Orientation Matters: How to Efficiently Solve OCST Problems with Problem-specific EAs

Wolfgang Steitz
Dept. of Information Systems and Business Administration
University of Mainz, Germany
steitzw@uni-mainz.de

Prof. Dr. Franz Rothlauf
Dept. of Information Systems and Business Administration
University of Mainz, Germany
rothlauf@uni-mainz.de

ABSTRACT

The optimal communication spanning tree (OCST) problem is a well known $\mathcal{NP}$-hard combinatorial optimization problem which seeks a spanning tree that satisfies all given communication requirements for minimal total costs. It has been shown that optimal solutions of OCST problems are biased towards the much simpler minimum spanning tree (MST) problem. Therefore, problem-specific representations for EAs like heuristic variants of edge-sets that are biased towards MSTs show high performance.

In this paper, additional properties of optimal solutions for Euclidean variants of OCST problems are studied. Experimental results show that not only edges in optimal trees are biased towards low-distance weights but also edges which are directed towards the graph's center are overrepresented in optimal solutions. Therefore, efficient heuristic search algorithms for OCST should be biased towards edges with low distance weight *and* edges that point towards the center of the graph. Consequently, we extend the recombination operator of edge-sets such that the orientation of the edges is considered for constructing offspring solutions. Experimental results show a higher search performance in comparison to EAs using existing crossover strategies of edge-sets. As a result, we suggest to consider not only the distance weights but also the orientation of edges in heuristic solution approaches for the OCST problem.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Performance

Keywords

optimal communications spanning tree, heuristics, evolutionary algorithm, recombination operators

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

The optimal communication spanning tree (OCST) problem [3] is a common $\mathcal{NP}$-hard combinatorial optimization problem which seeks a spanning tree that satisfies all communication requirements and leads to minimal total costs. For decades, researches studied various solution approaches for the OCST problem [10, 14]. The current state-of-the-art approaches are based on heuristics and metaheuristics, in particular evolutionary algorithms (EA), to solve the problem. Especially EAs using the edge-set encoding [9] show good performance for many real-world problems where the optimal solutions are similar to MSTs. The edge-set encoding is a direct representation for trees which directly represents trees as sets of edges and uses encoding-specific search operators to generate candidate solutions. There exist heuristic versions which rely on distance weights as well as non-heuristic versions.

In this paper, we study additional properties of high quality solutions for the Euclidean variants of the OCST problem. The analysis shows that the orientation of the edges matters and high quality solutions contain edges that point in the direction towards the center of the tree with higher probability. Consequently in a second step, we make use of this observation and extend the crossover operator of the edge-set encoding such that not only the distance weights but also the orientation of the edges is used for constructing an offspring from parental edges. Experimental results for a number of test problems reveal that this allows to improve search performance of EAs.

The following section defines the OCST problem, presents properties of optimal solutions, lists various OCST test instances, and describes how to determine optimal solutions for small problem instances of the OCST problem. Additionally, it provides experimental results on the orientation of edges in optimal solutions. In Sect. 3, the edge-set encoding is described and we extend the heuristic crossover operator of edge-sets in such a way that it relies on the weights *and* orientations of the edges. Sect. 3 studies the performance of EAs using the extended heuristic crossover operator for different test instances and Sect. 4 presents concluding remarks.

2. THE OCST PROBLEM

2.1 Problem Definition

The OCST problem is a common $\mathcal{NP}$-hard combinatorial optimization problem and was first introduced by Hu [3]. The goal is to seek a tree which connects all given

nodes and satisfies their communication requirements for a minimum total costs. It can be formulated as follows: Let $G = (V, E)$ be a weighted, undirected graph with $n = |V|$ nodes and $m = |E|$ edges. The communication or transport requirements between the n different nodes are given a priori in the $n \times n$ *demand matrix* $R = (r_{ij})$. Analogically, the $n \times n$ *distance weight matrix* $W = (w_{ij})$ specifies the distance weights. The weight $w(T)$ of a tree $T = (V, F)$ with $(F \subseteq E)$ and $|F| = n - 1$ is calculated as follows:

$$w(T) = \sum_{i,j \in V} w_{ij} b_{ij}, \quad (1)$$

where b_{ij} denotes the traffic flowing directly or indirectly over the edge between nodes v_i and v_j. The traffic is calculated according to the structure of T and the demand matrix R. T is the optimal communication spanning tree, if $w(T) \leq w(T')$ for all other spanning trees $w(T')$.

To measure the difference between two spanning trees T_i and T_j, the distance $d_{ij} \in \{0, 1, \ldots, n-1\}$ can be calculated as

$$d_{ij} = \frac{1}{2} \sum_{u,v \in V, u<v} |e^i_{uv} - e^j_{uv}|. \quad (2)$$

$e^i_{uv} = 1$ if edge e_{uv} is included in T_i and $e^i_{uv} = 0$ if not.

