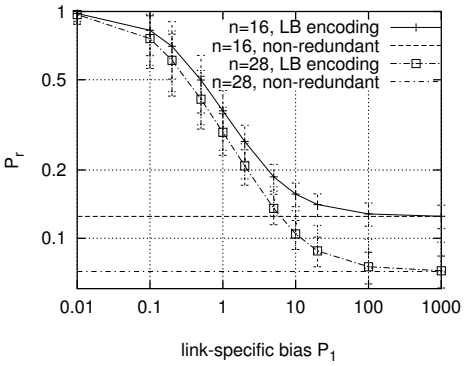


Figure 3: Probability that a link of a randomly generated LB-encoded tree is part of the MST over the link-specific bias P_1.

Based on this observation we are in the position to offer a recommendation for choosing P_1. With lower P_1 the LB encoding over-represents trees that are similar to the MST. As we have seen in section 2.3, where high-quality solutions of the MCST problem are similar to the MST, we expect higher performance when using the LB encoding with low values of P_1. However, if $P_1 < 1$, there is the possibility that some trees can not be represented. Therefore, it can be recommended to set $P_1 \approx 1$. This ensures that all possible solutions can be encoded while still over-representing solutions similar to the MST.

3.4 Other Representations for Trees

The following paragraphs shortly describe some representations that should serve as a benchmark for the LB encoding in section 4.

Prüfer numbers are a one-to-one mapping between trees and a string of length $n-2$ over an alphabet of n symbols. Therefore, it is possible to derive a unique tree with n nodes from the Prüfer number of length $n-2$ and vice versa (Even, 1973, pp. 104-106). The construction and deconstruction process from the Prüfer number to the encoded tree and vice versa is lengthy and can be found in detail in the literature (e.g. Rothlauf (2002)). Although, Prüfer numbers do not allow efficient heuristic optimization (compare section 2.2), they have been commonly used in the literature and should therefore be used as a worst-case scenario.

The characteristics vector (CV) encoding uses a binary vector of length $n(n-1)/2$ that indicates whether a possible link is used in the graph. All possible links are numbered, and each link must be assigned to a position in the vector. When using CV for the encoding of trees, every CV must have

exactly $n-1$ ones, the represented graph must be connected, and there are no cycles allowed. This makes the construction of trees from randomly chosen CV demanding as most of the randomly generated CV are infeasible, and do not encode trees. Therefore, to ensure that a CV encoded solution always represents a valid tree, repair mechanisms are necessary. The most common approach (compare e.g. Berry et al. (1994)) is to remove in a first step the links (set the allele in the corresponding CV to 0) that cause cycles in the represented graph, and in a second step to add links to connect disconnected subtrees until the represented tree is fully connected.

The CV encoding does not allow to store information about the importance of different links. Therefore, weighted encodings have been introduced that use continuous, instead of binary values, for the encoding of a tree. The weights are used by construction algorithms that create a tree from a vector of weights. Besides the LB and LNB encoding (compare section 3.3), common weighted encodings are the NetKey encoding (Rothlauf et al., 2002), or the weighted encoding from Raidl and Julstrom (Raidl & Julstrom, 2000). NetKeys are based on the random key encoding (Bean, 1992; Bean, 1994), which can be used for the encoding of permutations. The NetKey encoding is similar to the LB encoding and represents a tree as a continuous vector of length $n(n-1)/2$. The represented tree is constructed from the list of weights by Kruskal's algorithm avoiding links that result in a cycle. NetKeys show similar performance as the LB encoding using a large link-specific bias ($P_1 \to \infty$) (Rothlauf & Goldberg, 2003). NetKeys are uniformly redundant, this means the performance of metaheuristics is independent of the structure of the optimal solution.

4 Performance Evaluation

The following section demonstrates how the MCST problem can be efficiently solved by using metaheuristics and how the performance of metaheuristics can be increased by the use of the LB encoding with a proper choice of P_1. Section 4.1 compares the performance of different metaheuristics for existing problem instances from literature and section 4.2 examines the performance of EAs and simulated annealing for randomly created MCST problem instances of different type and size.

4.1 Existing Problem Instances

Test instances for the MCST problem have been proposed by Palmer (1994), Berry et al. (1995), and Raidl (2001). Palmer (1994) described MCST problems with six (palm6), twelve (palm12), and 24 (palm24) nodes. The nodes correspond to cities in the United States and the distances between the nodes are obtained from a tariff database. The inter-node traffic demands are inversely proportional to the distances between the nodes. Berry, Murtagh, and McMahon (1995) presented three instances, one with six nodes (berry6) and two with 35 nodes (berry35 and berry35u). For berry35u the distance weights $w_{ij} = 1$. Raidl (2001) proposed several test instances ranging from 10 to 100

nodes. The distances and the traffic demands were generated randomly and are uniformly distributed in the interval $[0, 100]$.

