On the Bias and Performance of the Edge-Set Encoding

Franz Rothlauf and Carsten Tzschoppe

Working Paper 11/2004
October 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`

On the Bias and Performance of the Edge-Set Encoding

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

Carsten Tzschoppe
Dept. of Business Administration and Information Systems
University of Mannheim
D-68131 Mannheim/Germany
carsten.tzschoppe@gmx.de

October 8, 2004

Abstract

The edge-set encoding is a direct encoding for trees which directly represents trees as sets of edges. In contrast to indirect representations, where usually standard operators are applied to a list of strings and the resulting phenotype is constructed by an appropriate genotype-phenotype mapping, encoding-specific initialization, crossover, and mutation operators have been developed for the edge-set encoding, which are directly applied to trees. There are two different variants of operators: heuristic versions that consider the weights of the edges and non-heuristic versions. An investigation into the bias of the different variants of the operators shows that the heuristic variants are biased towards the minimum spanning tree (MST), that means solutions similar to the MST are favored. In contrast, non-heuristic versions are unbiased. The performance of edge-sets is investigated for the optimal communication spanning tree (OCST) problem. Results are presented for randomly created problems as well as for test instances from the literature. Although optimal solutions for the OCST problem are similar to the MST, evolutionary algorithms using the heuristic crossover operator fail if the optimal solution is only slightly different from the MST. The non-heuristic version shows similar performance as the network random key encoding, which is an unbiased indirect encoding and is used as a benchmark. With proper parameter setting the heuristic version of the mutation operator shows good results for the OCST problem as it can make use of the fact that optimal solutions of the OCST problem are similar to the MST. The results suggest that the heuristic crossover operator of the edge-set encoding should not be used for tree problems as its bias towards the MST is too strong.

1 Introduction

A spanning tree $T(V, E)$ is a connected graph with $n = |V|$ vertices and $|E| = n - 1$ edges. T contains no cycles. Evolutionary algorithms (EAs) have successfully been applied to a large variety of tree problems like the degree-constrained minimum spanning tree (MST) problem [4, 12, 18], or the optimal communication spanning tree (OCST) problem [2, 8, 13].

When using EAs for tree problems it is necessary to encode a solution (tree) such that evolutionary search operators like crossover or mutation can be applied. There are two different possibilities for doing this: indirect representations usually encode a tree (phenotype) as a list of strings (genotypes) and apply standard search operators to the genotypes. The phenotype is constructed by an appropriate genotype-phenotype mapping (representation). There are many indirect mappings for trees like NetKeys [23], the link-and-node-biased encoding [14], determinant factorization [1], or Prüfer numbers [6, 17]. In contrast, direct representations encode a tree as a set of edges and apply search operators directly to the set of edges. Therefore, no representation is necessary. Instead, tree-specific search operators must be developed, as standard search operators can no longer be used. Examples for direct encodings are the edge-set encoding [18] or the NetDir encoding [20, sec. 7.2]. [18] proposed different variants of the edge-set encoding: heuristic variants where the encoding-specific search operators consider the weights of the edges and non-heuristic versions. Results from applying the edge-set encoding to two sets of degree-constrained MST problem instances indicated the superiority of edge-sets, particularly when the operators implement edge-cost-based heuristics, to several other codings of spanning trees – the Blob Code, network random keys, and strings of weights [18, p. 238].

The purpose of this paper is to thoroughly investigate the bias of the edge-set encoding and to examine if the heuristic variants are superior when used for the OCST problem. A bias of a direct encoding means that the encoding-specific initialization, crossover, and mutation operators prefer a specific type of solution and push a population in this direction. As the heuristic variants of the edge-set encoding prefer edges with low cost, these variants are expected to show a bias towards the MST. In a second step, the performance of edge-sets is investigated for random instances of the OCST problem as well as test problems from the literature. In contrast to the degree-constraint MST used in [18], there are no additional constraints regarding the structure of solutions and all possible trees are feasible. As optimal solutions of the OCST problem are biased towards the MST [21], heuristic versions of the edge-set encoding are expected to show good performance.

The following section summarizes the functionality of the edge-set encoding with and without heuristics. Section 3 investigates the bias of the encoding and section 4 examines its influence on the performance of evolutionary search for the OCST problem. The paper ends with concluding remarks.

2 The Edge-Set Encoding

The edge-set encoding directly represents trees as sets of edges. Therefore, encoding-specific initialization, crossover, and mutation operators are necessary. The following sections summarize the functionality of the different variants with and without heuristics [18].

2.1 The Edge-Set Encoding without Heuristics

2.1.1 Initialization

The purpose of the initialization algorithms is to create an unbiased initial solution. [18] proposed and investigated three different initialization strategies: PrimRST, RandWalkRST, and KruskalRST. PrimRST overrepresents star-like trees and underrepresents trees similar to lists. RandWalkRST has an average running time of $O(n \log n)$, however, the worst-case running time is unbounded. Therefore, [18] recommended the use of the KruskalRST which is based on the algorithm from Kruskal. In contrast to Kruskals' algorithm, KruskalRST chooses edges (i, j) not according to their corresponding weights w_{ij} but randomly. KruskalRST has a small bias towards star-like trees (which is lower than the bias of PrimRST).

```
procedure KruskalRST($V, E$):
$T \leftarrow \emptyset$, $A \leftarrow E$;    //$E$ is the set of available edges
while $|T| < |V| - 1$ do
  choose an edge $\{(u, v)\} \in A$ at random;
  $A \leftarrow A - \{(u, v)\}$;
  if $u$ and $v$ are not yet connected in $T$ then
    $T \leftarrow T \cup \{(u, v)\}$;
return $T$.
```

2.1.2 Recombination

To obtain an offspring T_{off} from two parental trees T_1 and T_2 with the edge sets E_1 and E_2, KruskalRST is applied to the graph $G_{cr} = (V, E_1 \cup E_2)$. Instead of KruskalRST, in principle PrimRST and RandWalkRST can be also used. The crossover operator has high heritability as in the absence of constraints, only parental edges are used to create the offspring. Crossover becomes more complicated for constraint MST problems as it is possible that the RST algorithm can create no feasible tree from $G_{cr} = (V, E_1 \cup E_2)$. Then, additional edges have to be chosen randomly to complete an offspring.

[18] distinguished two different recombination operators: the variant previously described is denoted KruskalRST crossover. The second variant is denoted KruskalRST* crossover. When using this variant, in a first step all edges $(E_1 \cap E_2)$ are included in the offspring T_{off}. Then T_{off} is completed by applying KruskalRST to the remaining edges $(E_1 \cup E_2) \backslash (E_1 \cap E_2)$. Results from [18] indicate a better performance of KruskalRST* for the degree-constraint MST problem.

2.1.3 Mutation

The mutation operator randomly replaces one edge in the spanning tree. This replacement can be realized in two different ways. The first variant of the mutation operator randomly chooses one edge that is not present in T and includes it in T. Then, one edge from the cycle is randomly chosen and removed ("insertion before deletion"). The second variant first randomly deletes one edge from T and then connects the two disjoint connected components using a random edge not present in T ("deletion before insertion"). The running time is $O(n)$ if there are no additional constraints.

