An Encoding in Metaheuristics for the Minimum Communication Spanning Tree Problem

Franz Rothlauf
Department of Information Systems, University of Mainz, Jakob Welder-Weg 9, 55099 Mainz, Germany,
rothlauf@uni-mainz.de

Problem-specific encodings can improve the performance of metaheuristics, such as genetic algorithms or simulated annealing. This paper studies the link-biased (LB) encoding, which is a tree representation, and applies metaheuristics using this encoding to the minimum communication spanning tree (MCST) problem. Given the communication requirements of the nodes, the MCST problem seeks a communication spanning tree with minimum total cost. Optimal solutions for MCST problems are similar to minimum spanning trees (MSTs), and the LB encoding exploits this property by encoding trees similar to MSTs with higher probability. The paper investigates how to systematically design problem-specific encodings for MCST problems, how to set the encoding-specific parameter that controls the bias of the LB encoding towards MSTs, and presents performance results for various MCST problems.

(*problem-specific representations, trees, metaheuristics, encodings, communication networks*)

---

1. Introduction

The minimum communication spanning tree (MCST) problem (Hu, 1974) seeks a spanning tree that connects the given nodes and satisfies their communication requirements with minimum total cost. This paper examines 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 similar to the minimum spanning tree (MST) with higher probability than random trees. For the MCST problem, optimal solutions are similar to MSTs (Rothlauf et al., 2003). The LB encoding exploits this problem-specific knowledge in problem-independent metaheuristics. Experiments determine an appropriate value of the encoding-specific parameter P_1 that controls the bias of the LB encoding towards MST-like trees. Results for a collection of MCST test instances as well as for randomly-generated problems show high performance for different types of metaheuristics when using the LB encoding with an appropriate setting of P_1. The work demonstrates how problem-independent metaheuristics can be improved by combining them with problem-specific representations.

The next section describes the MCST problem and its properties. Section 3 introduces the LB encoding. Section 4 gives details on the metaheuristics used in the experiments and Sect. 5 presents experimental results. The experiments determine a proper setting for the encoding-specific parameter P_1 and examine the performance of the LB encoding for various test problems.

2. Minimum Communication Spanning Tree Problem

Let $G = (V, E)$ be a connected, undirected graph with $n = |V|$ nodes and $m = |E|$ edges. There are communication demands between the n different nodes. An $n \times n$ *demand matrix* $R = (r_{ij})$ specifies the amount of traffic required between the nodes. An $n \times n$ *distance weight matrix* $W = (w_{ij})$ specifies the distance weights associated with each pair of nodes. The weight $w(T)$ of the spanning tree $T = (V, F)$, where $F \subseteq E$, is the weighted sum over all pairs of vertices of the cost of the path between all pairs in T:

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

where the $n \times n$ matrix $B = b_{ij}$ denotes the traffic flowing directly and indirectly over the edge between the nodes i and j. The traffic 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'. The OCST problem becomes the MST problem if $w(T) = \sum_{i,j \in F} w_{ij}$. The distance between two trees T_i and T_j is the number of edges in which they differ; that is, $d_{ij} = \frac{1}{2} \sum_{u,v \in V} |e^i_{uv} - e^j_{uv}|$, where e^i_{uv} is 1 if edge (u, v) exists in T_i and 0 if it does not exist in T_i.

The MCST problem is $\mathcal{NP}$-hard (Garey and Johnson, 1979, p. 207). Furthermore, Reshef (1999) showed that the problem is $\mathcal{MAX}$ $\mathcal{SNP}$-hard (Papadimitriou and Yannakakis, 1991). The current best approximation algorithm for the OCST problem with Euclidean distance weights finds a tree with cost $w(T) = O(\log n) \cdot w(G)$ (Peleg and Reshef, 1998; Reshef, 1999), where $w(G) = \sum_{i,j \in E} r_{ij} t^G_{ij}$ is the volume of communication in the complete graph G and t^G_{ij} is the sum of all weights along the shortest path between i and j in G. $w(G)$ is a trivial lower bound for $w(T)$ as the weight of a path between node i and j in a spanning tree T is greater than or equal to the weight of the path in G.

Many researchers have applied search heuristics to MCST problems (Palmer, 1994; Berry et al., 1995; Li and Bouchebaba, 1999; Picciotto, 1999; Kim and Gen, 1999; Chou et al., 2001; Rothlauf, 2006b; Soak et al., 2006) and found the problem representation to be important for the success of metaheuristics. Prüfer numbers (Prüfer, 1918), characteristic vectors, NetKeys (Rothlauf et al., 2002), or edge-sets (Raidl and Julstrom, 2003) are commonly used tree representations. Prüfer's mapping is a one-to-one correspondence between trees and strings of length $n - 2$ over an alphabet of n symbols. Therefore, it is possible to derive a unique tree with n nodes from a Prüfer number of length $n - 2$ and vice versa. The construction and deconstruction process can be done in time that is $O(n \log(n))$ and $O(n)$, respectively (Caminiti et al., 2004). Prüfer numbers are not an efficient representation (Gottlieb et al., 2001) but are commonly considered as the worst-case scenario. The characteristic vector (CV) encoding uses a binary vector of length $n(n-1)/2$ that indicates whether an edge is in the tree. Every CV must have exactly $n - 1$ ones and the represented graph must be connected and acyclic to form a tree. Therefore, to ensure that random CVs always encode trees, repair mechanisms are necessary. The most common approach (see e.g. Berry et al. (1995)) initially removes randomly all links that cause cycles in the represented graph, and then randomly adds feasible links to connect the disconnected subtrees. Weighted encodings use continuous values for the encoding of a tree. From the weights, a construction algorithm creates a tree. Common weighted encodings are NetKeys (Rothlauf et al., 2002)