problem instance	n	$w(T_{opt})$	$\mu(d_{mst,rand})$	$d_{mst,opt}$	P_{suc}								
									LB				
					N	CV	Prüfer	NetKeys	$P_1 = 0.05$	$P_1 = 0.5$	$P_1 = 1$	$P_1 = 2$	$P_1 = 100$
palmer6	6	693 180	3.36	1	16	0.42	0.07	0.365	0.275	0.88	0.66	0.495	0.325
palmer12	12	3428509	9.17	5	300	0.645	0.21	0.4	0	0.945	0.83	0.67	0.395
palmer24	24	1086656	21.05	12	800	0	0	0.71	0	0.08	0.50	0.67	0.75
raidl10	10	53674	7.20	3	70	0.785	0.09	0.81	0	1	1	1	0.77
raidl20	20	157570	17.07	4	450	0	0	0.435	0	0.975	0.955	0.88	0.455
berry6	6	534	3.51	0	16	0.71	0.185	0.475	1	1	0.98	0.88	0.505
berry35u	35	16273	-	-	2000	0	0	0.04	0.08	0.42	0.32	0.25	0.05
berry35	35	16915	32.05	0	300	0	0	0.025	1	1	1	0.985	0.02

Table 2: Properties of test problems from the literature and success probabilities P_{suc} of a GA using different types of representations. The results indicate high and robust GA performance when using the LB encoding with $P_1 \approx 1$.

Table 2 lists the properties of the test instances. It shows for the different test problems the number n of nodes, the cost $w(T_{opt})$ of the optimum solution T_{opt}, the mean distance $\mu(d_{mst,rand})$ between the MST and a randomly created tree, and the distance $d_{mst,opt}$ between the MST and the optimum solution. The distances are measured according to equation 1 and the optimum solutions are the best ever found (or best proven) solutions from the literature. As $w_{ij} = 1$ for the berry35u problem, all solutions are MSTs and $d_{mst,rand}$ and $d_{mst,opt}$ are not meaningful. For the calculation of $\mu(d_{mst,rand})$, 10 000 random trees have been generated for every problem and their average distance towards the MST have been calculated. The results reveal that the distances $d_{mst,opt}$ between the optimum solution and the MST are significantly lower than the average distances $\mu(d_{mst,rand})$ between randomly created trees and the MST and therefore the optimum solutions are biased towards the MST.

Furthermore, Table 2 compares the performance of a standard GA using the CV encoding, Prüfer numbers, NetKeys, and the LB encoding with different link-specific bias P_1 for the different test problems. As the optimum solutions are biased towards the MST, it is expected that representations like the LB encoding that can encode solutions similar to the MST with a higher probability result in higher performance in comparison to unbiased representations. To compare the performance of the different representations, a standard GA with only crossover and no mutation ($P_{cross} = 1$ and $P_{mut} = 0$), standard tournament selection without replacement (tournament size of two), and uniform crossover was used. The GA is run 200 times for each of the test instances with population size N, and P_{suc} denotes the percentage of these 200 runs where the optimal solution was found. As discussed in section 3.2, GA performance increases with greater N. Therefore, N is chosen such that $\max(P_{suc}) \approx 0.8 \ldots 1$.

The results in table 2 indicate that GA performance is about maximum when using the LB encoding with a small link-specific bias $P_1 \approx 1$. For some problem instances, a higher (palmer24) or lower (palmer6 and palmer12) value of P_1 can slightly increase GA performance, but choosing $P_1 \approx 1$ results on average in a good and robust GA performance. As expected, GA performance is very

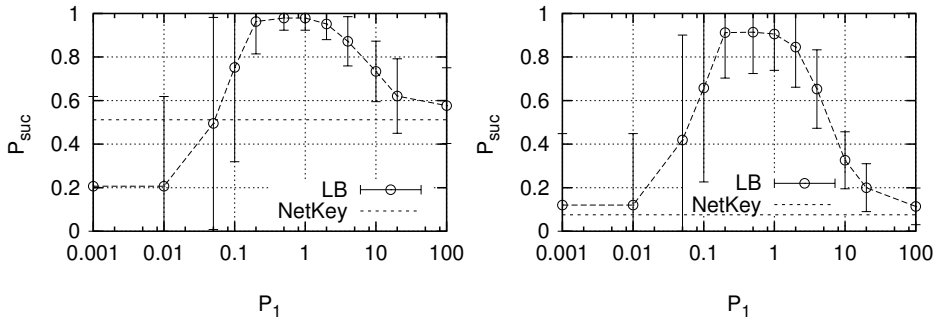
low for small values of P_1 as only the MST can be encoded (except for berry6 where the optimal solution is the MST). For high values of P_1 ($P_1 = 100$), all possible trees are represented with the same probability, and GAs using the LB encoding show similar performance as for NetKeys. GAs using Prüfer numbers and CV encoding perform as expected. The CV encoding shows only good results for small problem instances ($n < 10$), and Prüfer numbers fail for all different problems.