Like many other constrained graph problems, the OCST problem is $\mathcal{NP}$-hard [2]. Furthermore, since the problem is $\mathcal{MAX}$ $\mathcal{SNP}$-hard [6], no polynomial-time approximation scheme exists, unless $\mathcal{NP} = \mathcal{P}$. Only for a few restricted problem instances algorithms exist, which return optimal solutions [17]. In addition, various approximation algorithms for the OCST problem have been developed [7, 18, 19]. However, due to the $\mathcal{MAX}$ $\mathcal{SNP}$-hardness of the problem the solution quality of such approximation algorithms is very limited. To overcome the limitations of exact and approximation algorithms, many heuristics, especially EAs have been developed [5, 12, 9, 15, 1]. For an overview of EAs for the OCST problem, we refer to Rothlauf [10].

Rothlauf et al. [13] analyzed the properties of the OCST problem and showed that optimal solutions are biased towards the minimum spanning tree (MST). The distances between optimal solutions and MSTs are significantly smaller than the distances between optimal solutions and randomly generated solutions. Thus, if an optimization method for the OCST problem is biased towards MST-like solutions, the performance can be increased. This fact is used by the heuristic variants of edge-sets [9, 8], which favor edges with a low distance weight [16].

2.2 Test Instances

Several authors provided test instances for the OCST problem. In the appendix of [10] a comprehensive collection can be found. In addition, following Raidl [9] and Rothlauf [10], we use randomly created OCST test instances. For our test instances, the real-valued demands r_{ij} are randomly created and are uniformly distributed in $[0, 10]$. The real-valued distance weights w_{ij} are calculated as the Euclidean distances between the nodes i and j which are randomly placed on a 10×10 2-dimensional grid.

2.3 Finding Optimal Solutions

For finding optimal, or at least near-optimal solutions of OCST problems, we used a mathematical programming solver for small problem instances with $n \leq 12$ and a GA for larger problem instances. The OCST problems are modeled as an integer linear program [11], and CPLEX 10.2 is able to solve all problem instances with $n \leq 12$ in reasonable time.

The situation is different for larger problem instances ($n > 12$) which can not solved by CPLEX in reasonable time. Therefore, we used an iterative GA for such problems. 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 the found solution is optimal or near-optimal. We start the iterative GA by applying a standard GA n_{iter} times to an OCST 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 set to $p_{cross} = 0.7$ and the mutation probability (assigning a random value $[0, 1]$ to one allele) is set to $p_{mut} = 1/l$, where $l = n(n-1)2$.

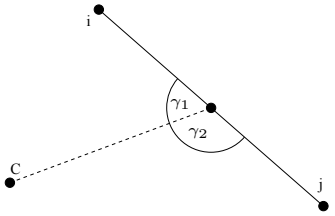


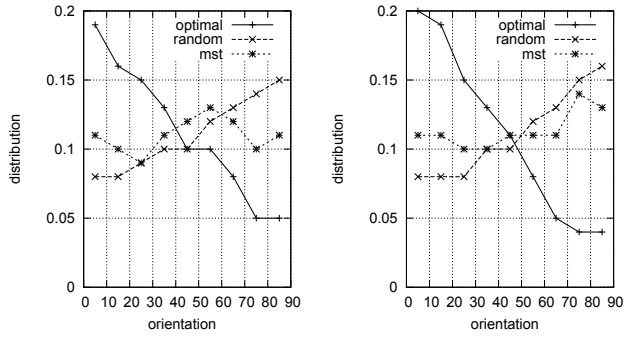
Figure 1: orientation of an edge e_{ij}

2.4 Orientation of Edges in Optimal Solutions

To identify properties of high-quality solutions for OCST problems, we analyze the orientation of the edges in optimal solutions. We assume that edges which are directed towards the center of a graph are overrepresented in optimal solutions. Such edges lead to shorter paths between pairs of edges and hence to lower costs of a solution.

Figure 1 illustrates the calculation of the orientation of an edge. The orientation of an edge e_{ij} is the angle $\gamma \in [0, 90]$ between e_{ij} and the line connecting the midway of e_{ij} and the center C of the tree. C is calculated as the average x-coordinates and y-coordinates of all nodes in the graph. Since $\gamma \leq 90$, the lower angle is chosen ($\gamma = min(\gamma_1, \gamma_2)$). For edges directly pointing to the center, $\gamma = 0$.