2.2 The Edge-Set Encoding with Heuristics

The following paragraphs describe how heuristics that rely on the weights w_{ij} can be included in the edge-set encoding. [18] introduced these variants of the edge-set encoding due to the assumption that in weighted tree optimization problems optimal solutions often prefer edges with low weights w_{ij}.

2.2.1 Heuristic Initialization

To favor low-weighted edges when generating the initial population, the algorithm KruskalRST starts by sorting all edges in the underlying graph according to their weights w_{ij} in ascending order. The first spanning tree is created by choosing the first edges in the ordered list. As these are the edges with lowest weights, the first generated spanning tree is a MST. Then, the k edges with lowest weights are permuted randomly and another spanning tree is created using the first edges in the list. The heuristic initialization results in a strong bias towards the MST. With increasing k, the bias of randomly created trees towards the MST is reduced. The number of edges, which are permuted increases according to

$$k = \alpha(i-1)n/N,$$

where N denotes the population size, i is the number of the tree that is actually generated ($i = 1, \dots, N$) and α, with $0 \leq \alpha \leq (n-1)/2$, is a parameter that controls the strength of the heuristic bias.

2.2.2 Heuristic Recombination

The heuristic recombination operator is a modified version of KruskalRST* crossover. Firstly, the operator transfers all edges $E_1 \cap E_2$ that exist in both parents T_1 and T_2 to the offspring. Then, the remaining edges are chosen randomly from $E' = (E_1 \cup E_2) \setminus (E_1 \cap E_2)$ using a tournament with replacement of size two. This means, the weights w_{ij} of two randomly chosen edges are compared and the edge with the lower weight is inserted into the offspring (if no cycle is created). If the underlying optimization problem is constrained, it is possible that the offspring has to be completed using edges not in E'.

2.2.3 Heuristic Mutation

The heuristic mutation operator is based on mutation by "insertion before deletion". In a pre-processing step, all edges in the underlying graph are sorted according to their weights in ascending order. Doing this, a rank is assigned to every edge. The rank one is assigned to the edge with the lowest weight. To favor low-weighted edges, the edge that is inserted by the heuristic mutation operator is not chosen randomly but according to its rank

$$R = \lfloor|\mathcal{N}(0, \beta n)|\rfloor \bmod m + 1,$$

where $\mathcal{N}(0, \beta n)$ is the normal distribution with mean 0 and standard deviation βn and $m = n(n-1)/2$. β is a parameter that controls the bias towards low-weighted edges. If a chosen edge already exists in T, the edge is discarded and the selection is repeated.

3 Investigating the Bias of the Edge-Set Encoding

A representation is unbiased if all possible phenotypes are encoded uniformly [20]. Consequently, a search operator is unbiased if it does not overrepresent specific solutions, and the application of the search operator alone does not modify the statistical properties of a population. An unbiased search operator allows a uniform, non-directed search through the search space. A biased representation resp. operator should only be used if it is known a priori that the optimal solution of the underlying optimization problem is similar to the overrepresented solutions [22]. In contrast, unbiased representations resp. operators should be used if no a priori problem-specific knowledge is available. Then, the probability of finding the optimal solution is independent of the structure of the optimal solution.

The following paragraphs investigate the bias of the edge-set encoding for randomly created trees with $n = 10$ and $n = 16$ nodes. To every edge (i, j) a non-negative weight w_{ij} is associated. Two possibilities for choosing the weights w_{ij} are considered:

- **Random weights:** The real-valued weights w_{ij} are generated randomly and are uniformly distributed in $]0, 100]$.
- **Euclidean weights:** The nodes are randomly placed on a 1000x1000 grid. The weight w_{ij} between node i and j is the Euclidean distance between the two nodes.

As the weights w_{ij} are randomly created and $w_{ij} \neq w_{kl}$, $\forall i \neq l, j \neq l$, we can assume that there is an unique MST for every problem instance. T is the MST if $c(T) \leq c(T')$ for all other spanning trees T', where $c(T) = \sum_{(i,j)\in T} w_{ij}$. The similarity between two spanning trees T_i and T_j can be measured using the distance $d_{ij} \in \{0, 1, \ldots, n-1\}$ as $d_{ij} = \frac{1}{2}\sum_{u,v\in V,\, u<v} |l^i_{uv} - l^j_{uv}|$, where l^i_{uv} is 1 if an edge from u to v exists in T_i and 0 if it does not exist in T_i.

3.1 Initialization

[18] examined the bias of different initialization methods and found KruskalRST to be slightly biased towards stars. As the bias is sufficiently small and due to its lower running time it is preferred in comparison to RandWalkRST and PrimRST, which shows a stronger bias towards stars.

Table 1 shows the average distances $d_{rand,MST}$ between the MST and randomly generated trees (the standard deviations are shown in brackets). For each problem instance (1000 of each type) we generated 10,000 random solutions using either an unbiased encoding (Prüfer numbers), KruskalRST (section 2.1.1), or the heuristic initialization (section 2.2.1). For the heuristic initialization α was set either to $\alpha = 1.5$ as recommended in [18] or to the maximum value $\alpha = (n-1)/2$, which results in the lowest bias. The results confirm that KruskalRST is not biased towards the MST. Furthermore, the heuristic versions show a strong bias towards the MST even when using a large value of α.

Table 1: Distances $d_{rand,MST}$ between random trees and MST

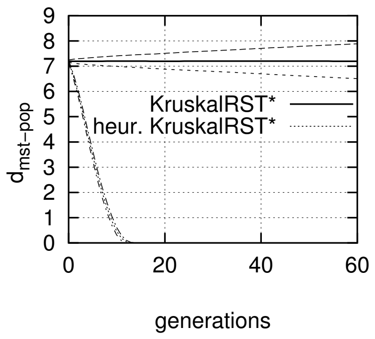
tree size	$n = 10$		$n = 16$	
weights	Euclidean	random	Euclidean	random
unbiased	7.20 (1.1)		13.12 (1.2)	
KruskalRST	7.20 (1.1)	7.20 (1.1)	13.13 (1.2)	13.13 (1.2)
heuristic ($\alpha = 1.5$)	1.06 (1.2)	0.84 (1.1)	1.87 (1.8)	1.39 (1.7)
heuristic ($\alpha = (n-1)/2$)	3.92 (2.7)	3.85 (2.7)	8.82 (4.4)	8.87 (4.6)

3.2 Recombination

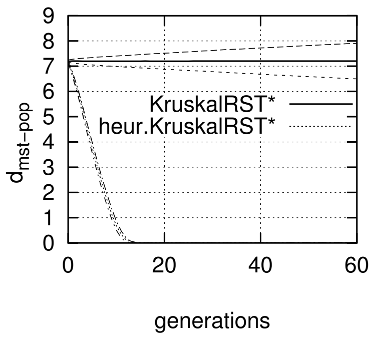
To investigate whether the crossover operator of the edge-set encoding leads to an overrepresentation of MST-like individuals, we randomly generate an initial population of 500 individuals and apply only the crossover operator iteratively. As no selection operator is used, no selection pressure pushes the population to high-quality solutions. The crossover operator is unbiased if the statistical properties of the population do not change by applying crossover alone. In our experiments we measure in each generation the average distance $d_{mst-pop} = 1/N \sum_{i=1}^{N} d_{i,MST}$ of the individuals T_i in the population towards the MST. If $d_{mst-pop}$ decreases, the crossover operator is biased towards the MST. If $d_{mst-pop}$ remains constant, the crossover operator is unbiased regarding the MST.