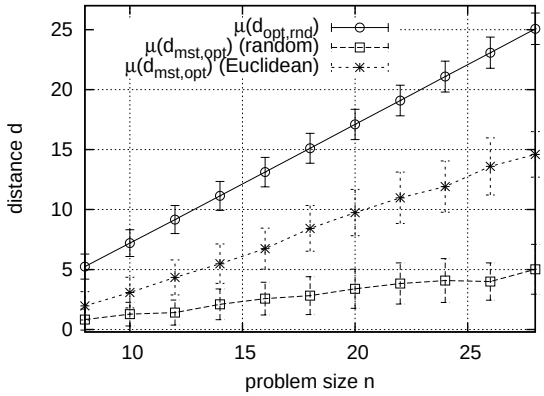


Figure 1: The plots show for random MCST problems with random or Euclidean distance weights how $\mu(d_{opt,rnd})$ and $\mu(d_{mst,opt})$ depend on the problem size n. $\mu(d_{mst,opt}) < \mu(d_{opt,rnd})$, so optimal solutions are biased towards MSTs.

or the weighted encoding from Raidl and Julstrom (2000). NetKeys represent a tree as a continuous vector r of length $n(n-1)/2$. The $r_{ij} \in [0,1]$ are interpreted as distance weights w'_{ij} ($w'_{ij} = r_{ij}$) and the represented tree is constructed from the w'_{ij} by Kruskal's algorithm. The construction of the tree from the r_{ij} can be done in time that is $O(n^2)$ (Prim's algorithm) or $O(n^2 \log(n))$ (Kruskal's algorithm). Edge-sets (Raidl and Julstrom, 2003) directly encode spanning trees by listing their edges. Search operators for edge-sets may be heuristic, considering the weights of edges, or naive, inserting and removing edges without regard to their weights. Crossover and mutation operators that heuristically prefer low-weight edges are strongly biased towards MSTs (Tzschoppe et al., 2004). Therefore, metaheuristics that apply heuristic search operators generally perform poorly on spanning tree problems whose optimal solutions are different from MSTs.

Rothlauf et al. (2003) performed a statistical analysis on the properties of optimal solutions T_{opt} for randomly generated MCST problems using random demands r_{ij} and either Euclidean (on a two-dimensional grid) or random distance weights w_{ij}. Figure 1 summarizes the results and compares the average distances $\mu(d_{opt,rnd})$ of optimal solutions from random trees to the average distances $\mu(d_{mst,opt})$ of optimal solutions from MSTs. The error bars indicate the standard deviations for 100 randomly chosen problem instances of each size n. The results show that optimal solutions are similar to MSTs.

3. The Link-biased Encoding

Indirect representations (also known as decoders) are a promising possibility of incorporating problem-specific heuristics into metaheuristics (Holland, 1975; Liepins and Vose, 1990; Storer et al., 1992). Such representations define an explicit, often problem-specific, genotype-phenotype mapping that assigns genotypes (decoder's space) to phenotypes (original space). The LB encoding, which is a specialized version of the link-and-node biased encoding (Palmer, 1994), is an indirect representation for trees that can exploit the problem-specific knowledge about MCST problems, which is that optimal solutions are similar to

MSTs. It encodes a tree using bias weights b_{ij} for all edges (i, j) and modifies the distance weights w_{ij} according to the b_{ij}. A tree construction algorithm like Kruskal's or Prim's constructs the encoded tree from the modified distance weights resulting in time that is $O(n^2)$ or $O(n^2 \log(n))$, respectively. The bias weights $b_{ij} \in [0, 1]$ modify the original distance weights w_{ij}:

$$w'_{ij} = w_{ij} + P_1 b_{ij} w_{\max}, \tag{1}$$

where w'_{ij} are the modified distance weights, $w_{\max} = \max(w_{ij})$, and P_1 is a user-defined, link-specific bias. We present an example for the construction of a tree from a bias vector b. To represent a tree with $n = 4$ nodes, the genotype has length $l = n(n-1)/2 = 6$. The link-biased individual is $b = \{0.1, 0.6, 0.2, 0.1, 0.9, 0.3\}$. With $P_1 = 1$ and using the distance weights $w = \{10, 30, 20, 40, 10, 20\}$, the modified weights become $w' = \{14, 54, 28, 44, 46, 32\}$ ($w_{max} = 40$). Figure 2 shows the tree that is calculated as the MST using the modified edge weights w'. The six possible edges are labeled from 1 to 6, and the tree consists of edges (A, B) (edge 1 with $w'_{AB} = 14$), (A, D) (edge 3 with $w'_{AD} = 28$), and (C, D) (edge 6 with $w'_{CD} = 32$). The parameter P_1 controls the influence of the bias weights b_{ij} on w'_{ij} and has a large impact on the structure of the encoded tree.

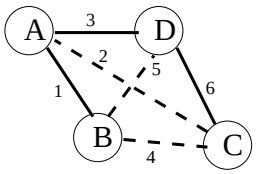


Figure 2: An example tree. The labels indicate edge numbers.

Since the LB encoding represents a finite number of trees with continuous genotypes, the LB encoding is a *redundant* representation; the number of genotypes exceeds the number of phenotypes. Radcliffe and Surry (1995) introduced a different notion of redundancy and distinguished between *degenerated* representations, where more than one genotype encodes one phenotype, and redundant representations where parts of the genotypes are not used for the construction of a phenotype. In this paper we follow the majority of the literature and define encodings to be redundant if the number of genotypes exceeds the number of phenotypes (Cohoon et al., 1988). Rothlauf and Goldberg (2003) showed that for $P_1 \to \infty$, the LB encoding becomes approximately *uniformly redundant* which 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 MSTs are preferred. Then, a randomly chosen LB-encoded genotype represents trees similar to an MST with higher probability. If $P_1 = 0$, only one tree, an MST with respect to the w_{ij}, can be represented since the b_{ij} have no effect on w'_{ij}.

