On the Bias and Performance of the Edge-Set Encoding

Franz Rothlauf, *Member, IEEE*

***Abstract*— The edge-set encoding of trees directly represents trees as sets of their edges. Non-heuristic operators for edge-sets manipulate trees' edges without regard for their weights, while heuristic operators consider edges' weights when including or excluding them. In the latter case, the operators generally favor edges with lower weights, and they tend to generate trees that resemble minimum spanning trees. This bias is strong, which suggests that EAs that employ heuristic operators will succeed when optimum solutions resemble MSTs but fail otherwise.**

The one-max tree problem is a scalable test problem for trees where the optimum solution can be pre-defined. Heuristic operators for edge-sets fail when optimum solutions are random trees or stars. Similarly, for the optimal communication spanning tree (OCST) problem, heuristic operators are efficient only for problem instances where optimal solutions are slightly different from MSTs. In contrast, for both problems the performance of non-heuristic operators is approximately independent of the type of the optimal solution.

Therefore, heuristic operators for edge-sets should be used only if optimal solutions closely resemble MSTs. If optimal solutions have low or no bias towards MSTs, heuristic operators for edge-sets fail, and non-heuristic operators should be preferred.

***Index Terms*— edge-sets, tree representation, bias, one-max tree problem, optimal communication spanning tree problem.**

I. 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 such as the degree-constrained minimum spanning tree (D-MST) problem [1]–[3] and the optimal communication spanning tree (OCST) problem [4]–[6].

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: either indirect or direct representations. 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). Examples are NetKeys [7], the link-and-node-biased encoding [8], determinant factorization [9], and Prüfer numbers [10], [11]. In contrast, direct representations encode a tree as a set of edges and apply search operators directly to this set. Therefore, no representation is necessary. Instead, tree-specific search operators must be developed as standard search operators can no longer be used. Examples are edge-set encoding [3] and NetDir encoding [12, Chap. 7.1]. Raidl and Julstrom [3] proposed different variants of the operators for edge-sets: heuristic operators which consider the edges' weights, and non-heuristic operators. The authors conclude that "tests on two sets of hard degree-constrained MST problem instances indicate the superioriy of edge sets, particularly when the variation operators implement edge-cost-based heuristics, to several other codings of spanning trees" [3, p. 238].

A bias of a direct encoding means that the encoding-specific initialization, crossover, or mutation operators prefer a specific type of solution and push a population in this direction. As the heuristic operators for edge-sets prefer low-weighted edges, they are expected to show a bias towards MSTs [3], [13]. The purpose of this paper is

1) to investigate thoroughly the bias of the operators for edge-sets, and
2) to study how this bias influences EA performance. It is of special interest how EA performance depends on the properties of the optimal solution.

The performance of edge-sets is studied for the one-max tree problem and the OCST problem. The one-max tree problem is a test problem where optimal solutions can be pre-defined. This allows us to study how EA performance depends on the structure of the optimal solution. For the OCST problem, we study how EA performance depends on the distance between the optimal solution and an MST. Since optimal solutions are similar to MSTs [14], EAs using heuristic operators for edge-sets are expected to show high performance.

The main findings of this paper are:

1) Heuristic crossover and mutation operators for edge-sets favor MST-like trees. Crossover's bias is stronger than that of mutation.
2) EAs that use heuristic operators for edge-sets perform well only on problem instances whose optimum solutions are similar to MSTs.
3) EAs with heuristic operators show good performance on instances of the OCST problem with random edge weights, but not on instances with Euclidean weights. Optimum solutions for the former resemble MSTs; those for the latter in general do not.
4) When the structure of optimum solutions is not known, heuristic operators are not appropriate.

The next section gives a brief overview of tree representations. Then, we focus on the functionality (Sect. III) and bias (Sect. IV) of edge-sets. Sections V and VI examine how edge-sets' performance depends on the type of optimal solution for the one-max tree problem and the OCST problem, respectively. The paper ends with concluding remarks.

Manuscript Submission.
The author is with the University of Mainz, Germany.

II. Representations for Trees

Tree optimization problems are common combinatorial optimization problems [15], [16] which are often $\mathcal{NP}$-hard [17]. Therefore, only few efficient algorithmic methods are available, and heuristic optimization methods are often used. Since the performance of such methods strongly depends on the encoding, there is a variety of tree encodings.

Palmer [5] compared different types of tree encoding and developed a new representation, the *link and node biased* (LNB) encoding. EAs using the LNB encoding showed good results in comparison with a greedy star search heuristic [5, Chap. 5].

The *characteristic vector* (CV) encoding [6], [18]–[20] represents a tree as a list of $n(n-1)/2$ binary values. Infeasible solutions (non-trees) can occur which are usually repaired. The CV encoding shows good performance only when used for trees on a small number of nodes [12, Sect. 6.3].

Abuali et al. [9] introduced *determinant factorization*. This representation is based on the in-degree matrix of the original graph, and each factor represents a spanning tree if the determinant corresponding to that factor is equal to one. Tests indicated EA performance similar to the LNB encoding.

Weighted encodings, such as weighted encoding [21], *NetKey* encoding [7], and LNB encoding as well as variants of it [22] represent a tree using a list of continuous weights. The weights define an order of the edges and the represented tree is constructed from this ordered list of edges using tree construction algorithms like Kruskal's or Prim's algorithm.

Cayley codes [23] such as *Prüfer numbers* [10] describe a one-to-one mapping between spanning trees on n nodes and strings of $n-2$ node labels. Other Cayley codes have been proposed by Neville [24] (*Neville II* and *Neville III*), Deo and Micikevicius [25] (*D-M code*), and Picciotto [26] (*Blob Code*, *Happy Code*, and *Dandelion Code*). [27] presented a unified approach for Cayley codes which is based on the definition of node pairs reducing the coding problem to the problem of sorting these pairs into lexicographic order. The *locality* of an encoding describes how well small changes in the genotype correspond to small changes in the phenotype. Because of problems with low locality, Prüfer numbers lead to low EA performance [11], [28]. The locality of the Blob code is higher than Prüfer numbers resulting in higher EA performance [29], [30]. Paulden and Smíth [31] extended this work and showed that a single mutation to a Dandelion string leads to, at the most, five edge changes in the corresponding tree, whereas the Prüfer number encoding has no fixed locality bound.

Recently, several *direct representations* for trees have been proposed; a direct representation for trees [32], the edge-set encoding studied in this paper, and the *NetDir* encoding [12]. For direct representations, there is no additional mapping from the phenotype space to a different genotype space, but tree-specific search operators are applied directly to trees.

III. Edge-Set Encoding

Edge-sets directly represent trees as sets of edges. Therefore, encoding-specific initialization, crossover, and mutation operators are necessary [3].

A. Edge-Set Encoding without Heuristics

1) Initialization: Raidl and Julstrom [3] proposed three different initialization strategies: PrimRST, RandWalkRST, and KruskalRST. PrimRST slightly 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, [3] recommended use of KruskalRST which is based on Kruskal's algorithm. When constructing a tree, KruskalRST chooses edges (i,j) not according to their weights w_{ij} but randomly. It has a small bias towards stars (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) 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 $G_{cr} = (V, E_1 \cup E_2)$. The crossover operator has high *heritability* [30], [33] as in the absence of constraints, only parental edges are used to create the offspring. Crossover becomes more sophisticated for constrained MST problems as then the RST algorithm can create infeasible tree from $G_{cr} = (V, E_1 \cup E_2)$.

Raidl and Julstrom [3] distinguished two different recombination operators: the variant previously described is denoted as KruskalRST crossover. The second variant is denoted as KruskalRST* crossover. For 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 [3] indicate a better performance of KruskalRST* for the D-MST problem.

3) Mutation: The two variants of the mutation operator randomly replace one edge in a tree. The first variant randomly chooses one edge that is not present in T and includes it. 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.

B. Edge-Set Encoding with Heuristics

Because of the assumption that in weighted tree optimization problems optimal solutions often prefer low-weight edges, Raidl and Julstrom [3] presented heuristics operators for edge-sets that consider edge weights w_{ij} when constructing a tree.

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 MSTs, which becomes lower with increasing k. k 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) 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 unconstrained, this procedure always yields a spanning tree.

3) Heuristic Mutation: The heuristic mutation operator is based on mutation by "insertion before deletion". A weight-based rank is assigned to all edges $(i, j) \in E$. The rank one is assigned to the edge with the lowest weight. To favor low-weighted edges, the edge that is inserted 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 zero mean and standard deviation βn and $m = n(n-1)/2$ denotes the number of available edges. β is a parameter that controls the bias towards low-weighted edges. If a chosen edge already exists in T, it is discarded and the selection is repeated.

IV. Bias of Edge-Set Encoding

A variation operator is unbiased if it does not tend to produce solutions of any particular structure [12]. Therefore, application of an unbiased search operator alone does not modify the statistical properties of a population but allows a uniform, non-directed search. A biased operator should only be used if it is known a priori that the optimal solution of the underlying optimization problem is similar to the operator's bias [34]. In contrast, unbiased operators should be used if no a priori problem-specific knowledge is available.

We study the bias of edge-sets for random trees and consider two possibilities for the edge weights w_{ij}:

- **Random weights:** The real-valued weights w_{ij} are generated randomly and are uniformly distributed in $]0, 10]$.
- **Euclidean weights:** The nodes are randomly placed on a 10x10 grid. The weight w_{ij} is the Euclidean distance between the nodes i and j.

As the weights w_{ij} are randomly created and $w_{ij} \neq w_{kl}$, $\forall ij \neq kl$, we can assume that there is a unique MST for every random problem instance. T is the MST if $c(T) \leq c(T')$ for all other spanning trees T', where $c(T) = \sum_{(i,j) \in T} w_{ij}$. For properties of MSTs, we refer to [35] who empirically showed that the relative frequencies with which an edge of a specific rank appears in the MST can be closely approximated by exponential functions [35, Sect. II]. The similarity between two spanning trees T_i and T_j can be measured using the distance $d_{ij} \in \{0, 1, \dots, n-1\}$ 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 edge (u, v) exists in T_i and 0 if not.