4.2 Random Problem Instances

In the previous section, it was shown that GAs using the LB encoding allow efficient problem solving for the existing test instances from literature. In this section, the performance of metaheuristics (GA and simulated annealing) using the LB encoding is examined for randomly created MCST problem instances. Section 2.3 has demonstrated that the optimum solutions for randomly created MCST problems are biased towards the MST. Using the LB encoding with proper link-specific bias P_1 allows us to exploit this property of MCST problems and to create metaheuristics that solve the MCST problem fast and reliable.

For the experiments, we created 100 random problem instances for different problem sizes ($8 \leq n \leq 22$). The real-valued demands r_{ij} are generated randomly and are uniformly distributed in the interval $[0, 100]$. The distance weights w_{ij} are generated either randomly in the interval $[0, 100]$ (random w_{ij}), or calculated as the Euclidean distances between the nodes placed randomly on a two-dimensional grid of size 1000x1000 (Euclidean w_{ij}). The performance of a metaheuristic is measured again using the success probability P_{suc} which describes the percentage of runs that find the optimum solution. As there are no exact optimization algorithms available that can solve even small instances of the MCST problem in a reasonable time, an iterated GA using the NetKey encoding is used for finding the optimum (or at least near-optimal) solution. NetKeys ensure good GA performance and represent all possible trees uniformly.

The GA used for finding the optimal solutions uses uniform crossover and tournament selection without replacement (tournament size two). The crossover probability is set to $p_{cross} = 0.7$ and the mutation probability (assigning a random value $[0, 1]$ to one allele) is set to $p_{mut} = 1/l$, where l is the string length. The GA is stopped after 200 generations are exceeded. To find the optimum solution for an MCST problem, this GA is applied n_{iter} times using a population size of N_0. T_0^{best} denotes the best solution that is found during the n_{iter} runs. In a next round, we double the population size and again apply a GA n_{iter} times with a population size of $N_1 = 2N_0$. T_1^{best} denotes the best solution that can be found in the second round. We continue this iteration and double the population size $N_i = 2N_{i-1}$ until $T_i^{best} = T_{i-1}^{best}$ and $n(T_i^{best})/n_{iter} > 0.5$, this means T_i^{best} is found in more than 50% of the runs in round i. $n(T_i^{best})$ denotes the number of runs that find the best solution T_i^{best} in round i. The optimum solutions for the 100 randomly generated problem instances of each size n are determined by an iterated GA with $N_0 = 200$ and $n_{iter} = 10$. The computational effort of finding the optimum solutions for all 100 problem in-



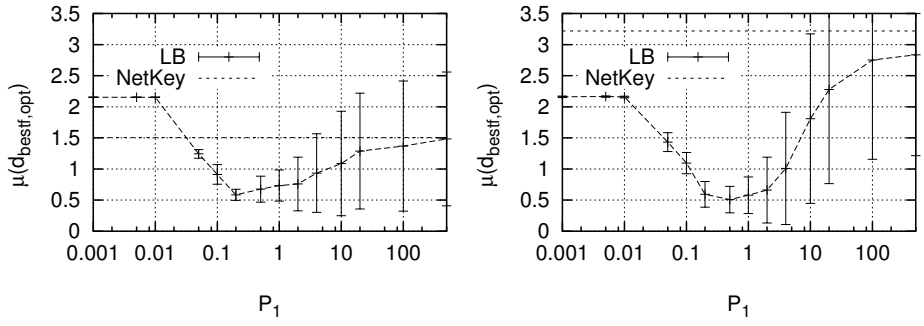
(a) GA, 16 nodes, random w_{ij} (b) SA, 16 nodes, random w_{ij}

Figure 4: Success probability P_{suc} of a GA (left) and a SA (right) for 100 randomly created 16 node MCST problems over the link-specific bias P_1. The plots show that the performance of both metaheuristics becomes maximum if the LB encoding with $P_1 \approx 1$ is used.

stances is high (e.g. some 100 hours of computing time for 100 problems of size $n = 22$).

To be able to make more general statements about the performance of metaheuristics using the LB encoding, we use a simple GA as well as a simulated annealing (SA) approach. The simple GA using only crossover and no mutation ($P_{cross} = 1$ and $P_{mut} = 0$) is a representative example for crossover-based search, whereas the SA is a representative for mutation-based search. In the SA a new solution is iteratively created by applying one mutation step to the current solution. In our experiments, a mutation means randomly exchanging the position of two values in the encoded solution. If the new solution has higher fitness it replaces the original solution. If it has lower fitness it replaces the original solution with probability $P(T) = \exp(\frac{w(T_{new})-w(T_{original})}{T})$. $P(t)$ depends on the temperature T which is reduced during the run according to a cooling schedule. With lowering T, the probability of accepting worse solutions decreases. Because SA uses only mutations, and solves in contrast to, for example a $(1+1)$ evolution strategy, multi-modal problems more easily, it is used as a representative example of mutation-based EAs. For further information about SA, the reader is referred to van Laarhoven and Aarts (1988) or Davis (1987).