To study the orientation of edges in optimal solutions, we compare the orientation of the edges in optimal solutions to those in randomly created trees and those in MSTs. We present results for trees with $n = 15$ and $n = 20$ nodes. For each problem size, we create 100 random Euclidean OCST test instances as described in Sect. 2.2. For each OCST instance, we generate 10,000 random trees, calculate the MST, and determine an optimal (or near-optimal) solution according to Sect. 2.3.



(a) 15 nodes (b) 20 nodes

Figure 2: Distribution of γ for optimal solutions, random solutions, and MSTs for 100 randomly generated OCST instances with $n = 15$ and $n = 20$

Figure 2 presents results for $n = 15$ (Fig. 2(a)) and $n = 20$ (Fig. 2(b)). For the experiments, we considered all edges for different types of trees and plot the average distribution of γ for optimal solutions ("optimal"), random solutions ("random"), and MSTs. If the distribution is about uniform, orientation does not matter and all angles γ occur with the same probability in a tree. Studying the distribution for random trees shows that edges with larger γ are slightly preferred and occur more often in random solutions. Therefore, random solutions have a bias towards edges with high γ. For MSTs, orientation of the angles is of less importance and the distribution of γ is about uniform. In contrast, for optimal solutions, we have a non-uniform distribution of γ and edges with low γ occur more often. For example, for $n = 20$, on average about 20% of all edges of optimal solutions have an angle $\gamma \leq 10$ whereas only about 4% of all edges have an angle $\gamma > 80$. Therefore, the results of our experiments support the assumption that edges with a low γ are overrepresented in optimal solutions.

We see that optimal solutions have a bias towards edges with low γ. Therefore, we study whether heuristic optimization methods can make use of this observation by favoring edges with low distance weight *and* low γ. Consequently, in the following paragraphs, we study how to extend existing EA approaches such that they do not only favor edges with low distance weights but also edges that point towards the center of a tree.

3. PROBLEM-SPECIFIC EAS FOR OCST PROBLEMS

In this section, we extend the heuristic crossover operator of the edge-set encoding such that it does not only has a bias towards low-weighted edges but also towards low angles γ. Furthermore, we study and compare the performance of the resulting crossover operator.

We start by presenting the edge-set encoding and possible edge selecting strategies for crossover. In Sect. 3.2, we extend the crossover operator of edge-sets such that edges with low distance weight and angle γ are preferred. In Sect. 3.3, we perform experiments to analyze the trade-off between distance weight and orientation. Finally, Sect. 3.4 studies the performance of EAs using the extended version of edge-sets for different test instances.

3.1 Edge-sets

The edge-set (ES) encoding presented by Raidl et al. [9, 8] is a direct representation which directly represents trees as sets of edges. In direct representations, encoding-specific search operators are used to generate new solutions. ES operators are either heuristic considering the distance weights of edges for the constructions of the offspring or non-heuristic. We focus on the heuristic variants of the crossover operator which result in higher performance in comparison to non-heuristic variants [9, 16]. Initialization and mutation of ES is implemented as described in [9]. The following paragraphs present the functionality of the crossover operator and discusses different edge-selection strategies.

Crossover creates an offspring from two parental trees $T_1 = (G, E_1)$ and $T_2 = (G, E_2)$ by iteratively selecting edges from the set $F = (E_1 \cup E_2)$ of parental edges eligible for inclusion. Therefore, the offspring tree exists solely of parental edges. Different variants are distinguished according to the order in which parental edges are selected and inserted into the offspring. Julstrom and Raidl [4] presented four different edge-selection strategies which can be applied within crossover operators:

Random: In this crossover variant, which is denoted as KruskalRST [9], the edges of the offspring are randomly selected from F. [9] also presented an extended variant named KruskalRST* which includes all edges $(E_1 \cap E_2)$ in the offspring and, in a second step, randomly chooses the remaining edges from $F \setminus (E_1 \cap E_2)$. Both edge-selection strategy are about unbiased [9].

Greedy: This strategy selects in each step the edge with the lowest weight F until the offspring tree is a complete spanning tree. If edges have the same weight, one is selected at random. This strategy results into a bias towards low-weighted edges.

Inverse-weight-proportional: Edges are selected from F according to probabilities inversely proportional to their weight. Therefore, the strategy has a bias towards edges with low weight.

2-tournament: The edges are selected from F via a tournament selection (tournament size equals 2) with replacement. The weights of two edges are compared and the edge with the lower weight is included in the offspring. This heuristic crossover operator is used in [9] and shows a bias towards low-weighted edges and MSTs [16].

To strengthen the inheritance of common features from the parents, all strategies can be designed as a *-strategy. Then, all edges $(E_1 \cap E_2)$ are included in the offspring and the remaining edges are selected from $F \setminus (E_1 \cap E_2)$ using an edge-selection strategy.