A. Initialization

Raidl and Julstrom [3] studied the bias of different initialization methods and found KruskalRST to be slightly biased towards stars. As the bias is sufficiently small and because of its lower running time it is preferred in comparison with RandWalkRST as well as with PrimRST, which shows a stronger bias towards stars.

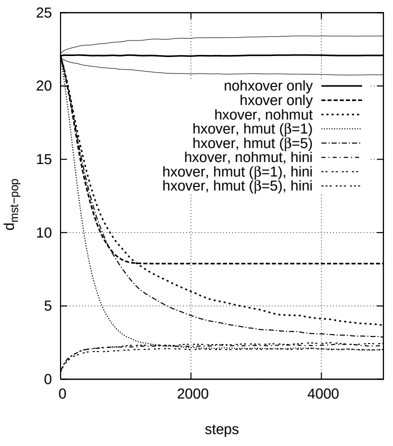
Table I shows the average distances $d_{rand,mst}$ between MSTs and randomly generated trees T_{rand} (the standard deviations are shown in brackets). For each problem instance (250 of each type) we generated 1,000 random solutions T_{rand} using either an unbiased encoding (Prüfer numbers), non-heuristic KruskalRST (Sect. III-A.1), or the heuristic initialization (Sect. III-B.1). α was set either to $\alpha = 1.5$ as recommended in [3], or to its maximum $\alpha = (n-1)/2$ which results in the lowest possible bias. The results confirm that KruskalRST is not biased towards MSTs ([3] showed that KruskalRST shows a small bias towards stars). Furthermore, as expected, heuristic initialization shows a strong bias towards MSTs even for the maximum value of α. Of interest is the very strong bias towards MSTs when using $\alpha = 1.5$. For this parameter setting, randomly generated solutions share on average 8.8 (out of nine possible) edges with an MST for 10 node problems and up to approximately 195 (out of 199) edges with an MST for 200 node problems (assuming random w_{ij}).

TABLE I
Mean and standard deviation of Distances $d_{rand,mst}$ between random trees and MSTs

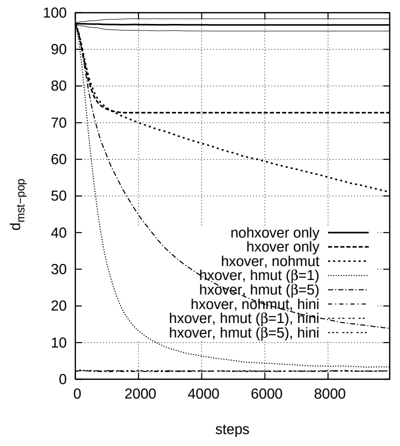
n	weights	unbiased	KruskalRST	heuristic initialization	
			non-heuristic	$\alpha = 1.5$	$\alpha = (n-1)/2$
10	Euclidean	7.20 (0.06)	7.20 (0.04)	0.44 (0.19)	3.80 (0.08)
	random		7.20 (0.07)	0.20 (0.13)	3.74 (0.08)
20	Euclidean	17.10 (0.04)	17.10 (0.04)	1.06 (0.28)	12.02 (0.14)
	random		17.10 (0.04)	0.42 (0.22)	12.09 (0.08)
25	Euclidean	22.08 (0.04)	22.07 (0.04)	1.37 (0.31)	16.70 (0.16)
	random		22.08 (0.04)	0.55 (0.26)	16.76 (0.08)
100	Euclidean	97.02 (0.04)	97.02 (0.04)	5.98 (0.67)	87.89 (0.18)
	random		97.02 (0.05)	2.12 (0.60)	88.22 (0.87)
200	Euclidean	197.03 (0.04)	197.02 (0.04)	11.99 (0.92)	176.45 (0.29)
	random		197.00 (0.04)	3.93 (0.69)	177.25 (0.12)

B. Recombination

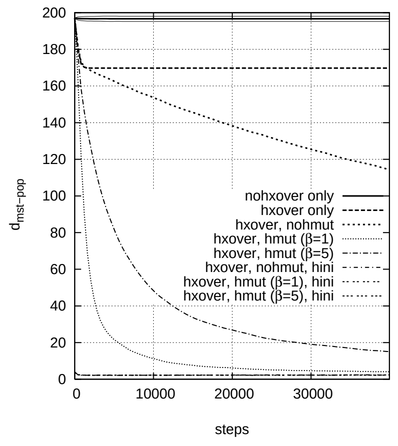
To investigate how strong edge-sets' crossover operators are biased towards MSTs, we randomly generate an initial population of either $\mu = 50$ or $\mu = 200$ individuals. Then, in each search step, one randomly created offspring replaces a randomly chosen individual in the population. The offspring is created either by recombination alone (with probability $p_c = 1$), or by recombination ($p_c = 1$) and mutation (with mutation probability $p_m = 1/n$). We present results for the following search operators:



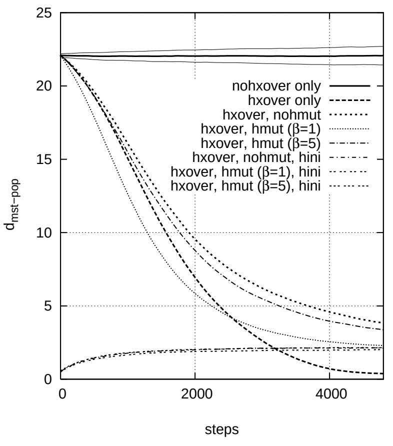
(a) $n = 25$, $\mu = 50$



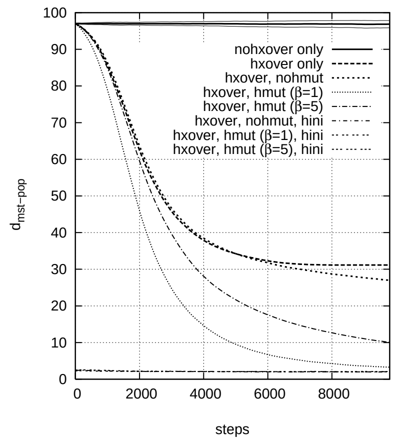
(b) $n = 100$, $\mu = 50$



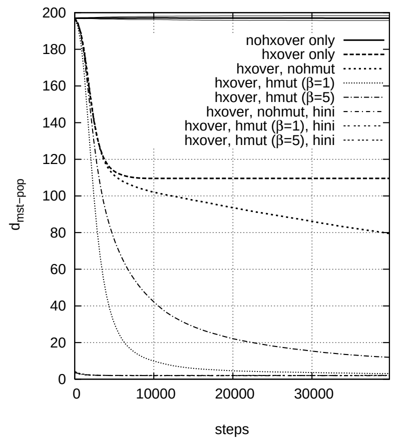
(c) $n = 200$, $\mu = 50$



(d) $n = 25$, $\mu = 200$



(e) $n = 100$, $\mu = 200$



(f) $n = 200$, $\mu = 200$

Fig. 1. The plots show the mean of the distance $d_{mst-pop}$ between a population of either $\mu = 50$ (top) or $\mu = 200$ (bottom) randomly generated individuals towards MSTs over the number of search steps. In each search step, two random parents are selected to form an offspring, which randomly replaces one solution of the population. No selection operator is used. The heuristic crossover operator shows a strong bias towards MSTs.

- **nohxover only**: non-heuristic KruskalRST* xover (Sect. III-A.2) with p_c=1, no mutation (p_m=0), and non-heuristic initialization with KruskalRST (Sect. III-A.1).
- **hxover only**: heuristic xover with $p_c = 1$ (Sect. III-B.2), no mutation ($p_m = 0$), and non-heuristic initialization.
- **hxover, nohmut**: heuristic xover ($p_c = 1$), non-heuristic mutation with $p_m = 1/n$ and "insertion before deletion" (Sect. III-A.3), and non-heuristic init.
- **hxover, hmut** (**$\beta = 1$**): heur. xover ($p_c = 1$), heuristic mutation ($p_m = 1/n$) with $\beta = 1$ (Sect. III-B.3), and non-heur. initialization.
- **hxover, hmut** (**$\beta = 5$**): heur. xover ($p_c = 1$), heur. mutation ($p = 1/n$, $\beta = 5$), and non-heur. init.
- **hxover, nohmut, hini**: heur. xover ($p_c = 1$), non-heur. mutation with $p_m = 1/n$ ("insertion before deletion"), and heur. init. with $\alpha = 1.5$ (Sect. III-B.1).
- **hxover, hmut (**β**=1), hini**: heur. xover (p_c=1), heur. mutation ($p = 1/n$, $\beta = 1$), and heur. init. ($\alpha = 1.5$).
- **hxover, hmut (**β**=5), hini**: heur. xover (p_c=1), heur. mutation ($p = 1/n$, $\beta = 5$), and heur. init. ($\alpha = 1.5$).

We show experimental results for different problem sizes (25, 100, and 200) and up to 40,000 search steps. After each search step, we measure the average distance $d_{mst-pop} = 1/\mu \sum_{i=1}^{\mu} d_{i,mst}$ of the individuals T_i ($i \in \{0, \ldots, \mu - 1\}$) in the population from the MST. As no selection operator is used, no selection pressure pushes the population to high-quality solutions.

We perform this experiment on 250 randomly generated tree instances of different n with random w_{ij}. For every instance we performed 10 runs with randomly chosen initial populations. Results for Euclidean w_{ij} are equivalent but omitted because of space limitations. Figure 1 shows the mean of $d_{mst-pop}$ over the number of search steps. For increased clarity, standard deviations are generally omitted; as examples we plot the standard deviations for "nohxover only". The plots compare the different crossover operators (non-heuristic KruskalRST* xover and heuristic xover) either alone or combined with mutation and/or initialization operators.