Figure 4 shows the success probability P_{suc} (number of runs that find the optimum solution) of a GA resp. SA over the link-specific bias P_1 for 100 randomly created MCST problem instances with problem size $n = 16$ and randomly chosen distance weights w_{ij}. For each problem instance, 50 runs were performed and P_{suc} was averaged over all 100 problem instances. In figure 4(a), we used a standard GA with population size $N = 200$ using uniform crossover ($P_{cross} = 1$), no mutation ($P_{mut} = 0$), and tournament selection without replacement (tournament size two). Each GA run was stopped either after all individuals in the population represented the same tree, or the number of generations exceeded 200. In figure 4(b) we used a SA strategy with starting temperature $T_0 = 5000$. In each iteration the temperature T was reduced by



(a) GA, 16 nodes, random w_{ij}

(b) SA, 16 nodes, random w_{ij}

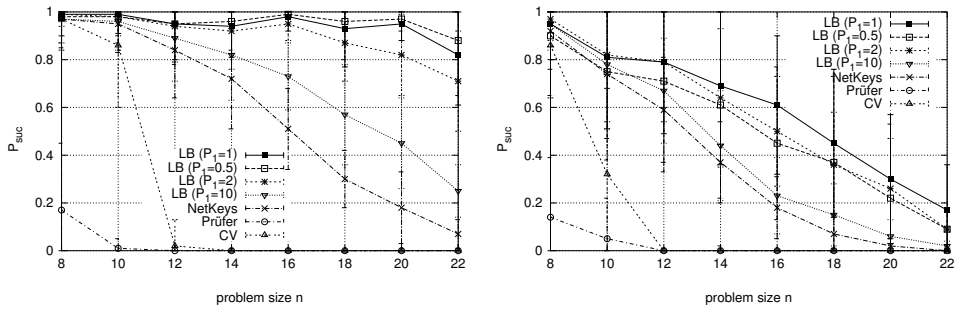
Figure 5: Average distance $\mu(d_{bestf,opt})$ between the best solution that has been found by a GA (left) and a SA (right) and the optimum solution over the link-specific bias P_1 for 100 randomly created MCST problems. $d_{bestf,opt}$ is minimum for $P_1 \approx 1$.

0.999 ($T_i = 0.999 * T_{i-1}$) and the SA was stopped after 5000 iterations.

The results show that GA resp. SA performance becomes maximum when using the LB encoding with $P_1 \approx 1$. A pairwise t-test is performed on the success probabilities P_{suc} of GA, resp. SA and for $0.1 \leq P_1 \leq 10$ the LB encoding outperforms the NetKey encoding with high significance ($p < 0.001$) supporting the hypothesis "GA resp. SA shows higher performance using the LB encoding with $0.1 \leq P_1 \leq 10$ than using NetKeys". Furthermore, as expected, both metaheuristics show similar performance for the LB encoding with a large P_1 and NetKeys. For large values of P_1 the LB encoding becomes uniformly redundant and all possible solutions are represented with the same probability. For low values of P_1, P_{suc} becomes small as only MCST problems can be solved, where the optimum solution is the MST.

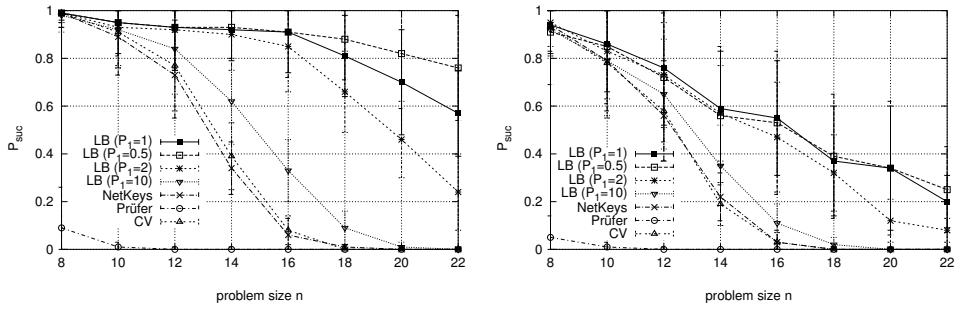
To better understand why the LB encoding shows maximum performance for $P_1 \approx 1$, figure 5 shows the average distance $\mu(d_{bestf,opt})$ between the best solution T_{bestf} that has been found by either the GA or the SA and the optimal solution T_{opt} for all 100 randomly generated 16 node MCST problems. The results confirm the findings described in section 2.3 and 3.3.2. Because optimum solutions are similar to the MST, and because the LB encoding encodes solutions similar to the MST with higher probability, $d_{bestf,opt}$ decreases with lower P_1. However, if P_1 is too small ($P_1 < 1$), not all possible solutions can be represented and $d_{bestf,opt}$ increases again. For $P_1 \to 0$ only the MST can be encoded and $d_{bestf,opt}$ becomes the average distance $\mu(d_{mst,opt})$ between the MST and the optimum solution (compare figure 1).