3.2 Extended Crossover Operators for Edge-Sets

Based on the insights gained in Sect. 2.4, we extend edge-sets such that the knowledge about the orientation of edges in optimal solutions is considered. We do this by modifying the edge-selection strategies of the crossover operators of ES. We develop two different variants based on the greedy crossover strategy and on the 2-tournament crossover edge-selection strategy (see Sect. 3.1). To ensure high heritability,

both variants are designed as *-strategy where all edges common in both parents are directly transferred to an offspring. The missing edges are chosen from the remaining parental edges $E' = (E_1 \cup E_2) \setminus (E_1 \cap E_2)$ either using greedy crossover selection or 2-tournament crossover selection.

In contrast to existing approaches, the edges to be inserted into the offspring are not selected according to their distance weight, but according to the distance weight *and* the orientation of the edges. Therefore, we sort the edges according to

$$w'_{ij} = \alpha w_{ij}/w_{max} + (1-\alpha)\gamma_{ij}/\gamma_{max}, \qquad (3)$$

where w_{ij} is the distance weight of edge e_{ij}, γ_{ij} denotes the angle of edge e_{ij}, and $\alpha \in [0;1]$ is a parameter that controls the influence of w_{ij} and γ_{ij}. The distance weights as well as the angle of edges are normalized using the maximum values $w_{max} = max(w_{ij})$ $(i,j = 1,\ldots,n)$ and $\gamma_{max} = max(a_{ij})$ $(i,j = 1,\ldots,n)$. Therefore, $w'_{ij} \in [0,1]$.

After transferring all common edges to the offspring and sorting the remaining edges according to w'_{ij}, we add edges to the offspring by using either greedy or 2-tournament selection from Sect. 3.1 using w'_{ij} instead of the distance weight w_{ij}. Therefore, with lower w'_{ij}, the probability of an edge e_{ij} to be included in the offspring increases when using greedy or 2-tournament crossover (and increases when using inverse-weight-proportional crossover). Setting $\alpha = 1$, only the distance weights are considered and we obtain the crossover variants described in Sect. 3.1. For $\alpha \neq 1$, the orientation of edges influences the probabilities of the edges to be included in the offspring.

3.3 Balancing weight and orientation

This section studies how to set α. Therefore, we investigate how strong distance weights and orientation should be considered for the construction of offspring solutions. The goal is to identify a proper value for the parameter α that results in high and robust performance of ES crossover operators. For that purpose, we generate random trees using the greedy selection strategy (see Sect. 3.1) which iteratively selects edges from the set E of all edges. However, the strategy does not select edges according to w_{ij} but according to w'_{ij} (see (3)). We study the quality of the solutions created by the greedy strategy and measure the distance of these spanning trees to optimal solutions for different values of α. Furthermore, optimal solutions are obtained for the different problem instances as described in Sect. 2.3. We present results for 100 randomly generated OCST problem instances with 10, 15, and 20 nodes (see Sect. 2.2 for the specification of the randomly generated test problems).

Figure 3 shows the average distance $d_{g,opt}$ between trees T_g generated by greedy selection and the optimal solutions T_{opt} over α. The mean values are plotted as bold lines and the standard deviations are plotted as regular lines. The plots indicate similar results for all three problem sizes. For $\alpha = 1$, the greedy selection method only depends on the distance weights and thus creates MSTs. Therefore, the resulting $d_{g,opt}$ for $\alpha = 1$ are equivalent to the average distances of MSTs towards optimal trees. Better solutions with a lower distance towards the optimal tree can be obtained for $\alpha \approx 0.8$. This means that considering both orientation and distance weight results into better solutions for OCST problems. Finally, with lower values of α the average distance rises constantly until $\alpha = 0$. Therefore, considering

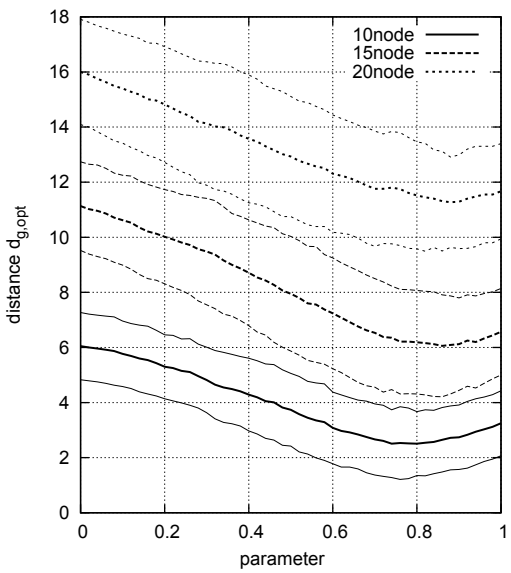


Figure 3: We show the average distance $d_{g,opt}$ between trees T_g generated by a greedy strategy and optimal solutions T_{opt} over α for randomly generated OCST instances with 10, 15, and 20 nodes. The greedy algorithm subsequently inserts edges in a tree starting with edges of low w'_{ij}.

only orientation and no distance weights ($\alpha = 0$) results into worse solutions than considering only distance weights and no orientation ($\alpha = 1$).