As before, we perform this experiment on 1000 randomly generated 10 and 16 node tree instances with random, resp. Euclidean weights w_{ij}. For every tree instance we performed 50 runs with different, randomly chosen initial populations (KruskalRST) and 60 generations.

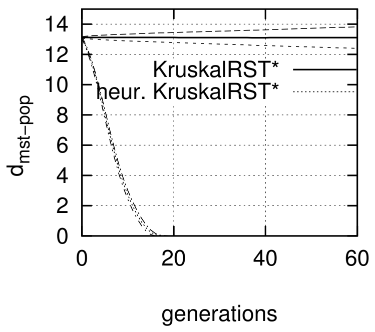
Figure 1 shows the mean and the standard deviation of $d_{mst-pop}$ over the number of generations. The plots compare the non-heuristic KruskalRST*



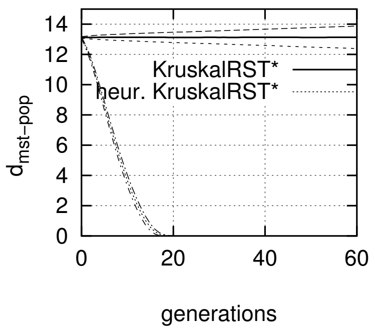
(a) 10 node / random weights



(b) 10 node / Euclidean weights



(c) 16 node / random weights



(d) 16 node / Euclidean weights

Figure 1: The plots show the mean and the standard deviation of the distance $d_{mst-pop}$ between a population of 500 randomly generated individuals towards the MST over the number of generations. Only crossover and no selection is used. The results show that the non-heuristic KruskalRST* crossover is unbiased as the distance between the population and the MST remains constant. In contrast, the heuristic crossover operator is strongly biased towards the MST.

crossover (section 2.1.2) with the heuristic KruskalRST* crossover (section 2.2.2). Only crossover and no selection is used. The results confirm the findings from [24] and reveal that the crossover operator without heuristics is unbiased and does not modify the statistical properties of the population ($d_{mst-pop}$ remains constant over the number of generations). In contrast, the crossover operator with heuristics shows a strong bias towards the MST and the population quickly converges to the MST.

3.3 Mutation

Finally, we investigate the bias of the mutation operator for 1000 random network instances of each type. As for the crossover operator we create a random population of 500 individuals using KruskalRST. Then, in every generation each individual is mutated exactly once using either the non-heuristic "insertion-before-deletion" mutation from section 2.1.3 or the heuristic version from section 2.2.3. Only mutation and no selection is used. For the heuristic mutation operator the parameter β is set to 1, 2, or 5. With lower β, edges with lower weights are preferred.

Figure 2 shows the mean and the standard deviation of $d_{mst-pop}$ over the number of generations. The results show that the non-heuristic mutation operator is unbiased, whereas the heuristic mutation is biased towards the MST. The bias increases with lower β. In contrast to the heuristic crossover operator, the population does not always converge completely towards the MST but the average distance of the population towards the MST remains stable after a few generations.

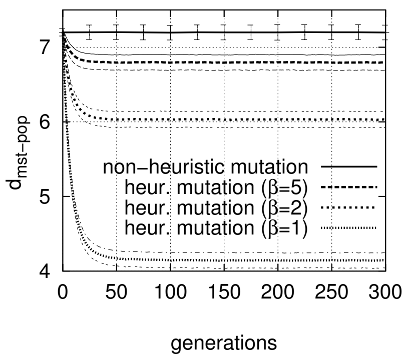
4 The Performance of the Edge-Set Encoding for the OCST problem

4.1 The OCST Problem

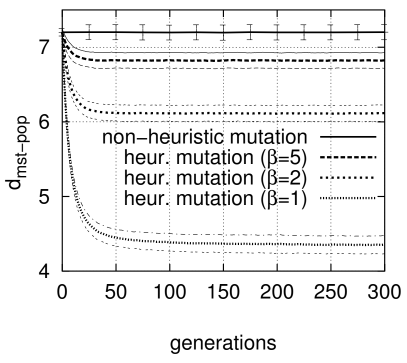
The OCST problem (also known as minimum communication spanning tree problem or simple network design problem) was first introduced in [8]. The problem seeks a spanning tree that connects all given nodes and satisfies their communication requirements for a minimum total cost. The problem can be defined as follows: Let $G = (V, E)$ be a complete undirected graph with $n = |V|$ nodes and $m = |E|$ edges. To every pair of nodes (i, j) a non-negative weight w_{ij} and a non-negative communication requirement r_{ij} is associated. The communication cost $c(T)$ of a spanning tree T is defined as

$$c(T) = \sum_{i,j \in V,\, i<j} r_{ij} \cdot w(p^T_{i,j}),$$

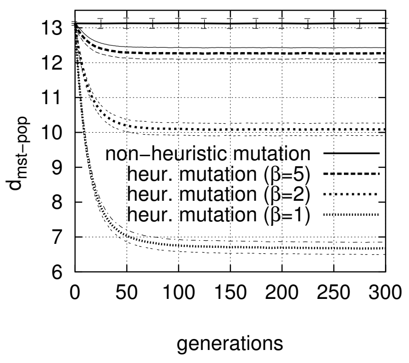
where $w(p^T_{i,j})$ denotes the weight of the unique path from node i to node j in the spanning tree T. The OCST problem seeks the spanning tree with minimal costs among all other spanning trees. The OCST problem becomes the MST problem if there are no communication requirements r_{ij} and $c(T) = \sum_{(i,j)\in E} w_{ij}$ (compare section 3).



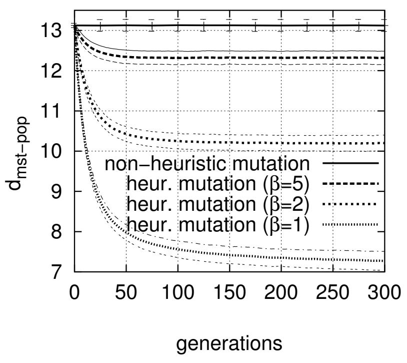
(a) 10 node / random weights



(b) 10 node / Euclidean weights



(c) 16 node / random weights



(d) 16 node / Euclidean weights

Figure 2: The plots show the mean and the standard deviation of the distance $d_{mst-pop}$ between a population of 500 individuals towards the MST over the number of generations. Only mutation ("insertion before deletion") and no selection is used. The results show that the non-heuristic mutation operator is unbiased. The heuristic mutation operator is biased and the bias increases with lower β.