The remaining paragraphs draw a more comprehensive picture of the performance of GA and SA for MCST problems with different types of distance weights w_{ij} and problem sizes n. Figure 6 summarizes the results and compares the performance of a GA and SA using different representations (NetKeys, Prüfer numbers, CV encoding, and the LB encoding with different P_1 ($P_1 = 0.5$, $P_1 = 1$, $P_1 = 2$, and $P_1 = 100$)). Each plot shows the probability P_{suc} of finding



(a) GA, random w_{ij}

(b) GA, Euclidean w_{ij}



(c) SA, random w_{ij}

(d) SA, Euclidean w_{ij}

Figure 6: Success probability P_{suc} of GA (top), resp. SA (bottom) over the problem size n for randomly generated MCST problems. The distance weights w_{ij} are chosen either randomly (left), or according to the Euclidean distances (right) between the nodes that are placed randomly on a two-dimensional grid. The plots show that GA, resp. SA show maximum performance when using the LB encoding with $P_1 \approx 1$.

the optimum solution over the problem size n. As before, we randomly generated 100 problem instances for each problem size n and performed 50 runs for each representation. In figure 6(a) (random w_{ij}) and 6(b) (Euclidean w_{ij}), a GA using the same parameters as described in the previous paragraphs (compare figure 4(a)) is used. In figure 6(c) (random w_{ij}) resp. 6(d) (Euclidean w_{ij}), results for an SA using the parameters from above (compare figure 4(b)) are presented. For Prüfer numbers, a SA search step results in a random change of one of the $n-2$ alleles, and for the CV encoding a search step results in randomly changing one edge (compare section 3.4).

The plots show similar results for GA and SA as well as for MCST problems with random and Euclidean distance weights. First of all, it can be seen that GA resp. SA performance decreases with increasing problem size n as the parameters of the metaheuristics are fixed and the larger problem instances are more difficult to solve. To get better results, either larger population sizes N (GA), or a different cooling schedule combined with a higher number of search steps (SA) would be necessary. Secondly and more importantly, the results

from the previous paragraphs can be confirmed as GA, resp. SA using the LB encoding with $P_1 \approx 1$ outperform other representations like the NetKey encoding, the CV encoding, or the Prüfer number encoding. The plots show that Prüfer numbers fail independently of the used search heuristic, and both GA as well as SA using them show low performance. GA using the CV encoding perform well if the problem instances are small ($n < 10$) but fail when applied to larger problem instances. The situation is different for the mutation-based SA approach as no repair mechanisms for invalid solutions are necessary (compare section 3.4). Each search step (changing one edge) results in a new and feasible solution, and SA using the CV encoding shows similar performance as NetKeys.

As the performance of Prüfer numbers (always) and CV encoding (for crossover-based search) is low, we focus on the comparison between NetKeys and the LB encoding. The plots in figure 6 indicate that the LB encoding with $P_1 \approx 1$ outperforms the other encodings. Consequently, we test the hypothesis H_0 that the performance of the LB encoding with $P_1 = 1$ does not outperform the NetKey encoding. Table 2 shows the result of a pairwise t-test. The numbers show that for smaller problem instances $n < 12$ there is no significant difference between NetKeys and the LB encoding. However, for larger problem instances, the differences are significant (p-value < 0.0001), thereby rejecting the null hypothesis H_0 and supporting the alternative hypothesis that the LB encoding with $P_1 = 1$ outperforms the NetKey encoding.

			problem size n							
	w_{ij}		8	10	12	14	16	18	20	22
GA	random	test statistic t	1.44	2.73	4.48	6.00	13.80	13.36	20.74	24.25
		p-value	0.15	0.007	10^{-5}	10^{-8}	10^{-15}	10^{-14}	10^{-17}	10^{-32}
	Euclidean	test statistic t	1.10	1.73	5.18	8.06	9.42	8.67	7.04	4.60
		p-value	0.27	0.08	10^{-7}	10^{-13}	10^{-14}	10^{-12}	10^{-9}	10^{-5}
SA	random	test statistic t	1.09	2.73	7.71	23.57	33.26	33.04	20.99	20.41
		p-value	0.27	0.007	10^{-13}	10^{-34}	10^{-55}	10^{-35}	10^{-24}	10^{-23}
	Euclidean	test statistic t	0.20	2.41	4.51	9.91	13.61	10.49	7.82	7.01
		p-value	0.84	0.016	10^{-5}	10^{-14}	10^{-17}	10^{-13}	10^{-9}	10^{-6}

Table 3: Results of a pairwise t-test on the null hypothesis that using the LB encoding with $P_1 = 1$ does not outperform the NetKey encoding ($P_{suc}(LB \text{ with } P_1 = 1) \leq P_{suc}(NetKey)$). The results show that for larger problem instances ($n > 10$) there is evidence to reject the null hypothesis in favor of the alternative hypothesis that the LB encoding outperforms NetKeys.