We introduced the orientation of edges as a new criteria that can be considered in the crossover operator of edge-sets. We performed experiments to analyze the trade-off between distance weight and orientation, which revealed that the influence of the distance weight should be higher than that of orientation. However, orientation matters since a combination of both yields better solutions in comparison to using only one criteria. We recommend setting $\alpha = 0.8$.

3.4 Performance of EAs using the extended operator

The final experiments compare the performance of EAs using different variants of the heuristic crossover operator. First, we study small problem instances where we determine optimal (or near-optimal) solutions as described in Sect. 2.3. Then, we examine larger problem instances with unknown optimal solutions.

3.4.1 Small problem instances

To study the performance of the crossover operator, we apply a simple steady-state EA using the extended crossover operators proposed in Sect. 3.2. For the experiments, we use non-heuristic initialization and mutation as described in [9]. A population consists of $N = 50$ individuals. In each search step, one offspring is created by crossover (crossover probability $p_c = 1$) and mutation (mutation probability $p_m = 1/l$). The two parents are selected at random. If the cost of the offspring is lower than the cost of the worst individual in the population ($(w(T_{off}) \leq \max(w(T_i)$ for $i \in \{0,\ldots,N-1\})$), the offspring replaces the worst individual in the population. We present results for OCST

instances of different sizes ($n = 8$, $n = 10$, $n = 12$, $n = 14$, $n = 16$, $n = 18$, $n = 20$), where the optimal solutions are calculated as described in Sect. 2.3. The EA terminates after $eval = 3,000$ fitness evaluations. For every problem size, 100 OCST instances are generated randomly and for each instance and configuration 20 EA runs are performed.

We compare three crossover operators which use the edge-selection strategies discussed in Sect. 3.1:

1. **random crossover (RX):** Non-heuristic crossover using KruskalRST*.
2. **greedy crossover ($GDOX$):** The edges are greedily selected according to w'.
3. **2-tournament crossover ($TDOX$):** The edges are selected via tournament selection and the quality of edges is measured by w'.

Table 1 lists the percentage P_{suc} of runs that find T_{opt}, the average costs $w(T_{best})$ of the best found solution, and the corresponding standard deviation σ. We show results for different values of α and the best values are printed bold. The results indicate a high performance of the EA with heuristic crossover operators. The EA with non-heuristic crossover (RX) has a high success probability P_{suc} for small problem sizes ($n \leq 14$). With larger problem sizes, EAs using the heuristic crossover TDOX perform better. Especially when setting $\alpha = 0.8$, high EA performance can be achieved. Comparing greedy (GDOX) and 2-tournament crossover (TDOX) shows that tournament selection performs better than the versions with greedy crossover throughout all problem sizes. Furthermore, the results reveal that the performance of the heuristic crossover variants increases, if the selection strategy does not consider only distance weights ($\alpha = 1$) but distance weights and orientation ($\alpha < 1$). Consequently, the lowest average costs $w(T_{best})$ are reached with TDOX and $\alpha = 0.8$.

Next, we study how the performance of the heuristic crossover depends on the distance $d_{opt,mst}$ between optimal solutions and MSTs. We use the same problem instances as in the experiment above. The plots in Fig. 4 show the percentage of runs P_{suc} which find the optimal solution over $d_{opt,mst}$ (left) and the gap $\frac{w(T_{best})-w(T_{opt})}{w(T_{opt})}$ (in percent) between the costs of the best found solution T_{best} and the optimal solution T_{opt} (right). We only present results for $n = 12$ since the results for other problem sizes are analogous. Furthermore, we show results only for $d_{opt,mst} \in \{1, ..., 7\}$ as in our experiments larger distances occur in only three instances. The findings confirm previous results that optimal solutions of OCST problems are similar to MSTs [16]. In our current study, we want to examine how EA performance depends on α for different $d_{opt,mst}$.