The results confirm previous findings [36] that crossover without heuristics has no significant bias towards an MST since it does not modify the statistical properties of the population ($d_{mst-pop}$ remains about constant over the number of search steps). However, we expect (as for the non-heuristic initialization operator) a minor bias towards stars [3, Sect. IIIb]. The figures extend previous work showing that heuristic crossover operators have a strong bias towards MSTs. Applying heuristic crossover operators alone (hxover only) pushes the population towards MSTs. After some search steps (for example, after $1,000$ search steps for $n = 25$ and $\mu = 50$), the population is fully converged and cannot move any more. If the population size is increased to $\mu = 200$, the diversity remains high enough and the population fully converges to MSTs (compare for example Figs. 1(a) and 1(d)). The plots for $n = 100$ and $n = 200$ show the same behavior.

The problem of low diversity can be amended when combining heuristic crossover with heuristic or non-heuristic mutation. Then, continuously new edges are introduced into the population and no premature convergence occurs. Consequently, with a higher number of search steps, the population keeps moving towards an MST. However, the population does not reach an MST as the mutation operators continuously insert new edges. Comparing variants of heuristic mutation operators (hxover, hmut (β=1) and hxover, hmut (β=5)) shows that with lower β the bias increases and the population converges faster towards MSTs.

As observed in Table I, heuristic initialization results in a strong bias towards MSTs. Combining crossover with heuristic initialization allows the population to recover from the strong initial bias and $d_{mst-pop}$ converges to the same values as when using non-heuristic initialization (see plots for "hxover, nohmut, hini", "hxover, hmut (β=1), hini", and "hxover, hmut (β=5), hini").

Summarizing the results, non-heuristic crossover results in no significant bias towards MSTs. In contrast, the bias of heuristic crossover towards MSTs is strong and after a few thousand search steps the average distance of a population from the MST is low (e.g. after 4000 search steps a population of 200 individuals encoding a tree with $n = 25$ nodes shares on average more than 21 edges with the MST).

C. Mutation

Finally, we study the bias of the mutation operator for 250 random tree instances ($n \in \{25; 100; 200\}$) with random w_{ij}. We create a random initial population of size μ using either non-heuristic or heuristic initialization. Then, in every search step a randomly chosen individual is mutated once using either the non-heuristic "insertion-before-deletion" mutation, or the heuristic variant with different β. The mutated offspring replaces a randomly selected individual in the population. Mutation only and no selection is used. With lower β, heuristic mutation prefers edges with lower weights.

Figure 2 shows the mean of $d_{mst-pop}$ over the number of search steps. The results confirm our assumption that the non-heuristic mutation operator is approximately unbiased, whereas the heuristic mutation is biased towards the MST. As expected, the bias increases with lower β. Because of the continuous insertion of new edges, a population does not converge completely towards an MST but $d_{mst-pop}$ remains constant after some time. Although heuristic initialization leads to a strong initial bias (see Table I), the mutation operator allows the population to recover and to converge to the same $d_{mst-pop}$ as for the non-heuristic initialization. Using a larger population ($\mu = 200$) delays convergence.

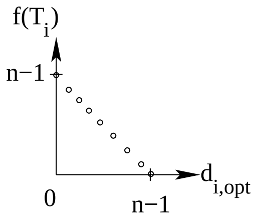


Fig. 3. One-max tree problem

V. Performance of Edge-Sets for the One-Max Tree Problem

We study the performance of edge-set encoding for the *one-max tree problem* [7]. This easy test problem for trees allows us to examine how the performance of EAs depends on the type of optimum solution. Raidl and Julstrom [3] showed for the one-max tree problem that all non-heuristic crossover operators PrimRST, KruskalRST, and RandWalkRST are nearly equally effective, and that the *-variants (e.g. KruskalRST*) should be preferred in comparison with the non-* variants (e.g. KruskalRST). We extend this work and also study the performance of the heuristic operators.

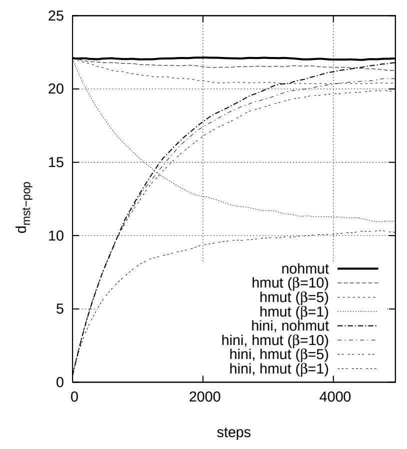
A. The One-Max Tree Problem

The one-max tree problem defines an optimization problem where the optimal solution is either chosen randomly or by hand [7]. The structure of this tree can be determined: it can be for example an MST, a star, or any other arbitrary tree.

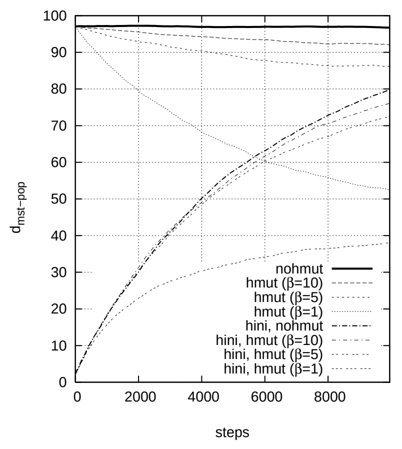
The fitness $f(T_i)$ of a tree T_i is defined as the number of edges it has in common with the best solution T_{opt} and is calculated as $f(T_i) = n - 1 - d_{i,opt}$, where $d_{i,opt}$ is the distance between T_i and T_{opt} (see Fig. 3). We assume in all experiments that edges exist between any pairs of nodes.

B. Experimental Results

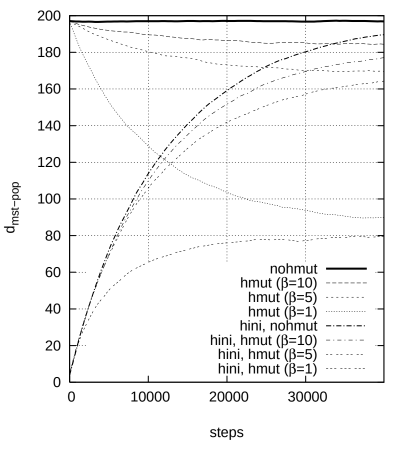
The EA is conventional (steady-state) with a population of μ individuals and a $(\mu + 1)$-selection strategy. In each search step, two individuals of the population are chosen randomly and are recombined to form an offspring. The offspring T_{off} is mutated with probability p_{mut}. If the fitness of T_{off} is equal or higher than the fitness of the worst individual in the population $(f(T_{off}) \geq \min(f(T_i))$, for $i \in \{0, \ldots, \mu - 1\})$, it replaces the worst individual. Otherwise, it is discarded. We present results for one-max problems of different sizes ($n \in \{10, 25, 50, 100, 150, 200\}$), where T_{opt} is either a randomly chosen tree (T_{rand}), a star (T_{star}) with randomly chosen center i ($i \in \{0, \ldots, n-1\}$), or an MST (T_{mst}). The EA is stopped after $eval$ search steps. Since EA performance depends on the number of search steps, we increase $eval$ for larger n (Table II). In addition to the search operators described in Sect. IV-B, we present results for the following search operators:



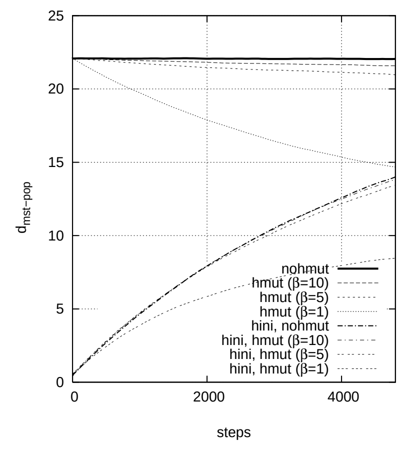
(a) $n = 25$, $\mu = 50$



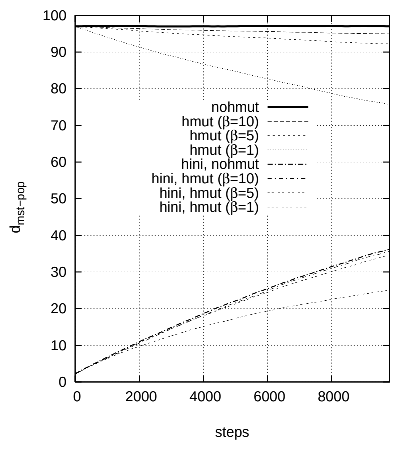
(b) $n = 100$, $\mu = 50$



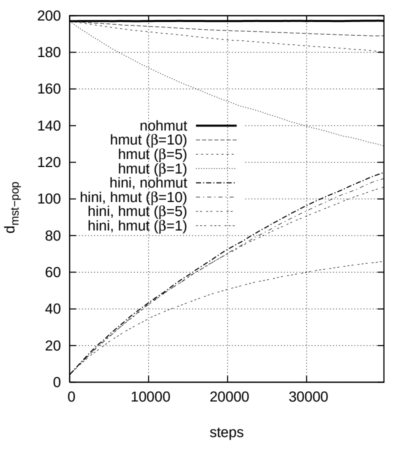
(c) $n = 200$, $\mu = 50$



(d) $n = 25$, $\mu = 200$



(e) $n = 100$, $\mu = 200$



(f) $n = 200$, $\mu = 200$

Fig. 2. The plots show the mean of the distance $d_{mst-pop}$ of a population of μ randomly generated individuals from the MST over the number of search steps for different problem sizes n. In each search step, one randomly chosen individual is mutated once and replaces a randomly chosen individual in the population. No selection operator is used. The heuristic mutation operator is biased and the bias increases with lower β.