Like many other spanning tree problems, the OCST problem is $\mathcal{NP}$-hard [5]. Further more, it was shown in [15] that the OCST problem is $\mathcal{MAX}$ $\mathcal{NP}$-hard, that means it cannot be solved using a polynomial-time approximation-scheme, unless $\mathcal{P} = \mathcal{NP}$. The OCST problem has been studied extensively in the literature and many researchers have tried to develop efficient optimization algorithms. The current best approximation algorithm for the OCST problem with Euclidean weights approximates the optimal solution with $c(T) = O(\log n)\cdot c(G)$ [16], where $c(G)$ is the cost of the network when using the complete graph G. $c(G)$ is a lower bound for $c(T)$ as the weight of the unique path between node i and j in a spanning tree T is greater or equal in comparison to the weight of the path with minimal weight connecting the nodes i and j in G. As there are no exact optimization methods available and the approximation algorithms only generate low-quality solutions, many researchers used EAs for solving the OCST problem [2,3,9,10,13,20].

It was shown in [21] that on average optimal solutions for OCST problems are similar to the MST. That means the average distance $d_{opt,MST}$ between the optimal solution and the MST is significantly lower than the average distance $d_{rand,MST}$ between a randomly created tree and the MST. Therefore, as the optimal solution of an OCST problem is biased towards the MST, representations as well as operators that favor or overrepresent trees, which are similar to the MST are expected to solve the OCST problem more efficiently.

4.2 Finding Optimal Solutions for OCST Problems

To investigate how the performance of the edge-set encoding depends on the structure of the optimal solution, an optimal or near-optimal solution must be determined. Due to the $\mathcal{NP}$-hardness of the OCST problem, optimal solutions can be determined only for small problem instances with reasonable computational effort. Therefore, we limit our investigations to 10 and 16 node problem instances. The following experiments, which should identify optimal or near-optimal solutions for OCST problems, are similar to the ones described in [21].

[7] showed that the probability α that an EA fails to find an optimal solution is $O(\exp(-N))$, where N is the EA's population size. Therefore, EA performance increases with N. Consequently, we apply an EA n_{iter} times to an OCST problem using a population size of N_0. T_0^{best} denotes the best solution of cost $c(T_0^{best})$ that is found during the n_{iter} runs. In a next round we double the population size and again apply an EA n_{iter} times with a population size of $N_1 = 2N_0$. T_1^{best} denotes the best solution with cost $c(T_1^{best})$ 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.

For finding the optimal solutions we use a standard genetic algorithm (sGA) with traditional parameter settings. The problem was encoded using the unbiased network random key representation [23]. This representation has high locality and represents all possible trees uniformly. The sGA uses uniform

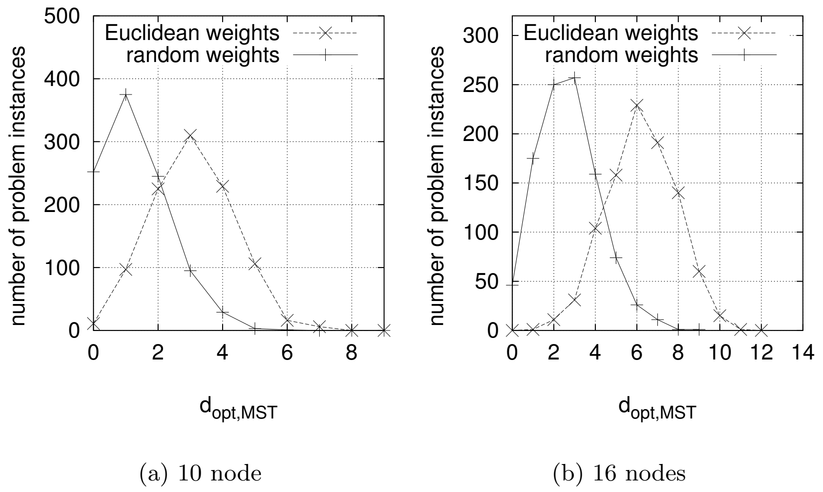


Figure 3: We randomly generated 1000 OCST problems and show the distribution of the problem instances over the distance $d_{opt,MST}$ between the optimal solution and the MST. Results are presented for 10 and 16 node problems using either random or Euclidean weights. The plots show that the optimal solutions for OCST problems are biased towards the MST.

crossover and tournament selection without replacement. The size of the tournament is three. The crossover probability is set to $p_{cross} = 0.8$ and the mutation probability (assigning a random value $[0, 1]$ to one allele) is set to $p_{mut} = 0.02$. For the sGA we started with $N_0 = 100$ and set $n_{iter} = 20$. Each sGA run is stopped after a maximum of 200 generations. The computational effort for the experiments is high.

Figure 3 presents the results of our experiments. We show the number of problem instances over the distance $d_{opt,MST}$ between the optimal solution and the MST for 1000 randomly created OCST problems with 10 (Fig. 3(a)) and 16 (Fig. 3(b)) nodes. The OCST problems are created randomly using either random weights in]0,100] or placing the nodes randomly on a 1000x1000 two-dimensional grid and calculating the weights as the Euclidean distances between the nodes (details are described in section 3). The demands r_{ij} between the nodes are random and uniformly distributed in]0,100].

Comparing the results to $d_{rand,MST}$ (unbiased representation in Table 1) reveals that the optimal solutions for OCST problems are strongly biased towards the MST. Furthermore, OCST problems with random weights show a stronger bias than OCST problems with Euclidean weights. Due to the bias of the optimal solutions towards the MST, the problem should be easy to solve for EAs using the edge-set encoding.

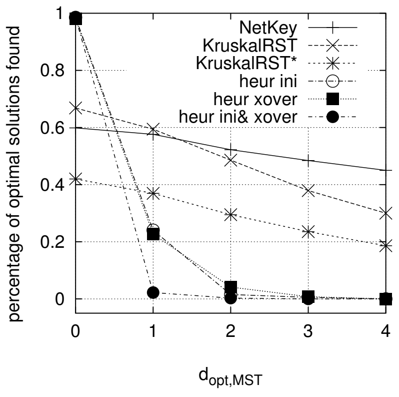
4.3 The Performance of the Edge-Set Encoding for Randomly Generated OCST Problems

After determining optimal solutions as described in the previous paragraphs we examine the performance of EAs using the edge-set encoding. We use the same randomly generated problem instances as in section 4.2 and investigate how the EA performance depends on the distance $d_{opt,MST}$ between the optimal solution and the MST. We use a generational EA with tournament selection without replacement of size two and no mutation. Each run is stopped after the population is fully converged or the number of generations exceeds 200. We perform 50 EA runs for each of the 1000 problem instances. In our experiments we compare the performance of EAs using

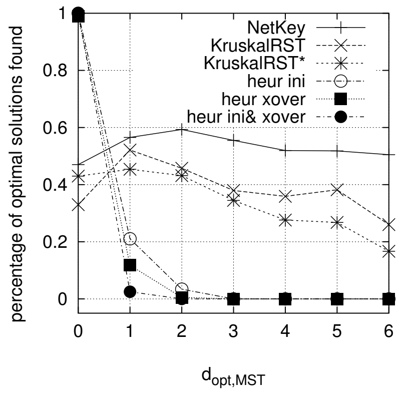
- non-heuristic KruskalRST crossover from section 2.1.2 with non-heuristic KruskalRST initialization from section 2.1.1 (indicated as "KruskalRST"),
- non-heuristic KruskalRST* crossover from section 2.1.2 combined with non-heuristic KruskalRST initialization (indicated as "KruskalRST*"),
- non-heuristic KruskalRST* crossover combined with heuristic initialization from section 2.2.1 with $\alpha = 1.5$ (indicated as "heur ini"),
- heuristic crossover from section 2.2.2 combined with non-heuristic KruskalRST initialization (indicated as "heur xover"),
- heuristic crossover combined with heuristic initialization with $\alpha = 1.5$ (indicated as "heur ini & xover"), and
- as benchmark the unbiased network random key encoding with uniform crossover (indicated as "NetKey").