How large must P_1 be set to allow the LB encoding to represent all possible trees? Kruskal's or Prim's algorithm use an ordered list of edges and iteratively insert edges with the lowest weight into the tree. The b_{ij} modify the original order of the edges and thus allow the LB encoding to represent other trees than the MST based on the original distance weights w_{ij}. If $P_1 \geq 1$, setting $b_{ij} = 1$ can result in a modified distance weight w'_{ij} which is equal to or greater than the largest original distance weight $\max(w_{ij})$. Therefore, a genotype can completely change the original ordered list of edges as for the edge (l, m), where $w_{lm} = \min(w_{ij})$ the modified distance weight becomes $w'_{lm} = \min(w_{ij}) + P_1 \max(w_{ij}) > \max(w_{ij})$ if $P_1 > 1$. As a result, for $P_1 > 1$ the LB encoding can encode all n^{n-2} different trees (assuming a proper setting of the b_{ij}).

Therefore, we make a recommendation for the choice of P_1. With lower P_1, the LB encoding represents trees similar to MSTs with higher probability. Since optimal solutions

for MCST problems are similar to MSTs, we expect better performance with low values of P_1. However, if $P_1 < 1$, not all possible trees can be represented. Therefore, we recommend setting $P_1 \approx 1$.

The idea of considering problem-specific knowledge for indirect representations is not new. Grefenstette et al. (1985) presented the ordinal representation as an early example of indirect and problem-specific representation for the TSP. The representation encodes a tour (permutation of n integers) by a genotype x^g of length n, where $x_i^g \in \{1, \ldots, n-i\}$ and $i \in \{0, \ldots, n-1\}$. For constructing a phenotype, a pre-defined permutation x^s of n integers is used. x^s can be problem-specific and for example consider edge weights. A phenotype (tour) is constructed from x^g by subsequently adding (starting with $i = 0$) the x_i^gth element of x^s to the phenotype (which initially contains no elements) and removing the x_i^gth element from x^s. Problem-specific knowledge can be considered by choosing an appropriate x^s because genotypes define perturbations of x^s and using small integers for the x_i^g results in a bias of the resulting phenotypes towards x^s. For example, for $x_i^g = 1$ ($i \in \{0, \ldots, n-1\}$), the resulting phenotype is x^s. Similarly, Storer et al. (1992) proposed problem space search (PSS) and heuristic space search (HSS) which combine problem-specific heuristics with problem-independent metaheuristics. PSS and HSS apply, in each search iteration of a metaheuristic, a fast, problem-specific base heuristic H which constructs a phenotype from a genotype. PSS applies H to perturbed versions of the genotype. The perturbations of the genotypes are usually small and based on a definition of neighborhood in the genotype space. For HSS, in each search step of the metaheuristic, the genotypes remain unchanged, but (slightly) different variants of the base heuristic H are used. Results presented for different applications show that this approach can lead to improved performance of metaheuristics (Storer et al., 1995; Naphade et al., 1997; Ernst et al., 1999).

4. Metaheuristics for the MCST problem

We give details on the different types of metaheuristics which are used in the experiments as representatives for crossover-based search (genetic algorithm) and local search (simulated annealing).

Genetic Algorithm We use a standard generational GA (Goldberg, 1989) with uniform crossover, no mutation, and standard 2-tournament selection without replacement. A GA run is stopped after all solutions in the population encode the same phenotype, or after 100 generations. N denotes the number of solutions in a population. P_{suc} is the percentage of runs where the optimal solution was found and t_{conv} is the average number of generations after which the GA is stopped. For string representations, uniform crossover chooses the alleles for the offspring randomly from the two parents. Therefore, it requires time that is either $O(n)$ (Prüfer numbers) or $O(n^2)$ (other representations). The naive variant of edge-sets generates an offspring T_{off} from $T_1(V, F_1)$ and $T_2(V, F_2)$ by including all edges $(F_1 \cap F_2)$ in the offspring and applying KruskalRST (Raidl and Julstrom, 2003) to the remaining edges $(F_1 \cup F_2) \backslash (F_1 \cap F_2)$. This requires time that is $O(n \log(n))$. For the heuristic variant, KruskalRST is replaced by iterative binary tournaments that choose two random edges from $(F_1 \cup F_2) \backslash (F_1 \cap F_2)$ and insert the edge with lower cost into T_{off} (if no cycle is created).

Simulated Annealing One step of simulated annealing examines a solution that is a neighbor of the current solution. For the LB encoding or NetKeys, a step randomly chooses a bias weight b_{ij} and sets it to a random value $[0, 1]$. For the CV encoding, a step inverts a randomly chosen allele and repairs the resulting (infeasible) CV. In all SA runs, a new, neighboring solution T_n replaces the original solution T_o if $w(T_n) < w(T_o)$. Otherwise, T_n replaces T_o with probability $P(U) = \exp(\frac{w(T_o)-w(T_n)}{U})$. The temperature U follows a fixed, step-wise cooling schedule, where $U_i = 0.99 * U_{i-1}$. Each SA run is stopped if the number of steps exceeds $iter_{max}$ or there are no improvements in the last $iter_{term}$ steps. t_{iter} denotes the number of steps after which the SA terminates. P_{suc} is the percentage of runs where the optimal solution is found. We set the initial temperature U_0 with respect to the target problem instance. For each problem instance, we generated 1,000 random solutions T_i using unbiased Prüfer numbers and set the initial temperature to $U_0 = 2\sigma(w(T_i))$, where $\sigma(w(T_i))$ denotes the standard deviation of the cost of the 1,000 solutions T_i.