- **nohxover, nohmut**: non-heuristic KruskalRST* crossover ($p_c = 1$), non-heuristic mutation ($p_m = 1/n$), and non-heuristic initialization (KruskalRST).
- **NetKey**: NetKey encoding with uniform crossover ($p_c = 1$), mutation ($p_m = 1/n$), and random initial population.

n	10	25	50	100	150	200
eval	500	2,500	10,000	40,000	80,000	160,000

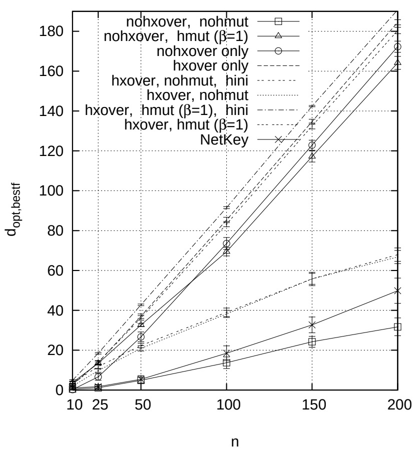
TABLE II

NUMBER OF SEARCH STEPS $eval$ FOR ONE-MAX TREE PROBLEMS

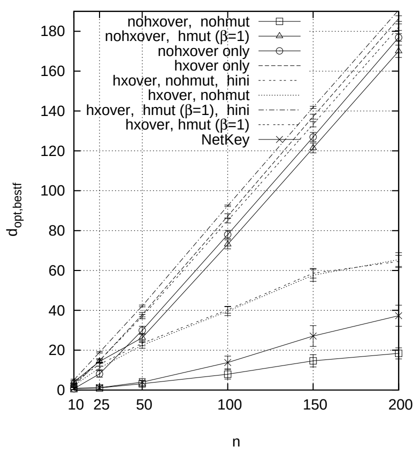
The NetKey encoding [7] (see also Sect. II) encodes a tree as a continuous vector r of length $n(n-1)/2$, where $r_{ij} \in [0, 1]$. The r_{ij} are used as distance weights and the represented tree is constructed from the r_{ij} by Kruskal's or Prim's algorithm. Construction of the tree from the r_{ij} can be done in time that is $O(n^2)$ (Prim's algorithm) or $O(n^2 \log(n))$ (Kruskal's algorithm). NetKeys are nearly uniformly redundant [12, Sects. 6.5.4 and 8.1.1], as the use of Kruskal's or Prim's algorithm introduces a small bias [3]. Since the bias is minor, the performance of EAs using NetKeys is approximately independent of the structure of the optimal solution [12, Sect. 8.1]. Therefore, NetKeys are used as a representative example of an unbiased encoding. We use standard uniform crossover, standard mutation that assigns a random value to r_{ij} with probability p_m, and initialization that assigns random values to the r_{ij}.

Figure 4 shows the mean and standard deviation of the distance $d_{opt,bestf}$ between the optimal solution T_{opt} and the best solution T_{bestf} that was found after $eval$ (Table II) search steps. We plot results for μ=50 (top) and μ=200 (bottom). T_{opt} is either a random tree ($T_{opt} = T_{rand}$), a star ($T_{opt} = T_{star}$), or an MST ($T_{opt} = T_{mst}$). For $T_{opt} = T_{mst}$ we use random $w_{ij} \in]0, \dots, 10]$ (otherwise, no w_{ij} need to be defined).

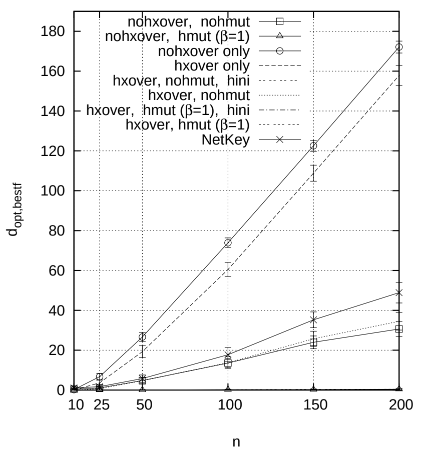
The results for edge-sets without heuristics (nohxover, nohmut) confirm the findings from [3]. This work studied the behavior of non-heuristic KruskalRST* crossover and showed that EAs using this crossover operator show slightly higher performance for $T_{opt} = T_{star}$ than $T_{opt} = T_{rand}$ (because of the small bias of KruskalRST towards stars). The experiments here extend these results and show that for $T_{opt} = T_{mst}$, EA performance is similar to the case $T_{opt} = T_{rand}$.



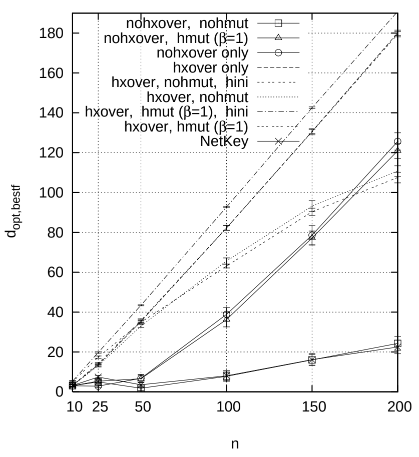
(a) $T_{opt} = T_{rand}$, $\mu = 50$



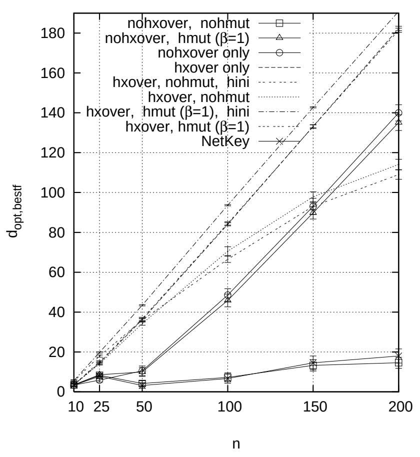
(b) $T_{opt} = T_{star}$, $\mu = 50$



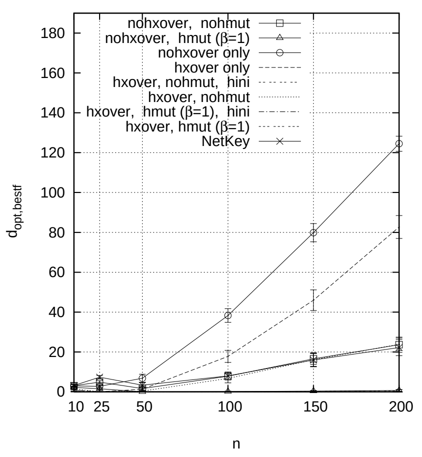
(c) $T_{opt} = T_{mst}$, $\mu = 50$



(d) $T_{opt} = T_{rand}$, $\mu = 200$



(e) $T_{opt} = t_{star}$, $\mu = 200$



(f) $T_{opt} = T_{mst}$, $\mu = 50$

Fig. 4. The figures show the performance of a ($\mu + 1$)-EA for the one-max tree problem. We plot the distance $d_{opt,bestf}$ between the best found solution T_{bestf} and the optimal solution T_{opt} over the problem size n for $\mu = 50$ (top) and $\mu = 200$ (bottom). T_{opt} is either a random tree (left), a star (middle), or an MST (right). EA performance is low for heuristic search operators if the optimal solution is not an MST.

Combining non-heuristic crossover with heuristic mutation (nohxover, hmut ($\beta = 1$)) results in low EA performance for $T_{opt} = T_{rand}$ and $T_{opt} = T_{star}$ because of the bias of the mutation operator. With larger μ, EA performance increases as it results in a lower bias of heuristic mutation (see also Fig. 2). For $T_{opt} = T_{mst}$, the optimal solution can easily be found.

Using only non-heuristic crossover (nohxover only) results in a slightly worse (but still similar) performance than in combination with heuristic mutation for $T_{opt} = T_{rand}$ and $T_{opt} = T_{star}$ because of problems with premature convergence. As no mutation is used, no new edges can be inserted into the population, and EA performance drops slightly. Using a larger μ reduces such problems and increases EA performance. Furthermore, as non-heuristic crossover has no bias towards the MST, EA performance is nearly independent of T_{opt}.

Using only heuristic crossover (hxover only) results in a lower performance than when using non-heuristic crossover for $T_{opt} = T_{rand}$ and $T_{opt} = T_{star}$ since heuristic crossover results in a strong bias towards MSTs. Because of this strong bias, additional mutation (hxover, hmut ($\beta = 1$)) is not helpful for $T_{opt} = T_{rand}$ and $T_{opt} = T_{star}$ and EA performance is similar to that when using only heuristic crossover. Only for $T_{opt} = T_{mst}$ does the performance of EAs solely using heuristic crossover increase. However, as diversity rapidly gets lost and the population converges fast (see Fig. 1), the optimal MST cannot be reached (if no heuristic mutation is used).

EAs using heuristic crossover and non-heuristic mutation combined with either heuristic initialization (hxover, nohmut, hini) or non-heuristic initialization (hxover, nohmut) show similar behavior. Because of the bias of heuristic crossover, EA performance is lower than when using non-heuristic crossover (nohxover, nohmut), but higher than the variants with heuristic mutation. If $T_{opt} = T_{mst}$, using heuristic mutation leads to high EA performance because of the bias of heuristic mutation.

Using heuristic crossover, heuristic mutation with $\beta = 1$, and either heuristic (hxover, hmut ($\beta = 1$), hini) or non-heuristic initialization (hxover, hmut ($\beta = 1$)) leads to low EA performance if T_{opt} is not an MST. Finally, the performance of NetKeys is practically independent of T_{opt} and similar to that of non-heuristic crossover and mutation.