The population size N, which is constant in all experiments, is chosen with respect to the performance of the non-heuristic KruskalRST* crossover operator. The aim is to find the optimal solution with a probability of about 50 %. Therefore, we choose for the 10 node problems a population size of $N = 60$ (random weights) resp. $N = 100$ (Euclidean weights) and for the 16 node problems a population size of $N = 200$ (random weights) resp. $N = 450$ (Euclidean weights).

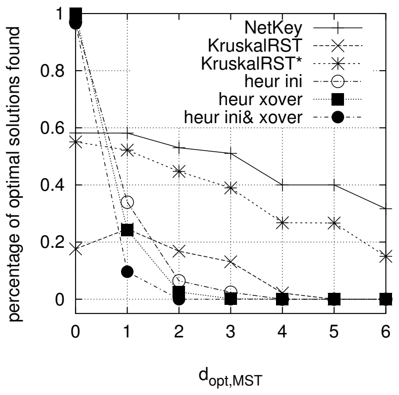
The results of the experiments are presented in Fig. 4 and Fig. 5. Fig. 4 shows the percentage of EA runs that find the correct optimal solutions over $d_{opt,MST}$. Fig. 5 shows the gap, $\frac{c(T_{found})-c(T_{opt})}{c(T_{opt})}$ (in percent), between the cost of the best found solution and the cost of the optimal solution over $d_{opt,MST}$. We show results for 1000 randomly generated problem instances. Results are plotted only for these $d_{opt,MST}$, where there are more than 10 problem instances. For example, we show results for 10 node problems with Euclidean weights only for $d_{opt,MST} \in \{0, \ldots, 6\}$ as there are only 8 (out of 1000) instances with $d_{opt,MST} = 7$ (compare Fig. 3(b)).



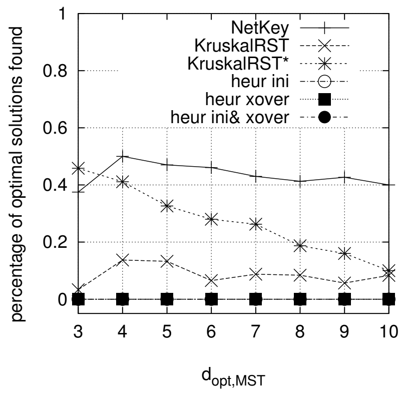
(a) 10 node / random weights



(b) 10 node / Euclidean weights

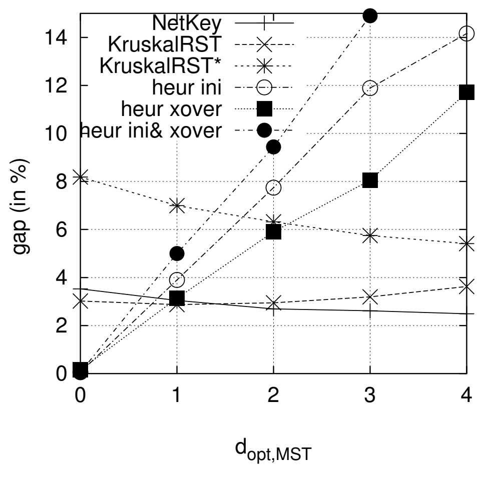


(c) 16 node / random weights

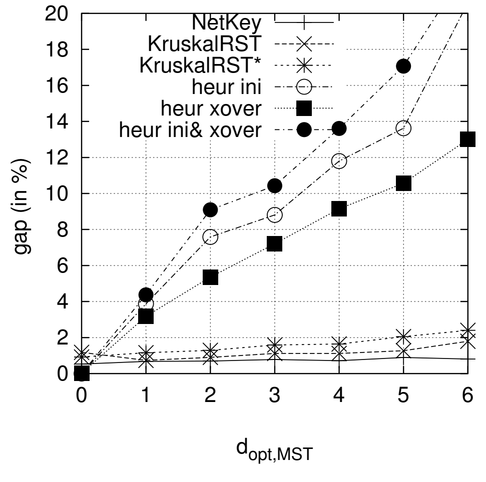


(d) 16 node / Euclidean weights

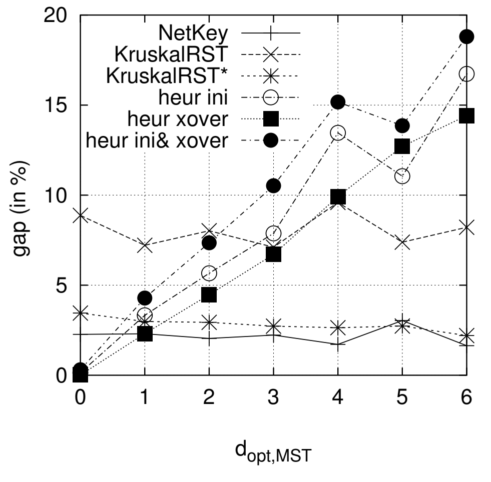
Figure 4: The figures compare the performance of an EA using different combinations of crossover and initialization operators for randomly generated 10 (top) and 16 (bottom) node OCST problems. The plots show the average percentage of optimal solutions that can be found over $d_{opt,MST}$. The heuristic crossover operator outperforms the non-heuristic version only if the optimal solution is very similar to the MST ($d_{opt,MST} \approx 0$). If $d_{opt,MST} > 1$ the heuristic crossover results in low EA performance. In contrast, when using the non-heuristic KruskalRST* crossover, EA performance remains about constant.



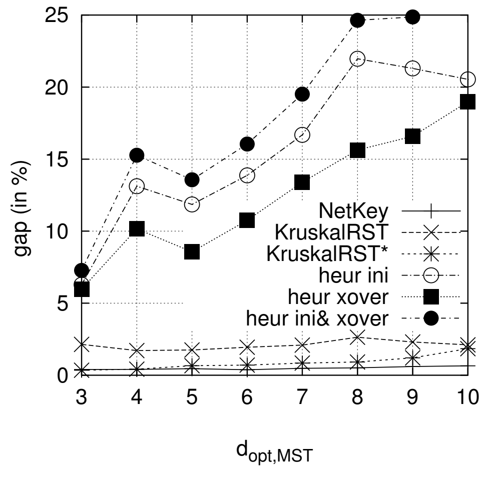
(a) 10 node / random weights



(b) 10 node / Euclidean weights



(c) 16 node / random weights



(d) 16 node / Euclidean weights

Figure 5: We show the mean of the gap between the cost of the best found solution and the cost of the optimal solution over $d_{opt,MST}$. The results confirm that the heuristic crossover operator outperforms the non-heuristic variants only if the optimal solutions are very similar to the MST ($d_{opt,MST} \approx 0$).