Initialization For representations that encode solutions as strings (CV, Prüfer numbers, NetKey, LB encoding), initial solutions are generated by assigning random values to the genotypic strings. This can be done in times that are $O(n)$ (Prüfer numbers) or $O(n^2)$ (other representations). For the naive variant of edge-sets, KruskalRST creates initial solutions in times that are $O(n^2)$. For the heuristic variant, Kruskal's algorithm with perturbed distance weights w_{ij} is used, so time is $O(n^2 \log(n))$. The encoding-specific initialization parameter $\alpha = 1.5$ (Raidl and Julstrom, 2003) controls the perturbation of the w_{ij}, resulting in a bias towards MST-like trees.

Tree Evaluation The set F of edges completely defines a solution $T(V, F)$. To evaluate a tree, we iteratively remove all edges $(i, j) \in F$ from T and, in each iteration obtain two unconnected subtrees $T_1(V_1, F_1)$ and $T_2(V_2, F_2)$. The traffic b_{ij} is the sum over all demands r_{lm}, where either $l \in V_1$ and $m \in V_2$ or $m \in V_1$ and $l \in V_2$. The current implementation uses a recursive search for determining whether two nodes l and m are connected and requires time that is $O(n^2 \log(n))$. In the experiments, the main computational effort is used for tree evaluation and the effort for generating new solutions by applying search operators or calculating the phenotype from the genotype is low.

Finding optimal solutions for the MCST Problem For finding optimal, or at least near-optimal solutions, we used a mathematical programming solver for problem instances with random w_{ij} and a GA for problem instances with Euclidean w_{ij}. Modeling the MCST problem as an integer linear program (Rothlauf, 2006a), CPLEX 10.2 is able to solve all problem instances with random w_{ij} and $n \leq 22$ in reasonable time. The situation is different for Euclidean w_{ij}, where CPLEX can solve problem instances only with $n < 12$ in reasonable time. Therefore, we used an iterative GA for all problems with Euclidean w_{ij}. Although GAs are heuristic search methods that cannot guarantee finding the optimal solution, we choose its design in such a way that we can assume that each solution is optimal or near-optimal.

For a fixed problem size n, the probability α that a GA fails to find an optimal solution is $O(\exp(-N))$ (Harik et al., 1997). Therefore, GA performance improves with N. Consequently, we apply a GA n_{iter} times to an MCST problem 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 apply again a GA n_{iter} times with $N_1 = 2N_0$ which finds the best solution T_1^{best}. We continue the iterations 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. For the experiments, we use a standard, generational GA with crossover and mutation. Problems are encoded using NetKeys, since GA performance is approximately independent of the structure of the optimal solution. The GA uses uniform crossover and tournament selection without replacement. The size of the tournament is three. The crossover probability is $p_{cross} = 0.7$ and the mutation probability (assigning a random value $[0, 1]$ to one allele) is $p_{mut} = 1/l$, where $l = n(n-1)2$.

5. Experimental Results

This section presents experimental results for the LB encoding. Section 5.1 studies the performance of GAs using the LB encoding for existing test instances from the literature. Because the setting of P_1 is crucial for the performance of metaheuristics using the LB encoding, Sect. 5.2 studies how the value of P_1 affects GA and SA performance. Finally, we examine metaheuristic's performance for small problem instances where the optimal solution is determined as described in the previous section (Sect. 5.3), as well as for larger problem instances up to 100 nodes (Sect. 5.4). The goal of the experiments is not to repeat existing comparisons of tree representations (Raidl and Julstrom, 2003; Rothlauf, 2006b) but to study how an appropriate value for P_1 can exploit problem-specific knowledge in problem-independent metaheuristics.

5.1. GA Performance for Existing Problem Instances

Test instances for the MCST problem have been proposed by Palmer (1994), Berry et al. (1995), and Raidl (2001). For details, we refer to Rothlauf (2006b, Sect. 8.2). Table 1 lists properties of the test instances. It shows the number n of nodes, the cost $w(T_{opt})$ of an optimal (or best known) solution T_{opt}, the mean distance $\mu(d_{opt,rnd})$ between an optimal solution and randomly created trees T_{rnd}, and the distance $d_{mst,opt}$ between an MST and T_{opt}. For $\mu(d_{opt,rnd})$, 10,000 random trees have been generated for each problem. The results confirm the results from Rothlauf et al. (2003) that $d_{mst,opt}$ is significantly lower than $\mu(d_{opt,rnd})$ and therefore optimal solutions are similar to MSTs.

Table 2 compares the performance of the GA from Sect. 4 using CVs, Prüfer numbers, NetKeys, edge-sets with either naive or heuristic initialization and crossover operators, and the LB encoding with different values of P_1. For different representations, we present P_{suc} for a GA that runs 200 times for each of the test instances with population size N. Best values are printed bold.

The results indicate that P_{suc} is high for the LB encoding with $P_1 \approx 1$. For some problem instances, a higher (palmer24) or lower (palmer6, palmer12, and raidl20) value of P_1 can slightly improve GA performance, but choosing $P_1 \approx 1$ results on average in robust GA performance. As expected, GA performance is low for small values of P_1 as only MSTs can be encoded (except for berry6 and berry35 where optimal solutions are MSTs). For high

Table 1: Properties of test problems from the literature

problem	n	$w(T_{opt})$	$\mu(d_{opt,rnd})$	$d_{mst,opt}$
palmer6	6	693,180	3.40	1
palmer12	12	3,428,509	9.05	5
palmer24	24	1,086,656	21.07	12
raidl10	10	53,674	7.08	3
raidl20	20	157,570	17.10	4
berry6	6	534	3.51	0
berry35	35	16,915	32.05	0

Table 2: GA success probabilities P_{suc} using different representations. The results indicate high and robust GA performance when using the LB encoding with $P_1 \approx 1$.