Summarizing the results, the performance of EAs that employ heuristic operators for the one-max tree problem is high when optimum solutions are MSTs but low otherwise.

VI. Performance of Edge-Sets for the OCST Problem

The OCST problem [4] seeks a spanning tree that connects all given nodes and satisfies their communication requirements for a minimum total cost. Let $G = (V, E)$ be a complete undirected graph with $n = |V|$ nodes and $m = |E| = n(n-1)/2$ edges. To every pair of nodes (i, j) a non-negative distance weight w_{ij} and a non-negative communication requirement r_{ij} are associated. The communication cost $c(T)$ of a spanning tree T is defined as

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

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 r_{ij} and $c(T) = \sum_{(i,j)\in E} w_{ij}$.

A. Approaches for Solving the OCST Problem

The OCST problem is not only $\mathcal{NP}$-hard [17, p. 207] but also $\mathcal{MAX}$ $\mathcal{SNP}$-hard [37] which means it cannot be solved using a polynomial-time approximation scheme unless $\mathcal{P} = \mathcal{NP}$ [38]. Therefore, the OCST problem belongs to the class of optimization problems that behave like MAX-3SAT [17].

Only for a few problem instances have algorithms been developed which return optimal solutions. The *optimum requirement spanning tree problem*, where all $w_{ij} = 1$, can be solved in polynomial time using the Gomory-Hu spanning tree algorithm [4], [39]. If all communication requirements are equal ($r_{ij} = 1$), the optimal solution is a star if the w_{ij} satisfy a stronger version of the triangle inequality [4]. All other uniform demand versions, where $w_{ij} \in \{1, \infty\}$ are $\mathcal{NP}$-hard [40], [41]. For the uniform demand version, [42] presented a heuristic that finds a tree T with cost $c(T) \leq 2c(T_{opt})$.

Peleg [43] showed that the OCST problem is reducible to a problem called the *minimum average stretch spanning tree* problem [44]. Therefore, it can be solved by a randomized algorithm that constructs a spanning tree with an average cost less than or equal to $\exp(O(\sqrt{\log n \log\log n}))$ [37], [44]. Other approximation algorithms are based on the volume of communication $c(G) = \sum_{i,j\in E} r_{ij} t^G_{ij}$ in the complete graph G, where t^G_{ij} is the sum of all the weights along the shortest path between i and j in G. $c(G)$ represents a trivial lower bound for $c(T)$. Randomized algorithms construct a spanning tree T with expected cost $c(T) = O(\log n \log\log n)c(G)$ [45]. Non-randomized, deterministic algorithms find a spanning tree with cost $c(T) = O(\log^2 n)c(G)$ [37], [46]. Charikar et al. [47] improved these results and presented a deterministic approximation algorithm that results in $c(T) = O(\log n \log\log n)c(G)$. Focusing on Euclidean w_{ij}, deterministic approximation algorithms output a spanning tree with cost $c(T) = O(\log n)c(G)$ [37], [47]. Despite the progress made, such approximation techniques cannot provide approximations better than $c(T) = \Omega(\log n)c(G)$ [44]. We conclude that no efficient algorithmic methods are available for standard OCST problems.

To overcome the limitations of exact and approximation algorithms, heuristic optimization methods have been used. Palmer [5] recognized that using a proper tree representation is crucial for performance of heuristic optimization methods. Therefore, following his work, a variety of encodings has been studied (for an overview compare Sect. II).

Rothlauf et al. showed that on average, optimal solutions for OCST problems are similar to MSTs [14]. Therefore, operators as well as encodings that are biased towards MSTs are expected to solve OCST problems more efficiently.

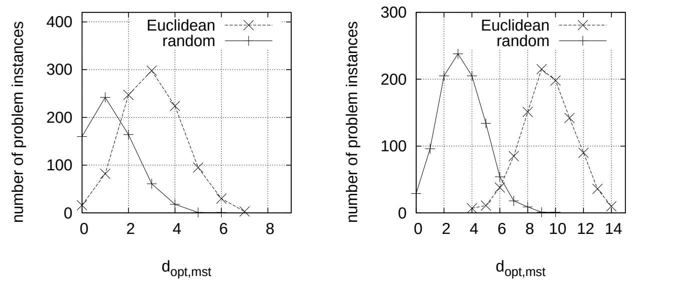
B. A GA for Finding High-Quality Solutions

To be able to study how the performance of edge-sets depends on the structure of T_{opt}, an optimal or near-optimal solution must be determined. However, because of the $\mathcal{NP}$-hardness of the problem, optimal solutions can be determined only for small problem instances with reasonable computational effort. The following algorithm should identify optimal or near-optimal solutions for small OCST problems.

Harik et al. [48] showed that the probability α that a genetic algorithm (GA) with non-overlapping populations fails to find an optimal solution is $O(\exp(-N))$, where N is the GA's population size. Therefore, GA performance increases with N. Consequently, we apply a GA n_{iter} times to an OCST problem using a population size of N_0. T^{best}_0 denotes the best solution of cost $c(T^{best}_0)$ that is found during the n_{iter} runs. In a next round we double the population size and again apply a GA n_{iter} times with a population size of $N_1 = 2N_0$. T^{best}_1 denotes the best solution with cost $c(T^{best}_1)$ that can be found in the second round. We continue this iteration and double the population size $N_i = 2N_{i-1}$ until $T^{best}_i = T^{best}_{i-1}$ and $n(T^{best}_i)/n_{iter} > 0.5$, this means T^{best}_i is found in more than 50% of the runs in round i. $n(T^{best}_i)$ denotes the number of runs that find the best solution T^{best}_i in round i. We assume that the solution T^{bestf}_{last} found in the last iteration is the optimal or near-optimal solution ($T_{opt} = T^{bestf}_{last}$).

For finding optimal solutions T_{opt} for OCST problems of size $n = 10$ and $n = 20$ we use a standard GA with traditional parameter settings. The problem was encoded using NetKeys [7] as they have high locality and represent all possible trees approximately uniformly. The GA uses uniform crossover and tournament selection without replacement. The size of the tournament is three, $p_c = 0.8$, and $p_m = 1/n$ (mutation assigns a random value $[0, 1]$ to an allele). For the GA we started with $N_0 = 100$ and set $n_{iter} = 20$. Each GA run is stopped after a maximum of 200 generations. The computational effort for the experiments is high.

Figure 5 shows the number of problem instances over the distance $d_{opt,mst}$ between T_{opt} (determined by the GA) and the MST for 1,000 randomly created OCST problems with 10 (Fig. 5(a)) and 20 (Fig. 5(b)) nodes. The problems are created randomly using either random or Euclidean w_{ij} (see Sect. IV). The r_{ij} are random and uniformly distributed in]0,10].



(a) 10 nodes (b) 20 nodes

Fig. 5. Distribution of the problem instances over the distance $d_{opt,mst}$ between T_{opt} and MST for 10 and 20 node problems using either random or Euclidean w_{ij}. Optimal solutions for OCST problems are biased towards MSTs.

Comparing $d_{opt,mst}$ to $d_{rand,mst}$ (unbiased representation in Table I) confirms existing results [14] that optimal solutions for OCST problems are biased towards MSTs. Furthermore, optimal solutions for OCST problems with random w_{ij} show a stronger bias than with Euclidean w_{ij}. Because of the bias of T_{opt} towards MSTs, the problem is supposed to be easy for EAs using edge-sets with heuristic operators.

C. *Influence of $d_{opt,mst}$ on the Performance of EAs using Edge-Sets*

We study how the performance of EAs using edge-sets depends on $d_{opt,mst}$. We use the 1,000 problem instances from Sect. VI-B and the steady-state EA from Sect. V-B. Each run is stopped after $eval$ search steps (Table III) and we perform 10 runs for each problem instance.

n	10		20	
μ	50	200	50	200
$eval$	2,000	5,000	10,000	20,000

TABLE III

NUMBER OF SEARCH STEPS $eval$ FOR SMALL OCST PROBLEMS

The plots in Fig. 6 show the percentage of EA runs that find T_{opt} over $d_{opt,mst}$ (left) and the gap $\frac{c(T_{bestf})-c(T_{opt})}{c(T_{opt})}$ (in percent) between the cost of the best found solution T_{bestf} and the cost of T_{opt} over $d_{opt,mst}$ (right). We present results for different population sizes (μ=50 and μ=200) and different problem sizes (n=10 and n=20). Results are only plotted for those $d_{opt,mst}$, where there are more than 10 problem instances. For example, for n=10, μ=50, and Euclidean w_{ij} (Fig. 6(a)) we show results only for $d_{opt,mst} \in \{0, \dots, 6\}$ as there are only 3 (out of 1,000) instances with $d_{opt,mst} = 7$ (compare Fig. 5(a)).

EAs using heuristic crossover always find T_{opt} if T_{opt} is very similar to the MST ($d_{opt,mst}$ is low). For larger $d_{opt,mst}$, EA performance drops sharply and the percentage of runs that find T_{opt} is low. Because of the strong bias of the heuristic initialization, combining heuristic crossover with heuristic initialization results in lower EA performance for larger $d_{opt,mst}$ in comparison with using non-heuristic initialization.

In contrast to heuristic crossover, the performance of EAs using non-heuristic crossover ("nohxover, nohmut" and "nohxover, hmut ($\beta = 1$)") decreases only slightly with larger $d_{opt,mst}$. Combining the non-heuristic crossover with heuristic mutation (nohxover, hmut ($\beta = 1$)) results in higher EA performance than when using non-heuristic mutation (nohxover, nohmut) if $d_{opt,mst}$ is low (compare the $n = 10$ and $n = 20$ node problems with random w_{ij}). This is a result of the bias of the heuristic mutation. In contrast, if $d_{opt,mst}$ is large (compare the $n = 20$ node problems with Euclidean w_{ij}) use of non-heuristic mutation (nohxover, nohmut) results in higher EA performance than use of biased heuristic mutation (nohxover, hmut ($\beta = 1$)). NetKeys show similar performance to edge-sets with non-heuristic crossover and mutation (nohxover, nohmut). Comparing different types of w_{ij}, OCST problems with random w_{ij} are easier for EAs using heuristic search operators since optimal solutions are more similar to MSTs. Using different population sizes yields no differences in EA performance.