5 Conclusions

This paper demonstrated how efficient metaheuristics can be developed by considering properties of the optimization problem for the design of problem representations. Metaheuristics like evolutionary algorithms or simulated annealing

only describe a general optimization principle, and to successfully apply such techniques to complex optimization problems, problem-specific adaptations are necessary. The design of efficient metaheuristics by using problem-specific representations is shown for the minimum communication spanning tree (MCST) problem, which finds a spanning tree that connects all nodes and satisfies their communication requirements for a minimum total cost. A representation (the LB encoding) is developed such that trees that are similar to the MST are encoded with a higher probability than random trees. The LB encoding makes use of the problem-specific property of the MCST problem that optimal solutions are similar to the MST. The LB encoding can be used for different types of metaheuristics and allows different types of metaheuristics to benefit from existing knowledge about the MCST problem.

Because the LB encoding considers knowledge about optimum solutions for the MCST problem, and encodes solutions similar to the MST with higher probability, its use can significantly increase the performance of metaheuristics for the MCST problem. Benchmarking results show that local search methods (simulated annealing) as well as recombination-based metaheuristics (genetic algorithms) using the LB encoding significantly outperform other representations for existing problem instances from the literature as well as for randomly created MCST problem instances of different type and size. A proper setting of the representation-specific parameter $P_1 \approx 1$ is important as this parameter controls the over-representation of MST-like solutions. If P_1 is too high ($P_1 \gg 1$) all possible trees are encoded with the same probability; if P_1 is too low ($P_1 \to 0$) the MST alone can be encoded and heuristic search is no longer possible.

This work demonstrated how efficient metaheuristics can be developed for a problem where no efficient algorithmic or approximation algorithms are available. The results of this work should encourage researchers to investigate whether other types of tree problems have similar properties like the MCST problem. The authors also believe that optimum solutions for other constraint tree problems are also similar to the MST. Therefore, using the LB encoding for solving such problems is a promising direction of future research and application. Furthermore, the design approach for metaheuristics illustrated in this paper should be applied to other optimization problems. It was demonstrated that using problem-specific knowledge for the design of metaheuristics is possible and results in high-quality optimization methods. Applying these design principles to other problems would allow us to create metaheuristic-based optimization methods that are able to solve difficult problems fast and reliably.

References

Abuali, F. N., Wainwright, R. L., & Schoenefeld, D. A. (1995). Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem. See Eschelman (1995), pp. 470–477.

Alon, N., Karp, R. M., Peleg, D., & West, D. (1995). A graph theoretic game

and its application to the k-server problem. *SIAM Journal on Computing*, 78–100.

Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.) (1997). *Handbook of Evolutionary Computation*. Bristol and New York: Institute of Physics Publishing and Oxford University Press.

Bartal, Y. (1996). Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. 37th IEEE Symp. on Foundations of Computer Science* (pp. 184–193).

Bartal, Y. (1998). On approximating arbitrary metrics by tree metrics. In *Proc. 30th Anual ACM Symp. on Theory of Computer Science* (pp. 161–168).

Bean, J. C. (1992, June). *Genetics and random keys for sequencing and optimization* (Technical Report 92-43). Ann Arbor, MI: Department of Industrial and Operations Engineering, University of Michigan.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, *6*(2), 154–160.

Berry, L. T. M., Murtagh, B. A., & McMahon, G. (1995). Applications of a genetic-based algorithm for optimal design of tree-structured communication networks. In *Proceedings of the Regional Teletraffic Engineering Conference of the International Teletraffic Congress* (pp. 361–370). Pretoria, South Africa.

Berry, L. T. M., Murtagh, B. A., McMahon, G., & Sugden, S. (1997). Optimization models for communication network design. In *Proceedings of the Fourth International Meeting Decision Sciences Institute* (pp. 67–70). Sydney, Australia.

Berry, L. T. M., Murtagh, B. A., McMahon, G., Sugden, S., & Welling, L. (1999). An integrated GA–LP approach to communication network design. *Telecommunication Systems*, *12*(2), 265–280.

Berry, L. T. M., Murtagh, B. A., & Sugden, S. J. (1994). A genetic-based approach to tree network synthesis with cost constraints. In Zimmermann, H. J. (Ed.), *Second European Congress on Intelligent Techniques and Soft Computing - EUFIT'94*, Volume 2 (pp. 626–629). Promenade 9, D-52076 Aachen: Verlag der Augustinus Buchhandlung.

Beyer, H.-G. (1995). Toward a theory of evolution strategies: On the benefits of sex - the (μ/μ,λ) theory. *Evolutionary Computation*, *3*(1), 81–111.

Cahn, R. S. (1998). *Wide area network design, concepts and tools for optimization*. San Francisco: Morgan Kaufmann Publishers.

Cayley, A. (1889). A theorem on trees. *Quarterly Journal of Mathematics*, *23*, 376–378.