				Net	edge-sets		LB				
problem	N	Prüfer	CV	Key	naive	heur.	P_1=0.05	P_1=0.5	P_1=1	P_1=2	P_1=100
palmer6	16	0.05	0.34	0.27	0.17	0	0.31	**0.84**	0.66	0.48	0.28
palmer12	300	0	0	0.31	0.31	0	0	**0.74**	0.62	0.48	0.34
palmer24	800	0	0	0.72	0.08	0	0	0.03	0.49	0.64	**0.77**
raidl10	70	0.07	0.80	0.78	0.44	0	0	**1**	**1**	0.98	0.78
raidl20	450	0	0	0.46	0.53	0	0	**0.99**	0.92	0.85	0.41
berry6	16	0.02	0.74	0.54	0.38	**1**	**1**	**1**	0.98	0.82	0.56
berry35	300	0	0	0.03	0	**1**	**1**	**1**	**1**	0.98	0.04

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 with NetKeys.

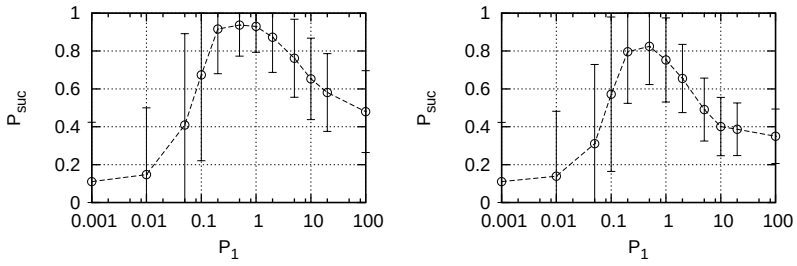
GAs using Prüfer numbers and CV encoding perform as expected (Rothlauf, 2006b). CVs show good results only for small problem instances ($n \leq 10$), and Prüfer numbers fail for all problems. Heuristic edge-sets fail if the optimal solution is only slightly different from the MST. When using the naive variant, the GA shows higher performance but is on average outperformed by the NetKey encoding (Tzschoppe et al., 2004).

5.2. Influence of P_1 on Performance of Metaheuristics

On random problem instances, the performance of metaheuristics using the LB encoding depends on P_1. The r_{ij} are uniformly distributed in $[0, 10]$ and the w_{ij} are either uniformly distributed in $(0, 100]$ (random w_{ij}), or are the Euclidean distances between nodes placed randomly on a 2-dimensional grid of size 1,000×1,000 (Euclidean w_{ij}).

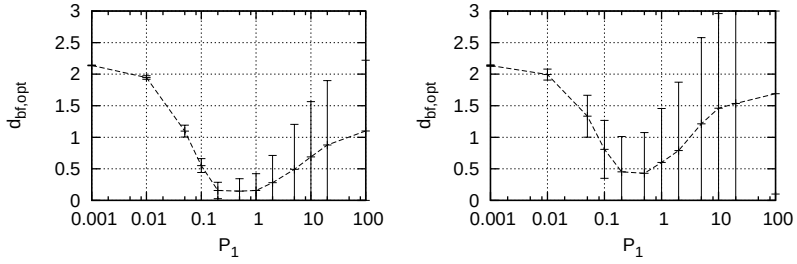
Figure 3 shows the success probability P_{suc} (mean and standard deviation) of metaheuristics over the link-specific bias P_1 for 100 randomly created MCST problem instances with $n = 16$ and random w_{ij}. For each problem instance, we performed 50 independent runs and averaged P_{suc} over all 100 problem instances. We show results for a GA with $N = 200$ (Fig. 3(a)) and SA (Fig. 3(b)). The SA runs are stopped either after $iter_{max} = 20,000$ search steps or if there is no improvement in the last $iter_{term} = 2,000$ search steps.

GA and SA performance is best when using the LB encoding with $0.2 \leq P_1 \leq 1$. For larger values of P_1, the LB encoding becomes uniformly redundant since all possible trees are represented with the same probability. Therefore, the performance of metaheuristics using the LB encoding with $P_1 \to \infty$ is similar to that with the unbiased NetKey encoding,



(a) GA (b) SA

Figure 3: Mean success probability P_{suc} of GA and SA for randomly created 16 node MCST problems over P_1. Standard deviations are indicated as error bars.



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

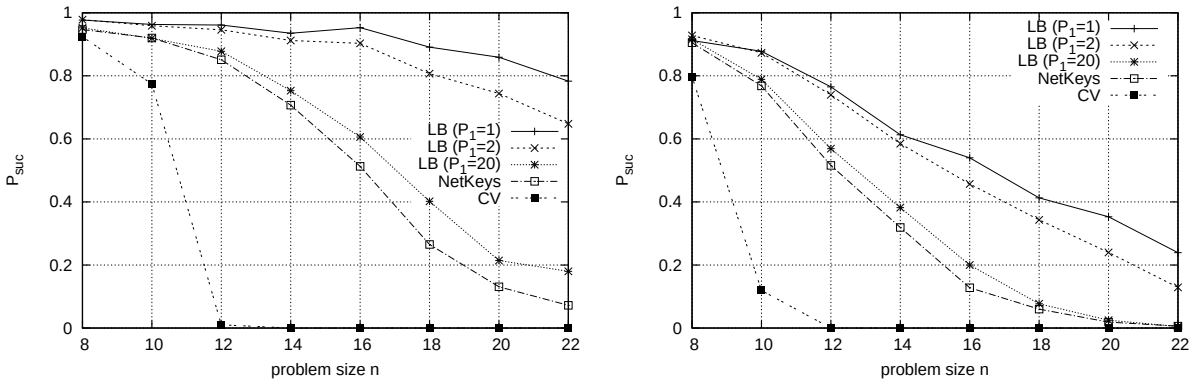
Figure 4: Mean distance $d_{bf,opt}$ between the best found solution T_{bf} and the optimal solution T_{opt} over P_1 for random MCST problems. Standard deviations are indicated as error bars.

which is $P_{suc} = 0.49$ (GA) and $P_{suc} = 0.39$ (SA). For low values of P_1, P_{suc} becomes small as MCST problems can be solved only where the optimal solution is an MST. A pairwise t-test is performed on the success probabilities P_{suc} of GA and SA; for $0.1 \leq P_1 \leq 5$, the LB encoding outperforms NetKeys as well as LB with $P_1 = 100$ with high significance ($p < 0.001$).