In summary, heuristic search operators only perform well for OCST problems where T_{opt} closely resembles an MST. Otherwise EAs using heuristic crossover fail. Using heuristic mutation has a similar, but weaker effect on EA performance. Non-heuristic crossover and mutation results in EA performance which is nearly independent of $d_{opt,mst}$. These finding are confirmed when examining the gap $\frac{c(T_{bestf})-c(T_{opt})}{c(T_{opt})}$.

D. *EA Performance for Test Instances from the Literature*

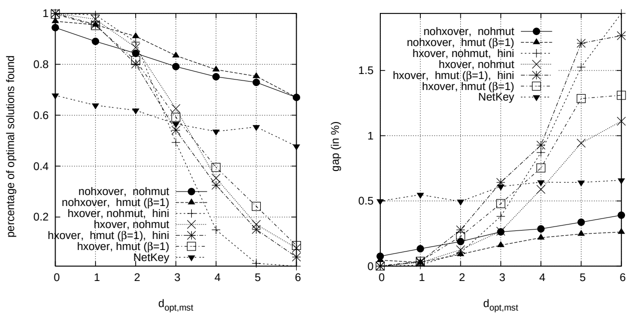
OCST test instances have been proposed by [5], [6], [49]. An analysis of the test instances was performed in [14]. For all test instances optimal or near-optimal solutions are known.

Palmer [5] introduced problem instances with six (palmer6), twelve (palmer12), 24 (palmer24), 47, and 98 nodes. The nodes correspond to US cities and the w_{ij} are obtained from a tariff database. The r_{ij} are inversely proportional to w_{ij}. Berry et al. [6] presented three problem instances, one with six nodes (berry6) and two with 35 nodes (berry35 and berry35u). For berry35u, the weights $w_{ij} = 1$. Raidl [49] proposed several test instances ranging from 10 to 100 nodes. The w_{ij} and r_{ij} were generated randomly and are uniformly distributed in $[0, 100]$.

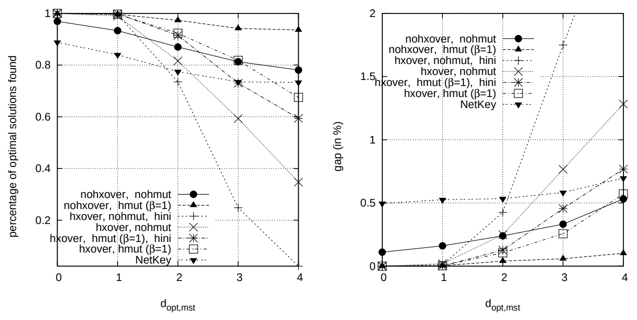
Table IV lists the properties of optimal or best known solutions T_{opt}. It shows the number of nodes n, the average distance $d_{rand,mst}$ of 10,000 randomly generated unbiased trees T_{rand}, the distance $d_{opt,mst}$, and the cost $c(T_{opt})$. In the instance berry35u, all distances are uniform ($w_{ij} = 1$), so all spanning trees are minimal. For all test instances, $d_{opt,mst}$ is always smaller than $d_{rand,mst}$. Therefore, optimal solutions are biased towards MSTs.

For our experiments we use the same steady-state EA as before. Each run is stopped after $eval$ search steps (Table IV) and we perform 25 runs for each test instance. We compare the performance of EAs that use the same search operators as described in Sects. IV-B and V-B. As optimal solutions are similar to MSTs, we extend our study and also present results for the following search operators:

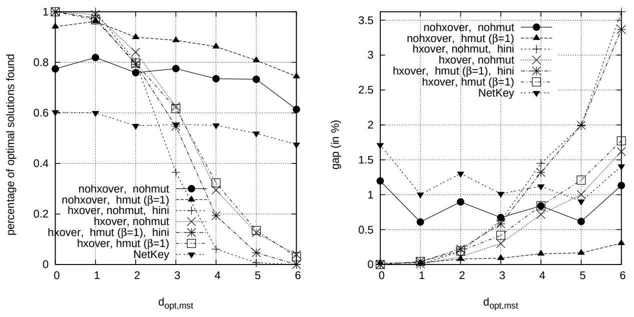
- **nohxover, nohmut (MST)**: edge-sets with non-heuristic crossover ($p_c = 1$), non-heuristic mutation ($p_m = 1/n$),



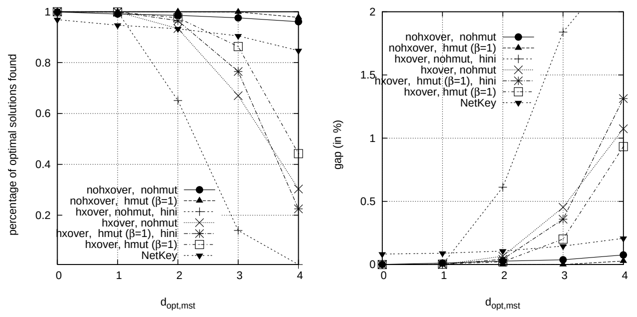
(a) $n = 10$, $\mu = 50$, Euclidean distance weights



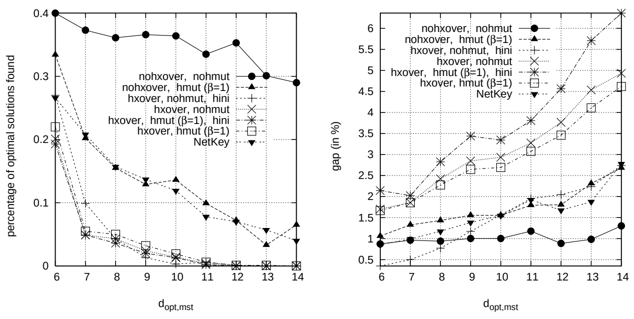
(b) $n = 10$, $\mu = 50$, random distance weights



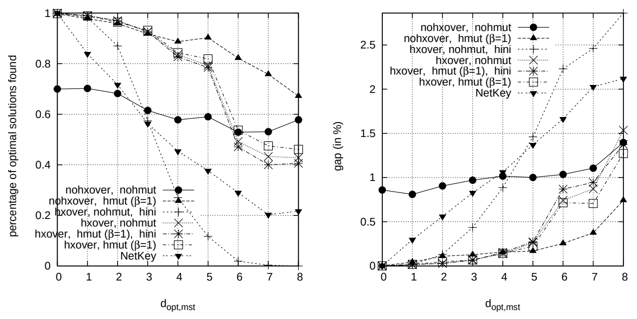
(c) $n = 10$, $\mu = 200$, Euclidean distance weights



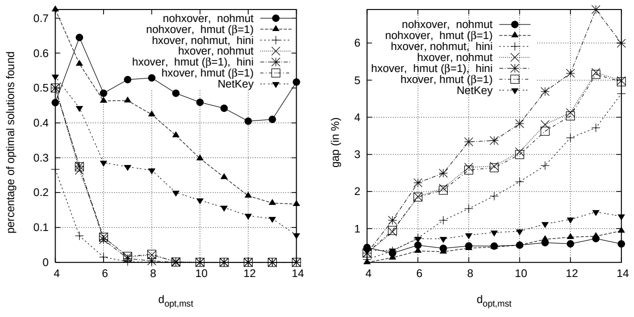
(d) $n = 10$, $\mu = 200$, random distance weights



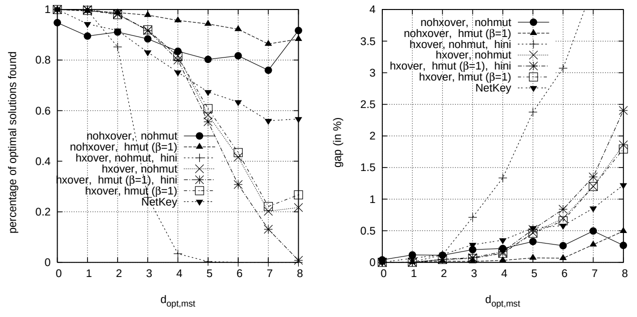
(e) $n = 20$, $\mu = 50$, Euclidean distance weights



(f) $n = 20$, $\mu = 50$, random distance weights



(g) $n = 20$, $\mu = 200$, Euclidean distance weights



(h) $n = 20$, $\mu = 200$, random distance weights

Fig. 6. Performance of a steady-state $(\mu + 1)$-EA for random n=10 (Figs. 6(a)-6(d)) and n=20 (Figs. 6(e)-6(h)) problems. Each figure shows the average percentage of optimal solutions that can be found over $d_{opt,mst}$ (left) and the mean of the gap between the cost of the best found solution T_{bestf} and the cost of T_{opt} over $d_{opt,mst}$. Heuristic crossover outperforms the non-heuristic variant only if optimal solutions closely resemble MSTs ($d_{opt,mst}$ is small). For larger $d_{opt,mst}$, heuristic variants result in low EA performance. In contrast, non-heuristic crossover results in nearly constant EA performance.