Chang, S.-G., & Gavish, B. (1993). Telecommunications network topological design and capacity expansion: formulations and algorithms. *Telecommun. Syst.*, *1*, 99–131.

Charikar, M., Chekuri, C., Goel, A., Guha, S., & Plotkin, S. (1998, November). Approximating a finite metric by a small number of tree metrics. In *Proc. 39th IEEE Symp. on Foundations of Computer Science* (pp. 111–125).

Chu, C.-H., Chou, C., & Premkumar, G. (2000). Digital data networks design using genetic algorithms. *European Journal of Operational Research*, *127*(1), 140–158.

Crescenzi, P., & Kann, V. (2003, Aug.). A compendium of NP optimization problems. `http://www.nada.kth.se/theory/compendium`.

Davis, L. (1987). *Genetic algorithms and simulated annealing*. San Mateo, CA: Morgan Kaufmann.

Davis, L., Orvosh, D., Cox, A., & Qiu, Y. (1993). A genetic algorithm for survivable network design. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 408–415). San Mateo, CA: Morgan Kaufmann.

Dionne, R., & Florian, M. (1979). Exact and approximate algorithms for optimal network design. *Networks*, *9*, 39–59.

Eschelman, L. (Ed.) (1995). *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Even, S. (1973). *Algorithmic combinatorics*. New York: The Macmillan Company.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W. H. Freeman.

Gaube, T., & Rothlauf, F. (2001). The link and node biased encoding revisited: Bias and adjustment of parameters. In Boers, E. J. W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P. L., Raidl, G. R., Smith, R. E., & Tijink, H. (Eds.), *Applications of Eolutionary Computing: Proc. EvoWorkshops 2001* (pp. 1–10). Berlin: Springer.

Gavish, B. (1983). Formulations and algorithms for the capacitated minimal directed tree problem. *Journal of the ACM*, *30*(1), 118–132.

Gavish, B., & Altinkemer, K. (1990, Summer). Backbone network design tools with economic tradeoffs. *ORSA Journal on Computing*, *2*(3), 58–76.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (2002). *The design of innovation*. Series on Genetic Algorithms and Evolutionary Computation. Dordrecht, The Netherlands: Kluwer.

Gomory, R. E., & Hu, T. C. (1961). Multi-terminal network flows. In *SIAM Journal on Applied Math*, Volume 9 (pp. 551–570).

Gottlieb, J., Julstrom, B. A., Raidl, G. R., & Rothlauf, F. (2001). Prüfer numbers: A poor representation of spanning trees for evolutionary search.

In Spector, L., Goodman, E., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., & Burke, E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 2001* (pp. 343–350). San Francisco, CA: Morgan Kaufmann Publishers.

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Hu, T. C. (1974, September). Optimum communication spanning trees. *SIAM Journal on Computing*, *3*(3), 188–195.

Johnson, D. S., Lenstra, J. K., & Kan, A. H. G. R. (1978). The complexity of the network design problem. *Networks*, *8*, 279–285.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. See Eschelman (1995), pp. 184–192.

Julstrom, B. A. (2001, 7 July). The blob code: A better string coding of spanning trees for evolutionary search. In Wu, A. S. (Ed.), *Proceedings of the 2001 Genetic and Evolutionary Computaton Conference Workshop Program* (pp. 256–261). San Francisco, California, USA.

Kershenbaum, A. (1993). *Telecommunications network design algorithms.* New York: McGraw Hill.

Kim, J. R., & Gen, M. (1999). Genetic algorithm for solving bicriteria network topology design problem. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., & Porto, W. (Eds.), *Proceedings of the 1999 IEEE Congress on Evolutionary Computation* (pp. 2272–2279). IEEE Press.

Krishnamoorthy, M., & Ernst, A. T. (2001). Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, *7*, 587–611.

Krishnamoorthy, M., Ernst, A. T., & Sharaiha, Y. M. (1999). *Comparison of algorithms for the degree constrained minimum spanning tree* (Tech. Rep.). Clayton, Australia: CSIRO Mathematical and Information Sciences.

Li, Y. (2001). An effective implementation of a direct spanning tree representation in GAs. In Boers, E. J. W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P. L., Raidl, G. R., Smith, R. E., & Tijink, H. (Eds.), *Applications of evolutionary Computing: Proc. EvoWorkshops 2001* (pp. 11–19). Berlin: Springer.

Liepins, G. E., & Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, *2*, 101–115.

Lin, S. (1982). Effective use of heuristic algorithms in network design. In *Proceedings of Symposia in Applied Mathematics*, Volume 26 (pp. 63–84).

Palmer, C. C. (1994). *An approach to a problem in network design using genetic algorithms*. unpublished PhD thesis, Polytechnic University, Troy, NY.

Palmer, C. C., & Kershenbaum, A. (1994). Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, Volume 1 (pp. 379–384). Piscataway, NJ: IEEE Service Center.