To understand better how metaheuristic's performance depends on P_1, Fig. 4 shows the distance $d_{bf,opt}$ (mean and standard deviation) between the best solution T_{bf} that has been found by GA or SA and the optimal solution T_{opt} for all 100 randomly generated 16 node MCST problems. Because optimal solutions are similar to MSTs and the LB encoding encodes solutions similar to MSTs with higher probability, $d_{bf,opt}$ decreases with lower P_1. However, if P_1 is too small ($P_1 < 1$), not all possible solutions can be represented and $d_{bf,opt}$ increases again. For $P_1 \to 0$ only MSTs can be encoded and $d_{bf,opt}$ becomes the average distance $\mu(d_{mst,opt})$ between MSTs and optimal solutions (compare Fig. 1).

5.3. Performance of LB Encoding for Small MCST Problems

Figure 5 shows how the mean success probability P_{suc} of a GA or SA using different representations depends on the problem size n for random MCST problems with either random or Euclidean w_{ij}. We set the parameters for the GA to $N = 200$ and for the SA to $iter_{max} = 20,000$ and $iter_{term} = 2,000$. As before, we generated 100 random problem instances for each problem size n and performed 50 runs for each problem.



(a) GA, random w_{ij}

(b) GA, Euclidean w_{ij}

(c) SA, random w_{ij}

(d) SA, Euclidean w_{ij}

Figure 5: Mean success probability P_{suc} of GA and SA with fixed parameters over n for random MCST problems. The w_{ij} are either random (left), or Euclidean (right).

The plots show similar results for GA and SA as well as for MCST problems with random and Euclidean distance weights. As expected, the performance of representations that prefer MST-like solutions is higher for MCST problems with random w_{ij} than with Euclidean w_{ij} (compare Fig. 1). GA and SA performance decreases with increasing n as the parameters of the metaheuristics are fixed and the larger problem instances are more difficult. To increase metaheuristics performance, either larger population sizes N (GA), or a different cooling schedule combined with a higher number of search steps (SA) would be necessary. Most importantly, GA and SA using the LB encoding with $P_1 = 1$ outperform other "non-problem-specific" representations such as NetKeys and CVs. We omit results for Prüfer numbers and edge-sets due to the poor performance of these encodings. Consequently, we test the hypothesis H_0 that the performance of the LB encoding with $P_1 = 1$ does not outperform NetKeys using a pairwise t-test. For $n \leq 12$ there is not always a significant difference between NetKeys and the LB encoding with $P_1 = 1$. However, for larger problem instances, the differences are significant ($p < 0.0001$), thereby rejecting the null hypothesis H_0 and supporting the alternative that the LB encoding with $P_1 = 1$ outperforms NetKeys.

Figure 6 studies running times and shows the average number t_{conv} of GA generations over n for the test instances with random w_{ij}. The results for SA and Euclidean w_{ij} are analogous. As expected, t_{conv} increases with n (Mühlenbein and Schlierkamp-Voosen, 1993). GAs using the LB encoding with $P_1 = 1$ need the lowest number of generations in comparison

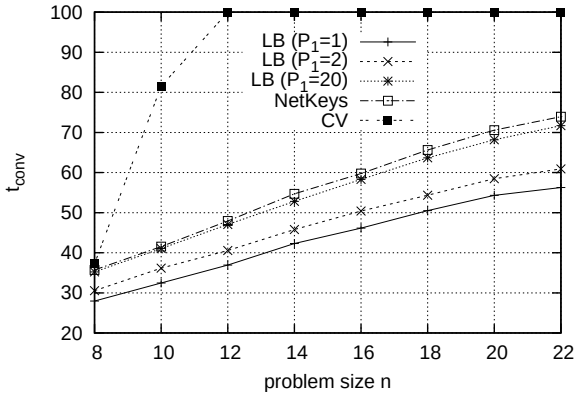


Figure 6: Average number t_{conv} of generations over n for GAs and random w_{ij}.

with the LB encoding with higher values of P_1, NetKeys, and CVs.

5.4. Performance of LB encoding for Large MCST Problems

We study the performance of metaheuristics using the LB encoding for larger random MCST problems of different type and size and compare it with NetKeys and approximation algorithms from Peleg and Reshef (1998) and Reshef (1999) (denoted as PeRe). They presented an $O(\log n)$ approximation for problems with Euclidean w_{ij} and an $O(\log^3 n)$ approximation for arbitrary (non-Euclidean) w_{ij}. The approximations use a recursive construction algorithm with a partitioning step which clusters the nodes in disjoint subsets and are the best currently available approximation algorithms for MCST problems.

Table 4 shows the average cost $w(T_{bf})$ of the best solution found, and the average number of evaluations *eval* for randomly created problems with 10, 20, 50, and 100 nodes. For each n, we generated 100 random problem instances and performed 20 runs for each problem instance (for $n = 100$, we only created 15 random problem instances). The w_{ij} are either random or Euclidean. In addition, the table lists P_{suc} for small problem instances ($n = 10$ and $n = 20$) where optimal solutions are determined as described in Sect. 4.

	n	10	20	50	100
GA	N	100	200	400	800
SA	$iter_{max}$	10,000	20,000	40,000	80,000
	$iter_{term}$	2,000	4,000	8,000	16,000

Table 3: GA and SA parameter

Because the performance of metaheuristics depends on parameter settings and metaheuristics with fixed parameters show lower performance with increasing n (compare Fig. 5), the GA and SA parameter must be changed. Table 3 shows how GA and SA parameters (N resp. $iter_{max}$ and $iter_{term}$) depend on n. By using higher population sizes (GA), or a larger number of search steps (SA), solutions with higher quality can be found.