TABLE IV

PERFORMANCE OF EA USING VARIANTS OF THE CROSSOVER OPERATOR FOR OCST TEST PROBLEMS FROM THE LITERATURE

problem	n	$d_{rand,mst}$	optimal solutions		$eval$		T_{rand}	MST	nohxover, nohmut		nohxover	hxover, hini		hxover	NetKey	
instance			$d_{opt,mst}$	$c(T_{opt})$					(MST)	(rnd)	hmut (β=1)	nohmut	hmut (β=1)	hmut (β=1)	(MST)	(rnd)
palmer6	6	3.36	1	693,180	300	p_{suc}	-	0	0.64	0.64	0.82	0.98	1	1	0.64	0.6
						$d_{bestf,opt}$	-	1	0.52	0.54	0.26	0.04	0	0	0.56	0.48
						gap (in %)	149	2.39	0.52	0.68	0.30	0.04	0	0	0.61	0.58
palmer12	12	9.17	5	3,428,509	5,000	p_{suc}		- 0	0.2	0.26	0.24	0	0.04	0.12	0.28	0.22
						$d_{bestf,opt}$	-	5	2.72	2.62	2.06	3.08	1.56	1.46	2.28	2.48
						gap (in %)	244	13.07	1.07	1.02	0.61	1.74	0.47	0.40	0.82	0.98
palmer24	24	21.05	12	1,086,656	20,000	p_{suc}	-	0	1	1	0	0.88	0	0	0.16	0.2
						$d_{bestf,opt}$	-	12	0	0	3.2	0.12	6.48	4.84	1.8	1.48
						gap (in %)	852	80.35	0	0	5.93	0.01	29.68	22.35	2.45	1.32
raidl10	10	7.20	3	53,674	1,000	p_{suc}	-	0	0.58	0.48	1	0	0.5	0.74	0.34	0.36
						$d_{bestf,opt}$	-	3	0.44	0.64	0	1.6	0.5	0.26	0.88	1.02
						gap (in %)	512	8.72	0.94	1.76	0	3.25	0.89	0.46	2.14	3.82
raidl20	20	17.07	2	157,570	5,000	p_{suc}	-	0	0.44	0	0.74	0	1	0.96	0.56	0.08
						$d_{bestf,opt}$	-	2	0.76	6.8	1.12	1.18	0	0.24	0.58	4.6
						gap (in %)	1,145	5.22	1.10	33.13	0.86	0.59	0	0.18	0.47	17.10
raidl50	50	47.09	13	806,864	40,000	p_{suc}	-	0	0.067	0	0.467	0	0	0	0	0
						$d_{bestf,opt}$	-	13	4.27	12.53	1.27	8.73	5.53	4.13	7.8	20.67
						gap (in %)	2,493	13.07	0.80	7.98	0.15	4.10	1.59	1.10	3.30	28.30
raidl75	75	72.02	18	1,717,491	40,000	p_{suc}	-	0	0	0	0.067	0	0	0	0	0
						$d_{bestf,opt}$	-	18	12.4	46.33	12.27	14.8	2.87	3.87	18.07	54.93
						gap (in %)	3,414	39.95	4.64	57.80	3.00	7.61	0.26	0.66	13.67	127.03
raidl100	100	97.09	32	2,561,543	80,000	p_{suc}	-	0	0	0	0.267	0	0	0	0	0
						$d_{bestf,opt}$	-	32	20	53.53	7.87	22.47	2.2	4.67	29.27	68.47
						gap (in %)	4,534	40.82	5.80	37.01	1.20	7.32	0.53	0.58	13.95	106.74
berry6	6	3.51	0	534	300	p_{suc}	-	1	1	0.88	1	1	1	1	1	0.8
						$d_{bestf,opt}$	-	0	0	0.12	0	0	0	0	0	0.22
						gap (in %)	140	0	0	0.57	0	0	0	0	0	1.15
berry35u	35	-	-	16,273	40,000	p_{suc}	-	0	0	0.52	0.02	0.02	0	0.02	0	0
						$d_{bestf,opt}$	-	-	6.64	6.1	12.44	6.6	13.82	11.86	11.6	11.26
						gap (in %)	308	37.22	2.19	1.54	8.31	2.13	10.16	6.92	6.25	6.66
berry35	35	32.05	0	16,915	20,000	p_{suc}	-	1	1	0.92	1	1	1	1	1	0.02
						$d_{bestf,opt}$	-	0	1	0.46	0.64	1	1	0.96	1	5.58
						gap (in %)	2,143	0	0	0.23	0	0	0	0	0	16.28

and non-heuristic initialization. One randomly chosen initial solution is an MST (all other $\mu-1$ initial solutions are generated randomly).

- **NetKey (MST)**: uniform crossover ($p_c = 1$), mutation ($p_m = 1/n$), and random initial population. One randomly chosen initial solution is an MST (all other $\mu - 1$ initial solutions are random).

The original variants "nohxover, nohmut" and "NetKey" with random initial populations are denoted as "nohxover, nohmut (rnd)" and "NetKey (rnd)", respectively. Furthermore, we show results for 10,000 randomly generated trees T_{rand} (indicated as T_{rand}).

Table IV lists the percentage of runs p_{suc} that find T_{opt}, the distance $d_{bestf,opt}$ between the best solution T_{bestf} that was found after $eval$ search steps and T_{opt} (averaged over all 25 runs), and the gap $\frac{c(T_{bestf})-c(T_{opt})}{c(T_{opt})}$ (in percent) between $c(T_{bestf})$ and $c(T_{opt})$ (also averaged over all 25 runs). For the 10,000 randomly generated solutions T_{rand}, we present results for the gap $\frac{c(T_{rand})-c(T_{opt})}{c(T_{opt})}$ (in percent averaged over all 10,000 random trees T_{rand}).

Since the difference between $c(T_{mst})$ and $c(T_{opt})$ is much lower than the difference between $c(T_{rand})$ and $c(T_{opt})$, already an MST is always a high-quality solution for all test problems. For berry6 and berry35, where the optimal solution is an MST, EAs using heuristic crossover (hxover) can easily find T_{opt}. For test instances with a large $d_{opt,mst}$ (for example the large raidl test instances), EAs using heuristic crossover have problems finding T_{opt}. Although, the gap between $c(T_{bestf})$ and $c(T_{opt})$ is low and similar to non-heuristic crossover and heuristic mutation (nohxover, hmut ($\beta = 1$)), the strong bias of the heuristic crossover does not allow EAs to reach T_{opt}. High EA performance is obtained (except for palmer24) when combining non-heuristic crossover with heuristic mutation (nohxover, hmut ($\beta = 1$)). Heuristic mutation results in a lower bias towards MSTs than heuristic crossover (compare Figs. 1 and 2) and also allows high EA performance for problems where $d_{opt,mst}$ is large.

The results confirm the findings from the previous sections. Non-heuristic crossover shows good performance independently of $d_{opt,mst}$. Heuristic crossover finds optimal solutions only if they resemble MSTs.

E. EA Performance for Randomly Generated OCST Problems with an Unknown Optimal Solution

Finally, we study edge-sets' performance for large OCST problem instances. We do not know T_{opt}, and EA performance is determined only by the fitness $c(T_{bestf})$ of the best found solution T_{bestf}. We must bear in mind that optimal solutions are similar to MSTs, and in comparison with randomly chosen trees, an MST is already a high-quality solution.

We use the same steady-state EA as before. Each EA run is stopped after $eval$ search steps (Table V). We present results for random problems of different sizes ($n \in \{10, 25, 50, 100, 150, 200\}$) with either random or Euclidean w_{ij}. For each type of problem, we create either 100 ($n < 100$), 25 ($100 \leq n \leq 150$), or 10 ($n = 200$) random instances. For

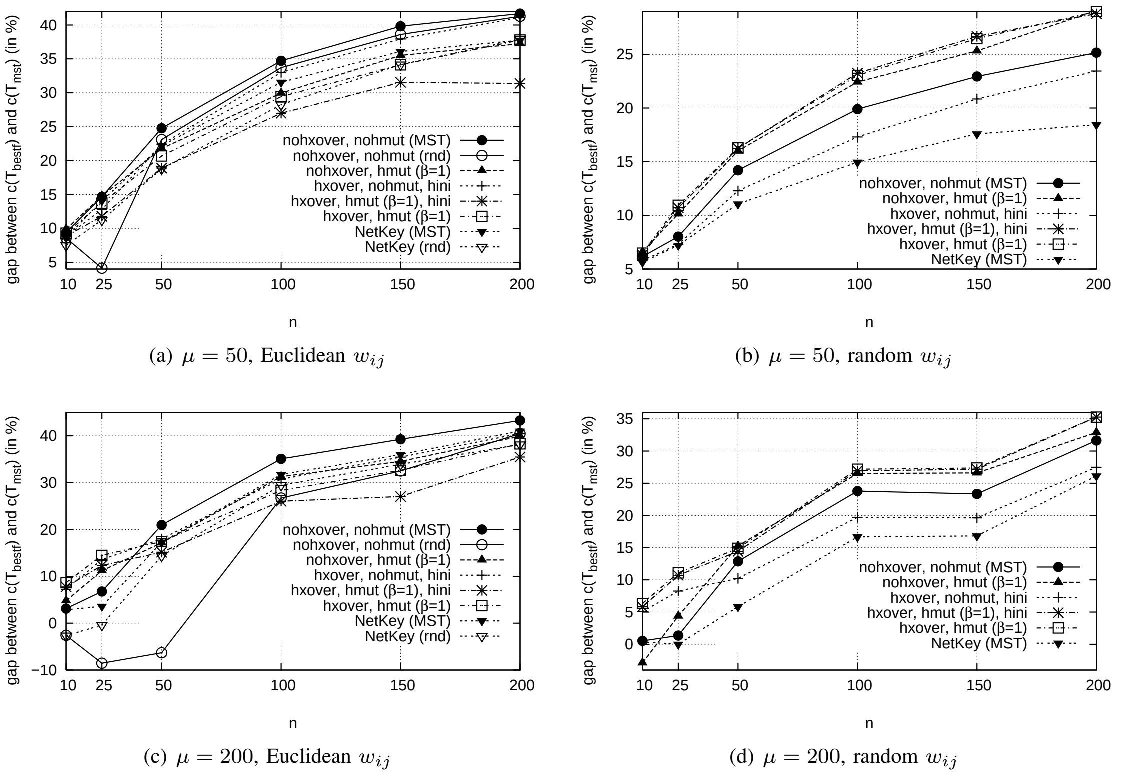


Fig. 7. The figures compare the performance of a steady-state EA (population size μ) for randomly generated OCST problems with either Euclidean (left) or random w_{ij} (right). Each figure shows the average gap between the cost of the best found solutions T_{bestf} and MSTs over the problem size n (higher performance results in higher gap). For random w_{ij}, heuristic search operators outperform non-heuristic variants as $d_{opt,mst}$ is low. With larger $d_{opt,mst}$ (Euclidean w_{ij}), the bias of the heuristic operators hinders finding optimal solutions and non-heuristic variants result in higher EA performance.

n	10	25	50	100	150	200
$eval$	1,000	5,000	20,000	80,000	160,000	320,000

TABLE V

NUMBER OF SEARCH STEPS $eval$ FOR LARGE OCST PROBLEMS

each instance, 10 EA runs are performed. We compare the search operators described in Sects. IV-B, V-B, and VI-D.