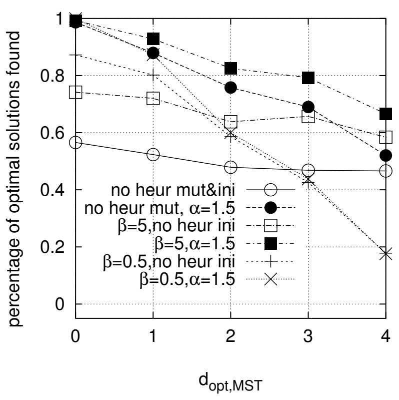
The results reveal that the heuristic crossover versions of the edge-set encoding (heur xover and heur ini & crossover) always find the optimal solution if the optimal solution is the MST. However for $d_{opt,MST} \neq 0$, the performance of EAs using the heuristic version drops sharply and the optimal solution can not be found if $d_{opt,MST} > 2$. In contrast, the performance of the non-heuristic KruskalRST* operator decreases only slightly with larger $d_{opt,MST}$ and allows the EA to correctly identify the optimal solution even for larger $d_{opt,MST}$. The performance of the non-heuristic crossover combined with an heuristic initialization ("heur ini") is similar to the heuristic crossover operator. It always finds the optimal solution if it is the MST, however with increasing $d_{opt,MST}$ the decrease of performance is slightly less than for the heuristic crossover.

In summary, the heuristic crossover operator performs well only for problems where the optimal solution is slightly different from the MST. Otherwise, EAs using the edge-set encoding with heuristic crossover fail. The performance of EAs using the non-heuristic variant is similar to the performance of the NetKey encoding with uniform crossover. These results are confirmed when examining the gap $\frac{c(T_{found})-c(T_{opt})}{c(T_{opt})}$ (Fig. 5). Heuristic variants of the encoding show high performance if the optimal solution is the MST. However, with increasing $d_{opt,MST}$ the quality of the solutions strongly decreases and the non-heuristic variants outperform the heuristic variant.

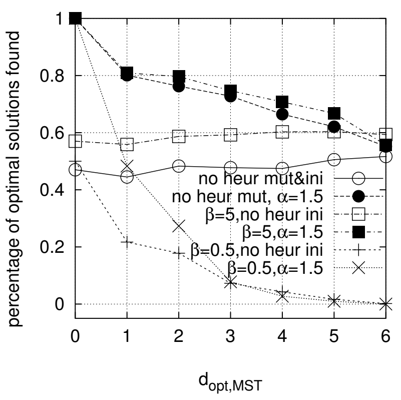
In the remainder of the section the performance of the edge-set-specific mutation operator is examined. As before 1000 random problems of different types are generated and the optimal solutions are calculated as described in section 4.2. For comparing the performance of different variants of the mutation operator, a simple simulated annealing (SA) strategy [25] is used as a representative example of mutation-based search. SA can be modeled as an EA with population size one and Boltzmann selection [11]. In each generation a new solution T_{off} is created by applying exactly one mutation to the parent solution T_{par}. If $c(T_{off}) < c(T_{par})$, T_{off} replaces T_{par}. If $c(T_{off}) > c(T_{par})$, T_{par} is replaced with probability $P(T) = \exp\left(-(c(T_{off}) - c(T_{par}))/T\right)$. With lower T, the probability of accepting worse solutions decreases.

In our experiments the start temperature $T_{start} = 50$ is reduced in every step by the factor 0.99. Therefore, $T_{t+1} = 0.99 * T_t$. The number of search steps is set to $t_{max} = 300$ for 10 node and $t_{max} = 1000$ for 16 node problems. We performed 50 independent runs for each problem instance and investigated the performance of an SA using

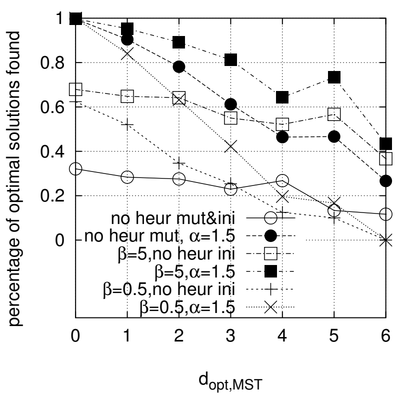
- non-heuristic mutation from section 2.1.3 and non-heuristic initialization from section 2.1.1 (denoted as "no heur mut&ini"),
- non-heuristic mutation and heuristic initialization from section 2.2.1 with $\alpha = 1.5$ (denoted as "no heur mut, $\alpha = 1.5$"),
- heuristic mutation from section 2.2.3 with $\beta = 5$ and non-heuristic initialization (denoted as "$\beta = 5$, no heur ini"),
- heuristic mutation with $\beta = 5$ and heuristic initialization with $\alpha = 1.5$ (denoted as "$\beta = 5, \alpha = 1.5$"),



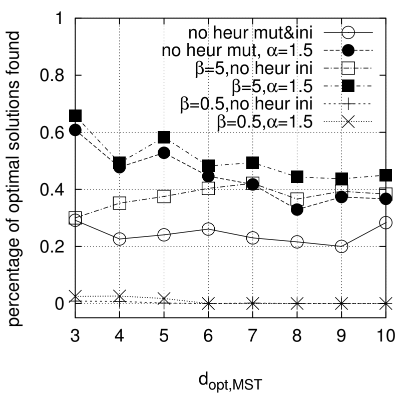
(a) 10 node / random weights



(b) 10 node / Euclidean weights



(c) 16 node / random weights



(d) 16 node / Euclidean weights

Figure 6: The figures show the performance of an SA using different variants of initialization and mutation operators of the edge-set encoding. The plots show the average percentage of optimal solutions that can be found over $d_{opt,MST}$ for 1000 randomly created OCST problems. The results show that the heuristic variants of the mutation operator outperform the non-heuristic variants for the OCST problem if β is set properly.

- heuristic mutation with $\beta = 0.5$ and non-heuristic initialization (denoted as "$\beta = 0.5$, no heur ini"), and
- heuristic mutation with $\beta = 0.5$ and heuristic initialization with $\alpha = 1.5$ (denoted as "$\beta = 0.5, \alpha = 1.5$").

We performed no experiments for NetKeys as the corresponding mutation operator can not be directly compared. The mutation operator for NetKeys, which changes one allele of the genotype, often does not change the corresponding phenotype, whereas the mutation operator of the edge-set encoding always changes one edge.