GA and SA using the LB encoding with $P_1 = 1$ find better solutions than NetKeys, especially with increasing n. The approximation algorithm never finds optimal or near-optimal solutions and the solution quality is much lower than that of GAs using NetKeys or the LB encoding. We have performed a pairwise t-test on the cost $w(T_{bf})$ of GA and SA and find for $n > 10$ that the LB encoding with $P_1 = 1$ produces better solutions than NetKeys with high significance ($p < 0.001$).

Table 4: Performance of GA, SA, and approximation algorithm (PeRe) for large random MCST problems with either Euclidean or random w_{ij}. Best values are printed bold.

w_{ij}	n		GA				SA				PeRe
			NetKey	LB P_1=0.2	LB P_1=1	LB P_1=5	NetKey	LB P_1=0.2	LB P_1=1	LB P_1=5	
Euclidean	10	P_{suc}	0.56	0.31	**0.78**	0.60	0.71	0.29	**0.75**	0.72	0
		$w(T_{bf})$	153,583	155,600	**152,717**	153,334	153,620	155,533	**153,250**	153,592	210,540
		eval	4,071	2,565	3,289	3,868	3,452	2,968	3,041	3,510	-
	20	P_{suc}	0.02	0.06	**0.28**	0.12	0.10	0.06	**0.30**	0.19	0
		$w(T_{bf})$	690,697	692,340	**677,477**	682,193	686,866	694,132	**680,282**	684,002	$1.1*10^6$
		eval	15,207	10,438	12,690	14,292	14,951	8,637	9,545	13,387	-
	50	$w(T_{bf})$	$4.78*10^6$	$4.56*10^6$	$\mathbf{4.47*10^6}$	$4.51*10^6$	$5.03*10^6$	$4.60*10^6$	$\mathbf{4.56*10^6}$	$4.66*10^6$	$9.7*10^6$
		eval	40,000	40,000	40,000	40,000	40,000	37,261	39,520	39,982	0
	100	$w(T_{bf})$	$2.21*10^7$	$1.85*10^7$	$\mathbf{1.83*10^7}$	$1.89*10^7$	$2.43*10^7$	$1.85*10^7$	$\mathbf{1.847*10^7}$	$1.96*10^7$	$5.34*10^7$
		eval	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	-
random	10	P_{suc}	0.76	0.89	**0.93**	0.86	0.86	0.88	**0.90**	0.87	0
		$w(T_{bf})$	6,844	6,802	**6,793**	6,814	6,821	6,807	**6,799**	6,835	19,133
		eval	3,634	1,844	2,733	3,340	3,441	2,286	2,697	3,359	-
	20	P_{suc}	0.18	**0.92**	0.84	0.55	0.38	**0.90**	0.80	0.50	0
		$w(T_{bf})$	19,825	**18,599**	18,612	18,777	19,349	**18,607**	18,687	19,097	90,681
		eval	14,222	8,258	10,914	12,597	15,542	6,188	9,017	13,240	-
	50	$w(T_{bf})$	108,465	**65,256**	65,610	69,213	148,263	**65,823**	70,674	90,417	592,808
		eval	40,000	36,612	40,000	40,000	40,000	32,015	39,386	40,000	-
	100	$w(T_{bf})$	$7.7*10^5$	$\mathbf{1.6*10^5}$	$1.8*10^5$	$2.3*10^5$	$1.2*10^6$	$\mathbf{1.8*10^5}$	$2.4*10^5$	$4.1*10^5$	$2.4*10^6$
		eval	80,000	80,000	80,000	80,000	80,000	80,000	80,000	80,000	-

Comparing Euclidean and random w_{ij} shows that for Euclidean w_{ij}, GA and SA performance is maximal for $P_1 = 1$. For MCST problems with random w_{ij}, SA and GA performance can be slightly improved by lowering P_1. This improvement is due to the fact that for random w_{ij}, optimal solutions are more similar to MSTs than for Euclidean w_{ij} (compare Fig. 1). Therefore, metaheuristics that use a representation with a higher bias towards MSTs (like the LB encoding with $P_1 = 0.2$) show higher performance than the LB encoding with $P_1 = 1$. However, for $P_1 < 1$, the LB encoding cannot encode all possible trees.

Randomly generated MCST problem instances with randomly chosen demands r_{ij} and either random or Euclidean w_{ij} can be solved effectively when using the LB encoding with $P_1 = 1$. This setting of P_1 results in robust behavior of metaheuristics and allows them to exploit the fact that MCST solutions resemble MSTs.

6. Conclusions

Metaheuristics such as genetic algorithms or simulated annealing describe general optimization principles that can be applied to tree optimization problems such as the minimum communication spanning tree (MCST) problem. The MCST problem finds a spanning tree that connects all nodes and satisfies their communication requirements with minimum total cost. The paper studies a representation, the LB encoding, that encodes trees similar

to MSTs with a higher probability than random trees. The LB encoding exploits the fact that MCSTs are similar to MSTs and thus incorporates problem-specific knowledge about MCSTs in problem-independent metaheuristics.

Because the LB encoding can capture knowledge regarding MCSTs, its proper use improves metaheuristics performance. Local (simulated annealing) as well as recombination-based search methods (genetic algorithms) using the LB encoding outperform other, non-problem-specific representations for existing problem instances from the literature as well as for randomly created MCST problem instances. A proper setting of value of the LB-specific parameter P_1 is important as it 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 ($P_1 \to 0$), MSTs alone can be encoded and heuristic search is no longer possible.

The LB encoding creates a solution by applying an MST algorithm to modified distance weights defined by the genotype (problem). The approximation algorithm from Peleg and Reshef (1998) uses a recursive construction algorithm with a partitioning step that clusters the nodes and creates a solution from a genotype (set of distance weights). A promising line of research is to replace the MST algorithm used in the LB encoding by the construction algorithm from Peleg and Reshef. The resulting hybrid approach can exploit problem-specific properties of MCSTs and be used with many standard metaheuristics.