Figure 7 presents the average gap $\frac{c(T_{mst})-c(T_{bestf})}{c(T_{mst})}$ (in percent) between the cost $c(T_{bestf})$ of the best found solution T_{bestf} at the end of each run and the cost $c(T_{mst})$ of the MST over the problem size n. The larger the gap, the better the solutions, which are found. We compare the performance between different search operators based on the fitness gap between T_{bestf} and MST since an MST is already a high-quality solution and the design of the search operators used is such that MSTs (or slight variants of it) are created with a high probability in either the initial population (heuristic initialization), or during the EA run (heuristic mutation or crossover). For random distance weights, we do not present results for "nohxover, nohmut (rnd)" and "NetKey (rnd)" as EAs using these representation variants have not been able to find solutions that have a cost similar to or lower than the MST.

There are differences between problem instances with Euclidean w_{ij} (left) and random w_{ij} (right). For Euclidean w_{ij}, best solutions are found when using non-heuristic search operators starting with an MST ("nohxover, nohmut (MST)"). Solution quality is low when using heuristic initialization, crossover, and mutation (hxover, hmut ($\beta = 1$), hini). The situation is different for random w_{ij}. Here, EA performance is high for heuristic initialization, crossover, and mutation; in contrast, EA performance is lower when using the non-heuristic variant. This is because OCST problems with random w_{ij} have optimum solutions that are more similar to MSTs (see Fig. 5) and, thus, search operators with a strong bias towards MSTs result in high EA performance. Consequently, heuristic variants like "hxover, hmut ($\beta = 1$), hini" with a strong bias towards MSTs perform better for random w_{ij} (optimal solutions are more similar to MSTs) than for Euclidean w_{ij}. Comparing the results for $\mu = 50$ and $\mu = 200$ reveals no great differences (especially for larger n).

The results confirm the previous findings. Heuristic search operators result in high EA performance if optimal solutions resemble MSTs (OCST problems with random w_{ij}). In contrast, for larger $d_{opt,mst}$ (OCST problems with Euclidean w_{ij}) non-heuristic operators show higher performance.

VII. SUMMARY AND CONCLUSIONS

This work investigates the bias of edge-sets [3] and examines how their performance depends on the type of optimal solution. Results are presented for the one-max tree problem and the optimal communication spanning tree (OCST) problem. As expected and intended by the design of the encoding [3], edge sets' heuristic initialization, crossover, and mutation operators are biased towards MSTs. The bias is especially strong for the heuristic crossover operator. In contrast, the non-heuristic search operators show no significant bias and their application results in an undirected and uniform search.

For the one-max tree problem, using edge-sets' heuristic crossover and mutation operators leads to low EA performance if the optimal solution is a random tree or star and to high EA performance if the optimal solution is an MST. In contrast, the performance of EAs using edge-sets' non-heuristic operators is practically independent of the type of optimal solution.

For OCST problems, the bias of optimal solutions towards MSTs is stronger for problems defined on random distance weights w_{ij} in comparison with problems defined on Euclidean w_{ij}. Because of the bias of edge-sets' heuristic operators towards MSTs, EAs using such operators can easily solve OCST problems where optimal solutions are similar to MSTs (OCST problems with random w_{ij}). However, with decreasing similarity between MSTs and optimal solutions (OCST problems with Euclidean w_{ij}), the performance of EAs using heuristic operators strongly decreases and non-heuristic operators lead to better results. For example, studying small OCST problems with 20 nodes and Euclidean w_{ij} shows that, if the optimal solution and the MST share less than 14 (out of 19 possible) edges, EAs using heuristic operators find a maximum of 10% of the optimal solutions whereas EAs using non-heuristic variants find about 50%.

The results show that edge-sets' heuristic operators are a good choice for tree problems only where optimal solutions closely resemble MSTs. The problems of edge-sets' heuristic operators emphasize the difficulty of a proper design of direct representations. In contrast to indirect representations, the behavior of new, problem-specific search operators is often unknown. Although optimal solutions for OCST problems are biased towards MSTs, edge-sets with heuristic operators that use this problem-specific knowledge and are biased towards MSTs can fail. Therefore, we recommend using unbiased representations/operators if there is no problem-specific knowledge a priori. Proper representations for such tree problems are for example edge-sets with non-heuristic operators or NetKeys. When biased representations/operators are used, their bias must match the properties of optimum solutions, otherwise failure is unavoidable.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[6] 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: Telkom South Africa, 1995, pp. 361–370.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 2nd ed. Heidelberg: Springer, 2006.

[13] B. Julstrom and G. Raidl, "Weight-biased edge-crossover in evolutionary algorithms for two graph problems," in *Proceedings of the 16th ACM Symposium on Applied Computing*, G. L. et al., Ed. ACM Press, 2001, pp. 321–326.

[14] 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.

[15] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

[16] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

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

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

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

[20] L. T. M. Berry, B. A. Murtagh, G. McMahon, and S. Sugden, "Optimization models for communication network design," in *Proceedings of the Fourth International Meeting Decision Sciences Institute*, Sydney, Australia, 1997, pp. 67–70.

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

[22] M. Krishnamoorthy and A. T. Ernst, "Comparison of algorithms for the degree constrained minimum spanning tree," *Journal of Heuristics*, vol. 7, pp. 587–611, 2001.

[23] A. Cayley, "A theorem on trees," *Quarterly Journal of Mathematics*, vol. 23, pp. 376–378, 1889.

[24] E. H. Neville, "The codifying of tree-structure," *Proceedings of the Cambridge Philosophical Society*, vol. 49, pp. 381–385, 1953.

[25] N. Deo and P. Micikevicius, "Prüfer-like codes for labeled trees," *Congressus Numerantium*, vol. 151, pp. 65–73, 2001.

[26] S. Picciotto, "How to encode a tree," Ph.D. dissertation, University of California, San Diego, USA, 1999.

[27] S. Caminiti, I. Finocchi, and R. Petreschi, "A unified approach to coding labeled trees," in *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium*, ser. LNCS, M. Farach-Colton, Ed., vol. 2976. Springer, 2004, pp. 339–348. [Online]. Available: http://springerlink.metapress.com/openurl.asp?genre=article&amp;issn=0302-9743&amp;volume=2976&amp;spage=339

[28] F. Rothlauf and D. E. Goldberg, "Tree network design with genetic algorithms - an investigation in the locality of the prüfernumber encoding," in *Late Breaking Papers at the Genetic and Evolutionary Computation*

Conference 1999, S. Brave and A. S. Wu, Eds. Orlando, Florida, USA: Omni Press, 1999, pp. 238–244.

[29] B. A. Julstrom, "The blob code: A better string coding of spanning trees for evolutionary search," in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program*, A. S. Wu, Ed. San Francisco, California, USA: ISGEC, 2001, pp. 256–261.

[30] S. Caminiti and R. Petreschi, "String coding of trees with locality and heritability," in *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005*, ser. LNCS, L. Wang, Ed., vol. 3595. Springer, 2005, pp. 251–262. [Online]. Available: http://dx.doi.org/10.1007/11533719_27

[31] T. Paulden and D. K. Smith, "From the dandelion code to the rainbow code: A class of bijective spanning tree representations with linear complexity and bounded locality," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 124–144, 2006.

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

[33] G. R. Raidl and J. Gottlieb, "Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem," *Evolutionary Computation*, vol. 13, no. 4, pp. 441–475, 2005.

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

[35] G. R. Raidl, G. Koller, and B. A. Julstrom, "Biased mutation operators for subgraph-selection problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 145–156, 2006.

[36] 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.

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

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

[39] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," in *SIAM Journal on Applied Math*, vol. 9, 1961, pp. 551–570.

[40] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan, "The complexity of the network design problem," *Networks*, vol. 8, pp. 279–285, 1978.

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

[42] R. Wong, "Worst case analysis of network design problem heuristics," *SIAM J. Algebraic Discr. Meth.*, vol. 1, pp. 51–63, 1980.

[43] D. Peleg, "Approximating minimum communication spanning trees," Proc. 4th Colloq. on Structural Information and Communication Complexity, Ascona, Switzerland, 1997.

[44] N. Alon, R. M. Karp, D. Peleg, and D. West, "A graph theoretic game and its application to the k-server problem," *SIAM Journal on Computing*, vol. 24, pp. 78–100, 1995.

[45] Y. Bartal, "On approximating arbitrary metrics by tree metrics," in *Proc. 30th Anual ACM Symp. on Theory of Computer Science*, 1998, pp. 161–168.

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

[47] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin, "Approximating a finite metric by a small number of tree metrics," in *Proc. 39th IEEE Symp. on Foundations of Computer Science*, 1998, pp. 111–125.

[48] 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.

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