The results of the experiments are presented in Fig. 6. It shows the percentage of SA runs that find the correct optimal solutions over $d_{opt,MST}$. It can be seen that an SA using heuristic initialization always finds the optimal solution if $d_{opt,MST} = 0$. When using heuristic mutation with a low bias ($\beta = 5$), SA performance is always higher than when using non-heuristic mutation (for all considered $d_{opt,MST}$). A small bias of the mutation operator does not push the population towards the MST but allows a diversed population and efficient SA search for solutions somehow similar to the MST. However, when increasing the bias of the heuristic mutation to $\beta = 0.5$, SA performance becomes lower than for the non-heuristic even for small $d_{opt,MST}$ (especially for the Euclidean problem instances). Then, the heuristic bias of the mutation operator is too strong and pushes the population too strongly towards the MST. The results reveal that by increasing the bias of the mutation operator (lowering β) problems where the optimal solutions are similar to the MST can be solved more efficiently; however, problems where the optimal solutions are different from the MST can be solved less efficiently.

To summarize our findings, the heuristic crossover operator of the edge-set encoding does not allow efficient search due to its strong bias towards the MST. Only problems where the optimal solutions are slightly different from the MST can be solved. The heuristic mutation operator results in good performance if β is large as the resulting low bias of the mutation operator prefers solutions similar to the MST and does not push a population too strongly towards the MST. However, if the bias towards the MST induced by β becomes stronger only optimal solutions similar to the MST can be found and mutation-based search fails. The results for the heuristic mutation operator show that the proper adjustment of β is important and crucial for the success of local search.

4.4 The Performance of the Edge-Set Encoding for OCST Test Problems from the Literature

Test instances for the OCST problem have been proposed by [2, 13, 19]. An analysis of the test instances was performed in [21]. The following paragraphs examine the performance of EA and SA using edge-sets for these test instances.

[13] described OCST problems with six (palmer6), twelve (palmer12), 24 (palmer24), 47, and 98 nodes. The nodes correspond to cities in the United States and the weights (distances between the nodes) are obtained from a tariff database. The inter-node traffic demands are inversely proportional to the

weights. [2] presented three instances of the OCST problem, one with six nodes (berry6) and two with 35 nodes (berry35 and berry35u). For berry35u the weights $w_{ij} = 1$. [19] proposed several test instances ranging from 10 to 100 nodes. The weights w_{ij} and the demands r_{ij} were generated randomly and are uniformly distributed in the interval $[0, 100]$. We present experiments for the 10 and 20 node test instances as the optimal solutions can be found with reasonable effort.

Table 2 lists the properties of the optimal solutions for the test instances. It shows the number of nodes n, the distance $d_{opt,MST}$, and the cost $c(T_{opt})$ of the optimal solution. In the instance berry35u, all distances are uniform ($w_{ij} = 1$), so all spanning trees are minimal. For all test instances, $d_{opt,MST}$ is smaller than the average distance of a randomly created solution towards the MST (compare also [21]). Therefore, all test problems are biased towards the MSTs.

For investigating the performance of EAs using different variants of the edge-set encoding we use the same parameter setting as described in the previous section. The population size N was set to $N = 80$ for the the small problem instances (palmer6, palmer12, berry6) and to $N = 1000$ for the more complicated problems (raidl20, berry35u, berry35). As before, an EA runs a maximum of 200 generations or until the population is converged. Table 2 lists the mean μ and the standard deviation σ of the cost of the best solution found at the end the EA run averaged over 50 independent runs. The results reveal that for berry6 and berry35, where the optimal solution is the MST, EAs using the heuristic crossover always find the optimal solution. However, with increasing $d_{opt,MST}$, EAs using the heuristic crossover find worse solutions than using the non-heuristic crossover or the NetKey encoding.

Similarly, table 3 presents the mean μ and standard deviation σ of the cost of the best found solution when using SA with different mutation and initialization operators. We use the same parameter setting as in the previous section ($T_0 = 50$ and $T_t = T_{t-1} * 0.99$) and only change the number of search steps t_{max} for the different problem instances. The results show that heuristic mutation mostly outperforms non-heuristic mutation if $d_{opt,MST} < 5$. For larger $d_{opt,MST}$ (palmer12 and palmer24) the performance of heuristic mutation decreases with lower β (compare especially palmer24) as the bias of the mutation operator towards the MST becomes stronger. If β is set properly, heuristic mutation outperforms the non-heuristic version especially if $d_{opt,MST}$ is low.

The results confirm the findings from the previous section. The non-heuristic KruskalRST* crossover shows similar performance to NetKeys and the performance is about independent of $d_{opt,MST}$. The heuristic crossover operator fails and only finds optimal solutions if they are the MST. The heuristic mutation operator results in good SA performance if β is properly set and if the optimal solutions are similar to the MST. Consequently, as most OCST problems have optimal solutions similar to the MST, local search using the mutation operator outperforms the non-heuristic variants.

Table 2: Performance of EA using different variants of the crossover operator for OCST test problems from the literature

problem		optimal solutions		N	NetKeys		KruskalRST		KruskalRST*		heur. ini		heur. xover		heur. ini & xover	
instance	nodes	$d_{opt,MST}$	$c(T_{opt})$		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
palmer6	6	1	693,180	80	695,103	5,002	697,197	7,320	694,113	3,291	709,770	0	705,457	7351	709,770	0
palmer12	12	7	3,428,509	80	3,500,159	55,248	3,521,759	64,144	3,542,906	70,441	3,738,047	15,615	3,734,503	57,469	3,876,488	0
palmer24	24	12	1,086,656	500	1,086,780	290	1,558,453	69,883	1,087,772	631	1,884,444	0	1,879,996	15,395	1,959,790	0
raidl10	10	3	53,674	80	54,400	1,776	54,251	1,264	54,985	3106	55,868	0	57,625	589	58,352	0
raidl20	20	4	157,570	1,000	159,447	3,025	205,675	12,138	157,570	0	169,746	4,204	165,788	0	169,086	4,123
berry6	6	0	534	80	534	0	534	0	534	0	534	0	534	0	534	0
berry35u	35	-	16,273	1,000	17,067	334	27,210	1,290	16,459	120	41,021	4,217	16,478	119	43,589	5,306
berry35	35	0	16,915	1,000	16,919	21	36,985	2,189	17,102	306	16,915	0	16,915	0	16,915	0

Table 3: Performance of SA using different variants of the mutation operator for OCST test problems from the literature

		no heur. mut,				heur. mut ($\beta = 0.5$)				heur. mut ($\beta = 1$)				heur. mut ($\beta = 5$)			
problem		no heur. ini		($\alpha = 1.5$)		no heur. ini		($\alpha = 1.5$)		no heur. ini		($\alpha = 1.5$)		no heur. ini		($\alpha = 1.5$)	
instance	t_{max}	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
palmer6	300	700,337	7,525	699,144	7,379	707,039	16,599	693,478	2,109	700,337	7,525	698,249	7,135	698,548	7,229	699,442	7,434
palmer12	300	3,527,997	79,570	3,483,322	40,332	4,042,556	371,097	3,610,149	46,581	3,477,738	46,109	3,493,127	39,767	3,500,706	56,927	3,481,875	41,608
palmer24	1,000	1,161,370	70,209	1,137,968	42,835	1,932,729	269,545	1,830,098	22,734	1,438,231	164,891	1,525,089	100,232	1,097,958	19,578	1,100,706	25,670
raidl10	300	55,988	4,384	53,853	517	53,867	448	53,762	303	53,674	0	53,674	0	54,282	1,674	53,762	303
raidl20	1,000	182,168	15,793	159,660	3,140	161,791	14,596	157,570	0	159,270	3057	157,570	0	164,944	6,561	157,849	803
berry6	300	538	11	536	7	534	0	534	0	534	0	535	4	536	7	535	7
berry35	5000	17,142	423	16,915	0	16,915	0	16,915	0	16,915	0	16,915	0	16,915	0	16,915	0