Papadimitriou, C. H., & Yannakakis, M. (1991). Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, *43*, 425–440.

Peleg, D. (1997). Approximating minimum communication spanning trees. Proc. 4th Colloq. on Structural Information and Communication Complexity, Ascona, Switzerland.

Peleg, D., & Reshef, E. (1998). Deterministic polylog approximation for minimum communication spanning trees. *Lecture Notes in Computer Science*, *1443*, 670–682.

Picciotto, S. (1999). *How to encode a tree*. Doctoral dissertation, University of California, San Diego, USA.

Premkumar, G., Chu, C., & Chou, H. (2001). Telecommunications network design decision - a genetic algorithm approach. *Decision Sciences*, *31*(2), 483–506.

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, *36*, 1389–1401.

Prüfer, H. (1918). Neuer Beweis eines Satzes über Permutationen. *Archiv für Mathematik und Physik*, *27*, 742–744.

Raidl, G. R. (2001, February). Various instances of optimal communication spanning tree problems. personal communciation.

Raidl, G. R., & Julstrom, B. A. (2000). A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In Carroll, J., Damiani, E., Haddad, H., & Oppenheim, D. (Eds.), *Proceedings of the 2000 ACM Symposium on Applied Computing* (pp. 440–445). ACM Press.

Raidl, G. R., & Julstrom, B. A. (2003). Edge-sets: An effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, *7*(3), 225–239.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag.

Reeves, C. (1997). Genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, *9*, 231–250.

Reshef, E. (1999, April). *Approximating minimum communication cost spanning trees and related problems*. Master's thesis, Feinberg Graduate School of the Weizmann Institute of Science, Rehovot 76100, Israel.

Rothlauf, F. (2002). *Representations for genetic and evolutionary algorithms*. Number 104 in Studies on Fuzziness and Soft Computing. Berlin: Springer.

Rothlauf, F., Gerstacker, J., & Heinzl, A. (2003). *On the optimal communication spanning tree problem* (Technical Report 15/2003). University of Mannheim.

Rothlauf, F., & Goldberg, D. E. (1999). Tree network design with genetic algorithms - an investigation in the locality of the prüfernumber encoding. In Brave, S., & Wu, A. S. (Eds.), *Late Breaking Papers at the Genetic and Evolutionary Computation Conference 1999* (pp. 238–244). Orlando, Florida, USA: Omni Press.

Rothlauf, F., & Goldberg, D. E. (2000). Prüfernumbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature, PPSN VI* (pp. 395–404). Berlin: Springer-Verlag.

Rothlauf, F., & Goldberg, D. E. (2003). Redundant representations in evolutionary computation. *Evolutionary Computation*, *11*(4), 381–415.

Rothlauf, F., Goldberg, D. E., & Heinzl, A. (2002). Network random keys – A tree network representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, *10*(1), 75–97.

Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung*. Doctoral dissertation, Technical University of Berlin.

Schwefel, H.-P. (1995). *Evolution and optimum seeking*. New York: Wiley & Sons.

Sinclair, M. C. (1995). Minimum cost topology optimisation of the COST 239 European optical network. In Pearson, D. W., Steele, N. C., & Albrecht, R. F. (Eds.), *Proceedings of the 1995 International Conference on Artificial Neural Nets and Genetic Algorithms* (pp. 26–29). New York: Springer-Verlag.

Tang, K. S., Man, K. F., & Ko, K. T. (1997). Wireless LAN desing using hierarchical genetic algorithm. In Bäck, T. (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 629–635). San Francisco: Morgan Kaufmann.

Thierens, D. (1995). *Analysis and design of genetic algorithms*. Leuven, Belgium: Katholieke Universiteit Leuven.

Tzschoppe, C., Rothlauf, F., & Pesch, H.-J. (2004). The edge-set encoding revisited: On the bias of a direct representation for trees. In Deb, Kalyanmoy et al. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference 2004* (pp. unknown). Heidelberg: Springer.

van Laarhoven, P. J. M., & Aarts, E. H. L. (1988). *Simulated Annealing: Theory and Applications*. Dordrecht, The Netherlands: Kluwer.

Wolpert, D. H., & Macready, W. G. (1995). *No free lunch theorems for search* (Tech. Rep. No. SFI-TR-95-02-010). Santa Fe, NM: Santa Fe Institute.

Wong, R. (1980). Worst case analysis of network design problem heuristics. *SIAM J. Algebraic Discr. Meth.*, *1*, 51–63.

Wu, B. Y., Lancia, G., Bafna, Y., Chao, K. M., Ravi, R., & Tang, C. Y. (1998, January). A polynomial time approximation schem for minimum routing cost spanning trees. In *Proc. 9th ACM-SIAM Symp. on Discrete Algorithms* (pp. 21–32).

Zhou, G., & Gen, M. (1997). Approach to degree-constrained minimum spanning tree problem using genetic algorithm. *Engineering Design & Automation*, *3*(2), 157–165.