The results show that EA performance using TDOX with $\alpha = 1$ (this is the original crossover operator proposed by [9] considering only distance weights) and $\alpha = 0.9$ is maximal for OCST instances with low distance $d_{opt,mst}$. Since these problem instances are strongly biased towards MSTs and optimal solutions are only a few edges different from the MST, selection strategies that are based on the distance weights alone are very successful. However, with increasing distance $d_{opt,mst}$, the performance of EAs using TDOX with high α drops sharply. This findings confirm previous results that the heuristic variants of edge-sets fail for OCST problems

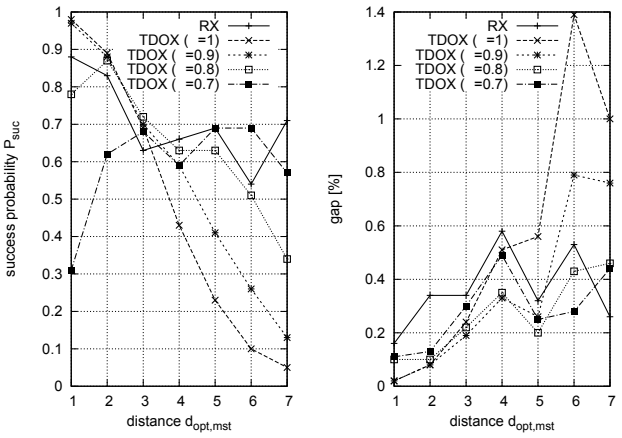


Figure 4: Comparison of EA performance using different crossover operators for randomly generated OCST instances ($n = 12$). We show the average success probability P_{suc} over $d_{opt,mst}$ (left) and the average gap $\frac{w(T_{best})-w(T_{opt})}{w(T_{opt})}$ (in %) over $d_{opt,mst}$ (right).

where the optimal solutions are more different from MSTs [16]. In contrast, EAs using heuristic crossover (TDOX) who consider the orientation of edges ($\alpha = 0.8$ or $\alpha = 0.7$) show high performance for problems with larger $d_{opt,mst}$. Comparing EAs using TDOX with $\alpha = 1$ to $\alpha = 0.8$ shows that setting α to 1 allows to find the optimal solution more often if it is only slightly different from the MST (which is no surprise since only distance weights are considered for selecting the edges of the offspring) but for higher $d_{opt,mst}$ setting $\alpha = 0.8$ is a much better choice. Furthermore, in all cases, the gap to the optimal solution is relatively small for the variants $\alpha = 0.8$ and $\alpha = 0.7$.

Overall, EA performance is high when using heuristic crossover with tournament selection. The variants with $\alpha = 1$ show only high performance if the distances between optimal solutions and MSTs are low, whereas the variants with $\alpha = 0.8$ perform well independently of the structure of the optimal solution. Therefore, our conjecture from the previous section can be confirmed and setting $\alpha = 0.8$ yields in high and robust EA performance.

3.4.2 *Larger problem instances*

In this section we study the performance of the heuristic crossover operators for larger OCST instances with unknown optimal solution. The performance is measured by the costs of the best found solution.

We use the same EA as in the previous experiments. A population consists of $N = 200$ individuals and each EA run is stopped after $eval$ fitness evaluations (see table 2). As EA performance usually increases with the number of fitness evaluations, we increase the number of fitness evaluations with larger n. We present results for OCST instances with different problem sizes ($n = 25$, $n = 50$, $n = 100$). For each problem size, we create 100 random instances. Due to computational restrictions, we investigate only 25 instances for $n = 100$. The same crossover operators as before are used and for each configuration and each problem instance we perform 20 EA runs.