5 Summary and Conclusions

This work investigates the bias of the edge-set encoding, which was proposed by [18], and examines its performance for the optimal communication spanning tree (OCST) problem. The edge-set encoding belongs to the class of direct representations for trees. Instead of defining an additional genotype-phenotype mapping encoding-specific initialization, crossover and mutation operators are applied directly to the trees.

The investigation into the bias of the edge-set encoding reveals that the heuristic versions of the initialization, crossover, and mutation operators, are biased towards the MST defined on the weights. The bias is especially strong for the heuristic crossover operator, which results in a quick convergence of a population of trees towards the MST. In contrast, the non-heuristic search operators of the edge-sets are unbiased and their application results in an undirected and uniform search through the search space.

Due to the strong bias of the heuristic search operators towards the MST, tree optimization problems can easily be solved if optimal solutions are the MST. However, if optimal solutions are only slightly different from the MST, the heuristic crossover operator fails due to its strong bias towards the MST. Therefore, the heuristic crossover operator is not appropriate for solving tree optimization problems. In contrast, the non-heuristic crossover operator of the edge-sets results in good performance for OCST problems even if the optimal solutions are quite different from the MST. Its performance is similar to the unbiased, indirect NetKey encoding [23]. For the mutation operator the strength of the bias towards the MST can be controlled by an encoding-specific parameter β. With high β the bias towards the MST is low, with low β it is strong. Therefore with low β, tree problems can be solved more efficiently if the optimal solutions are similar to the MST but otherwise less efficiently. If β is set appropriately, the heuristic mutation operator is a good choice for OCST problems as optimal solutions of this problem are similar to the MST.

The problems of the heuristic variants of the edge-set encoding emphasize the difficulty of a proper design of representations and operators. Especially the design of direct representations is difficult as in contrast to indirect representations, the behavior of new, problem-specific search operators is often unknown. The analysis of the edge-set encoding has shown that although optimal solutions for the OCST problems are biased towards the MST [21], direct representations like the heuristic edge-set encoding that use this problem-specific knowledge and are biased towards the MST, can fail if the bias is too strong. Therefore, the authors recommend the use of unbiased representations if no problem-specific knowledge is known a priori. Proper representations for tree problems are for example non-heuristic versions of the edge-set encoding or NetKeys. In the case that biased representations resp. operators like the heuristic mutation operator of the edge-set encoding are used, it must be confirmed that the bias of the search matches the properties of the optimal solutions. Otherwise failure is unavoidable.

References

[1] F. N. Abuali, R. L. Wainwright, and D. A. Schoenefeld, "Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, L. Eschelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 470–477.

[2] L. T. M. Berry, B. A. Murtagh, and G. McMahon, "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*, Pretoria, South Africa, 1995, pp. 361–370.

[3] H. Chou, G. Premkumar, and C.-H. Chu, "Genetic algorithms for communications network design - an empirical study of the factors that influence performance," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 236–249, June 2001.

[4] S. Fekete, S. Khuller, M. Klemmstein, B. Raghavachari, and N. Young, "A network-flow technique for finding low-weight bounded-degree spanning trees," *Journal of Algorithms*, vol. 24, pp. 310–324, 1997.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[6] J. Gottlieb, B. A. Julstrom, G. R. Raidl, and F. Rothlauf, "Prüfer numbers: A poor representation of spanning trees for evolutionary search," in *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, L. Spector, E. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. San Francisco, CA: Morgan Kaufmann Publishers, 2001, pp. 343–350.

[7] G. R. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," in *Proceedings of the Forth International Conference on Evolutionary Computation*, T. Bäck, Ed. New York: IEEE Press, 1997, pp. 7–12.

[8] T. C. Hu, "Optimum communication spanning trees," *SIAM Journal on Computing*, vol. 3, no. 3, pp. 188–195, Sept. 1974.

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

[10] Y. Li and Y. Bouchebaba, "A new genetic algorithm for the optimal communication spanning tree problem," in *Proceedings of Artificial Evolution: Fifth European Conference*, C. Fonlupt, J.-K. Hao, E. Lutton, E. Ronald, and M. Schoenauer, Eds. Berlin: Springer, 1999, pp. 162–173.

[11] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm," in *Parallel Computing.* Amsterdam, The Netherlands: Elsevier Science, 1995, vol. 21, pp. 1–28.

[12] S. C. Narula and C. A. Ho, "Degree-constrained minimum spanning trees," *Computers and Operations Research*, vol. 7, pp. 239–249, 1980.

[13] C. C. Palmer, "An approach to a problem in network design using genetic algorithms," unpublished PhD thesis, Polytechnic University, Troy, NY, 1994.

[14] C. C. Palmer and A. Kershenbaum, "Representing trees in genetic algorithms," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, vol. 1. Piscataway, NJ: IEEE Service Center, 1994, pp. 379–384.

[15] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Comput. System Sci.*, vol. 43, pp. 425–440, 1991.

[16] D. Peleg and E. Reshef, "Deterministic polylog approximation for minimum communication spanning trees," *Lecture Notes in Computer Science*, vol. 1443, pp. 670–682, 1998.

[17] H. Prüfer, "Neuer Beweis eines Satzes über Permutationen," *Archiv für Mathematik und Physik*, vol. 27, pp. 742–744, 1918.

[18] G. R. Raidl and B. A. Julstrom, "Edge-sets: An effective evolutionary coding of spanning trees," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 225–239, 2003.

[19] G. R. Raidl, "Various instances of optimal communication spanning tree problems," personal communciation, February 2001.

[20] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 1st ed., ser. Studies on Fuzziness and Soft Computing. Heidelberg: Springer, 2002, no. 104.

[21] F. Rothlauf, J. Gerstacker, and A. Heinzl, "On the optimal communication spanning tree problem," Department of Information Systems, University of Mannheim, Tech. Rep. 15/2003, 2003.

[22] F. Rothlauf and D. E. Goldberg, "Redundant representations in evolutionary computation," *Evolutionary Computation*, vol. 11, no. 4, pp. 381–415, 2003.

[23] F. Rothlauf, D. E. Goldberg, and A. Heinzl, "Network random keys – A tree network representation scheme for genetic and evolutionary algorithms," *Evolutionary Computation*, vol. 10, no. 1, pp. 75–97, 2002.

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

[25] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Dordrecht, The Netherlands: Kluwer, 1988.