Acknowledgements

The author thanks the two referees and the Area Editor for their comments and suggestions that have led to an improved version of the paper.

References

Berry, L. T. M., B. A. Murtagh, G. McMahon. 1995. Applications of a genetic-based algorithm for optimal design of tree-structured communication networks. *Proc. of the Regional Teletraffic Engrg. Conf. of the Internat. Teletraffic Congress*. Telkom South Africa, Pretoria, South Africa, 361–370.

Caminiti, Saverio, Irene Finocchi, Rossella Petreschi. 2004. A unified approach to coding labeled trees. Martin Farach-Colton, ed., *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium*, *LNCS*, vol. 2976. Springer, 339–348.

Chou, H., G. Premkumar, Chao-Hsien Chu. 2001. Genetic algorithms for communications network design - an empirical study of the factors that influence performance. *IEEE Trans. on Evolutionary Comput.* **5** 236–249.

Cohoon, J., S. Hegde, W. Martin, D. Richards. 1988. Floorplan design using distributed genetic algorithms. *IEEE Internat. Conf. on Comp. Aided-Design*. IEEE, 452–455.

Ernst, Krishnamoorthy, Storer. 1999. Heuristic and exact algorithms for scheduling aircraft landings. *Networks: An Internat. J.* **34** 229–241.

Garey, Michael R., David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

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

Gottlieb, Jens, Bryant A. Julstrom, Günther R. Raidl, Franz Rothlauf. 2001. Prüfer numbers: A poor representation of spanning trees for evolutionary search. IlliGAL Report No. 2001001, Univ. of Illinois at Urbana-Champaign, Urbana.

Grefenstette, J. J., R. Gopal, B. J. Rosmaita, D. Van Gucht. 1985. Genetic algorithms for the traveling salesman problem. J. J. Grefenstette, ed., *Proc. of an Internat. Conf. on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ, 160–168.

Harik, G. R., E. Cantú-Paz, D. E. Goldberg, B. L. Miller. 1997. The gambler's ruin problem, genetic algorithms, and the sizing of populations. T. Bäck, ed., *Proc. of the Forth Internat. Conf. on Evolutionary Computation*. IEEE Press, New York, 7–12.

Holland, J. H. 1975. *Adaptation in natural and artificial systems*. Univ. of Michigan Press, Ann Arbor, MI.

Hu, T. C. 1974. Optimum communication spanning trees. *SIAM J. on Computing* **3** 188–195.

Kim, J. R., M. Gen. 1999. Genetic algorithm for solving bicriteria network topology design problem. Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, Ali Zalzala, William Porto, eds., *Proc. of the 1999 IEEE Congress on Evolutionary Computation*. IEEE Press, 2272–2279.

Li, Y., Y. Bouchebaba. 1999. A new genetic algorithm for the optimal communication spanning tree problem. C. Fonlupt, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, eds., *Proc. of Artificial Evolution: Fifth European Conf.*. Springer, Berlin, 162–173.

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

Mühlenbein, H., D. Schlierkamp-Voosen. 1993. Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation* **1** 25–49.

Naphade, Kedar S., S. David Wu, Robert H. Storer. 1997. Problem space search algorithms for resource-constrained project scheduling. *Annals of Operations Research* **70** 307–326.

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

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

Peleg, David, Eilon Reshef. 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. Ph.D. thesis, Univ. of California, San Diego, USA.

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

Radcliffe, Nicholas J., Patrick D. Surry. 1995. Fitness variance of formae and performance prediction. L. Darrell Whitley, Michael D. Vose, eds., *Foundations of Genetic Algorithms 3*. Morgan Kaufmann Publishers, San Mateo, CA, 51–72.

Raidl, G. R., B. A. Julstrom. 2003. Edge-sets: An effective evolutionary coding of spanning trees. *IEEE Trans. on Evolutionary Comput.* **7** 225–239.

Raidl, Günther R. 2001. Various instances of optimal communication spanning tree problems. personal communciation.

Raidl, Günther R., Bryant A. Julstrom. 2000. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. Janice Carroll, Ernesto Damiani, Hisham Haddad, Dave Oppenheim, eds., *Proc. of the 2000 ACM Symposium on Applied Computing*. ACM Press, 440–445.

Reshef, E. 1999. 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. 2006a. *Design of Metaheuristics*. Habilitationsschrift. University of Mannheim.

Rothlauf, F. 2006b. *Representations for Genetic and Evolutionary Algorithms*. 2nd ed. Springer, Heidelberg.

Rothlauf, F., J. Gerstacker, A. Heinzl. 2003. On the optimal communication spanning tree problem. Tech. Rep. 15/2003, Department of Information Systems, Univ. of Mannheim.

Rothlauf, F., D. E. Goldberg. 2003. Redundant representations in evolutionary computation. *Evolutionary Comput.* **11** 381–415.

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

Soak, S.M., D.W. Corne, B.H. Ahn. 2006. On a property analysis of representations for spanning tree problems. *EA 2005*. No. 3871 in LNCS, Springer, 107–118.

Storer, Robert H., S. David Wu, Renzo Vaccari. 1992. New search spaces for sequencing problems with application to job shop scheduling. *Management Science* **38** 1495–1509.

Storer, Robert H., S. David Wu, Renzo Vaccari. 1995. Problem and heuristic space search strategies for job shop scheduling. *ORSA J. on Computing* **7** 453–467.

Tzschoppe, C., F. Rothlauf, H.-J. Pesch. 2004. The edge-set encoding revisited: On the bias of a direct representation for trees. Deb, Kalyanmoy et al., ed., *Proc. of the Genetic and Evolutionary Computation Conference 2004*. Springer, Heidelberg, 1174–1185.