Table 1: EA performance for small problem instances

n		MST	RX	GDOX				TDOX			
				α=1	α=0.9	α=0.8	α=0.7	α=1	α=0.9	α=0.8	α=0.7
8	P_{suc}	-	0.96	0.93	0.97	0.96	0.94	0.95	**0.98**	**0.98**	0.97
	$w(T_{best})$	973.26	901.42	901.95	901.06	901.32	901.64	901.48	901.11	**901.03**	901.12
	σ	-	1.55	0.74	0.31	0.56	0.7	0.77	0.44	0.37	0.55
10	P_{suc}	-	**0.85**	0.51	0.66	0.66	0.66	0.7	0.79	0.82	0.8
	$w(T_{best})$	1650.37	1482.49	1487.69	1484.61	1485.08	1485.53	1483.36	**1482.04**	1482.05	1482.64
	σ	-	5.88	5.16	3.9	4.03	3.85	4.43	3.41	2.79	3.18
12	P_{suc}	-	**0.67**	0.21	0.31	0.45	0.42	0.41	0.52	0.64	0.64
	$w(T_{best})$	2499.57	2212.42	2234.34	2221.06	2215.66	2218.28	2217.05	2211.51	**2209.34**	2209.79
	σ	-	17.95	14.52	10.22	8.46	9.61	12.23	9.21	7.27	6.83
14	P_{suc}	-	**0.38**	0.07	0.15	0.18	0.18	0.17	0.28	0.36	**0.38**
	$w(T_{best})$	3644.15	3171.37	3216.2	3192.44	3179.24	3188.28	3180.51	3167.86	**3161.19**	3162.88
	σ	-	43.03	25.82	21.33	17.96	21.02	24.29	17.94	14.77	15.58
16	P_{suc}	-	0.08	0.01	0.03	0.06	0.06	0.03	0.08	0.14	**0.17**
	$w(T_{best})$	4860.12	4221.45	4286.16	4231.47	4213.01	4234.03	4223	4189.25	**4178.67**	4180.18
	σ	-	80.71	44.13	35.59	31.65	39.66	44.97	31.3	27.77	26.27
18	P_{suc}	-	0.01	0	0.02	0.02	0.03	0	0.03	0.05	**0.06**
	$w(T_{best})$	6338.01	5481.18	5529.07	5452.04	5426.05	5472.88	5432.22	5379.33	**5360.58**	5368.14
	σ	-	149.98	69.78	55.01	56.28	78.01	75.16	53.45	47.71	52.7
20	P_{suc}	-	0	0	0	0	0.01	0	0	0.01	**0.02**
	$w(T_{best})$	8124.94	7089.84	7050.09	6948.55	6927.66	6989.15	6933.24	6852.88	**6806.16**	6811.12
	σ	-	241.08	93.49	80.94	93.13	121.5	105.38	84.92	77.3	79.8

Table 2: Number of fitness evaluations *eval* for large problem instances

n	25	50	100
eval	5,000	20,000	80,000

Table 3 presents the average costs $w(T_{best})$ of the best found solution T_{best} over the 20 EA runs. Additionally, the standard deviation σ of the costs and the running time t_{CPU} (in seconds) of one run is shown. The results indicate a high performance of EAs with heuristic crossover especially if the orientation of the edges is included in the selection strategy. EAs using TDOX with $\alpha = 0.8$ perform best for all problem sizes since the average total cost are always lowest. Furthermore, heuristic variants clearly outperform non-heuristic ones. Comparing the different heuristic configurations reveals the superiority of the variants with 2-tournament selection.

The plots in Fig. 5 summarize the results. We show the average gap $\frac{w(T_{mst})-w(T_{best})}{w(T_{mst})}$ (in percent) between the best found solution T_{best} and the MST over the problem size n. A large gap indicates high EA performance. We have chosen the MST as reference since the MST is often a high-quality solution for OCST problems. The plots show that solutions found by EAs using TDOX and $\alpha = 0.8$ perform best and result in a largest gap towards the MST. Non-heuristic variants (RX) show low performance.

The results for the CPU times (see Table 3) are analogously to the results for the solution quality. Using heuristic crossover does not increase the required CPU time. The variants with greedy edge selection slightly need more time in comparison to the variants with tournament selection. This is due to the fact that with tournament selection no sorting of edges is required but only comparisons between pairs of edges are performed.

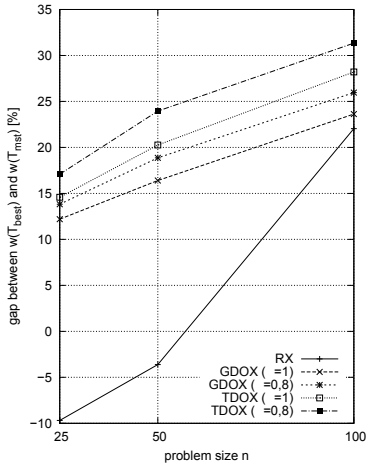


Figure 5: EA performance using different crossover variants for randomly generated OCST instances. The plots show the average gap between the cost of the best found solution and the MST over the problem size n. The larger the gap, the higher the EAs performance.

Table 3: EA performance for large problem instances

n		MST	RX	GDOX				TDOX			
				α=1	α=0.9	α=0.8	α=0.7	α=1	α=0.9	α=0.8	α=0.7
25	$w(T_{best})$	13049.08	14312.49	11457.06	11175.14	11244.56	11553.89	11146.29	10907.53	**10818.91**	10910.77
	σ	-	687.29	133.46	127.56	168.08	206.66	136.88	107.55	97.49	117.89
	t_{cpu}	-	0.57	0.56	0.55	0.55	0.54	0.57	0.57	0.56	0.56
50	$w(T_{best})$	59415.65	61559.62	49668.48	47930.97	48206.69	49554.99	47378.36	45605.4	**45181.33**	45602.33
	σ	-	5896.98	672.24	673.99	871.51	1077.2	649.56	444.76	442.07	554.85
	t_{cpu}	-	8.59	8.84	8.5	8.07	7.83	8.54	8.26	7.94	7.72
100	$w(T_{best})$	268781.16	209537.14	205277.83	196011.25	199009.46	205574.85	192976.41	185458.41	**184532.28**	186093.21
	σ	-	17877.01	3214.57	3068.94	4539.86	5525.38	2501.16	1498.26	1544.91	1819.92
	t_{cpu}	-	173.11	197.72	183.34	173.58	168.19	185.28	175.43	167.93	162.25

The results confirm the findings of the previous experiments. The performance of EAs with heuristic crossover can be increased by considering the orientation of the edges when selecting the edges for the offspring ($\alpha = 0.8$). Heuristic variants using a tournament selection strategy outperform greedy selection strategies.

4. SUMMARY AND CONCLUSIONS

This work studies the OCST problem and shows that edges in optimal solutions are not uniformly oriented but edges pointing towards the center of a tree occur with higher probability. Thus, the performance of EA search operators can be improved by using this knowledge to create high-quality solutions.

To show how this observation can be considered for the design of efficient EAs for the OCST problem, the edge-set encoding is used and extended. The edge-set encoding is a direct representation for trees which represents trees directly as sets of edges. Offsprings are created by iteratively selecting edges from the parental trees. Unlike existing heuristic edge-selection strategies, the proposed GDOX and TDOX crossover operators consider distance weights of edges *and* orientation of edges. The edges of parental trees are sorted according to their weight as well as orientation and edges that have low weight and point towards the center of the tree are included with higher probability in the offspring. We examine the performance of the extended crossover operators GDOX and TDOX for various test problems and find that crossover operators of the edge-sets using both criteria, weight and orientation, outperform existing approaches which only consider distance weights. Especially crossover using tournament selection as edge-selection strategy are very effective and efficient throughout all test instances.

The results presented in this work suggest to use heuristic crossover operators which prefer edges that point towards the center of a tree and have low distance weights to improve the performance of EAs for the OCST problem. By considering also the orientation of edges, the strong bias of existing heuristic crossover operators towards MST-like solution can be reduced. Furthermore, crossover strategies with tournament edge-selection strategies should be preferred in comparison to greedy variants, because of their higher efficiency and robustness.

Future work will address how the knowledge about the orientation of edges can be used in other encodings, e.g. NetDir [10]. Another promising area is the development of problem-specific mutation and initialization operators for edge-sets which consider the orientation of the edges.

5. REFERENCES

[1] T. Fischer and P. Merz. A memetic algorithm for the optimum communication spanning tree problem. In T. Bartz-Beielstein, M. J. B. Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics, 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007, Proceedings*, volume 4771 of *LNCS*, pages 170–184. Springer, 2007.

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

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

[4] B. A. Julstrom and G. R. Raidl. Weight-biased edge-crossover in evolutionary algorithms for two graph problems. In *SAC '01: Proceedings of the 2001 ACM symposium on Applied computing*, pages 321–326, New York, NY, USA, 2001. ACM.

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

[6] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234, New York, NY, USA, 1988. ACM Press.

[7] D. Peleg and E. Reshef. Deterministic polylog approximation for minimum communication spanning trees. In *ICALP '98: Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 670–681, London, UK, 1998. Springer-Verlag.

[8] G. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 104–111, Piscataway, NJ, 2000. IEEE Service Center.

[9] G. R. Raidl and B. A. Julstrom. Edge sets: an effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, June 2003.

[10] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Heidelberg: Springer, 2 edition, 2006.

[11] F. Rothlauf. *Design and Application of Metaheuristics*. Universität Mannheim, Habilitationsschrift, 2007.

[12] F. Rothlauf, D. E. Goldberg, and A. Heinzl. Network random keys: a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computaton*, 10(1):75–97, 2002.

[13] F. Rothlauf and A. Heinzl. On optimal solutions for the optimal communication spanning tree problem. Technical report, Department of Information Systems 1, University of Mannheim, 2003.

[14] P. Sharma. Algorithms for the optimum communication spanning tree problem. *Annals of Operations Research*, 143(1):203–209, 2006.

[15] S.-M. Soak. A new evolutionary approach for the optimal communication spanning tree problem. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E89-A(10):2882–2893, 2006.

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

[17] B. Y. Wu and K.-M. Chao. *Spanning Trees and Optimization Problems*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2004.

[18] B. Y. Wu, K.-M. Chao, and C. Y. Tang. Approximation algorithms for some optimum communication spanning tree problems. *Discrete Appl. Math.*, 102(3):245–266, 2000.

[19] B. Y. Wu, K.-M. Chao, and C. Y. Tang. A polynomial time approximation scheme for optimal product-requirement communication spanning trees. *J. Algorithms*, 36(2):182–204, 2000.