The Node-Depth Encoding: Analysis and Application to the Bounded-Diameter Minimum Spanning Tree Problem

Telma W. de Lima
Institute of Mathematics and Computer Sciences
University of Sao Paulo
telma@icmc.usp.br

Franz Rothlauf
Dept. of IS and Business Administration
University of Mainz
rothlauf@uni-mainz.de

Alexandre C. B. Delbem
Institute of Mathematics and Computer Sciences
University of Sao Paulo
acbd@icmc.usp.br

ABSTRACT

The node-depth encoding has elements from direct and indirect encoding for trees which encodes trees by storing the depth of nodes in a list. Node-depth encoding applies specific search operators that is a typical characteristic for direct encodings. An investigation into the bias of the initialization process and the mutation operators of the node-depth encoding shows that the initialization process has a bias to solutions with small depths and diameters, and a bias towards stars. This investigation, also, shows that the mutation operators are unbiased. The performance of node-depth encoding is investigated for the bounded-diameter minimum spanning tree problem. The results are presented for Euclidean instances presented in the literature. In contrast with the expectation, the evolutionary algorithm using the biased initialization operator does not allow evolutionary algorithms to find better solutions compared to an unbiased initialization. In comparison to other evolutionary algorithms for the bounded-diameter minimum spanning tree evolutionary algorithms using the node-depth encoding have a good performance.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving and Search-Heuristics Methods

General Terms

Algorithms

Keywords

Genetic Algorithms, Representations, Performance Analysis

1. INTRODUCTION

Evolutionary Algorithms (EAs) are often a good choice for complex optimization problems involving characteristics that make such problems difficult for classical optimization methods [5]. In particular, EAs have often been applied successfully to tree problems, such as the degree-constraint minimum spanning tree problem (d-MSTP) [12, 17], the optimal communication spanning tree problem (OCSTP) [9, 13], or the bounded-diameter minimum spanning tree problem (BDMSTP) [11, 8].

In EAs, the proper choice of the encoding is crucial. Solutions need to be encoded so that genetic search operators like mutation and recombination can be applied. There are two different approaches: direct and indirect representations. In direct representations, a tree is represented by a set of edges and the search operators are directly applied to the set of edges. Since solutions are often constrained, it is usually necessary to develop search operators specific for the problem that consider such constraints. Examples of direct representations are the edge-set encoding [17] or the NetDir encoding [20]. On the other hand, indirect representations use an explicit encoding process to encode a tree (phenotype) as a list of strings (genotypes). Therefore, standard search operators can usually be applied to the string genotypes. Examples for indirect tree encodings are NetKeys [21], link and node biased [13], Prüfer numbers [15], Blob code [14], or determinant factorization [2].

The node-depth encoding [6] has elements from both, direct and indirect representations. It encodes trees as a list of strings with the depth of nodes and constructs phenotypes by an explicit genotype-phenotype mapping which is typical for indirect representations. Furthermore, the node-depth encoding applies search operators specific for the problem to the genotypes which are typical for direct representations.

For a proper use of encodings such as the node-depth encoding, it is important to have a proper understanding of the properties of the encoding. Relevant properties of encodings are locality and bias [16, 20]. The locality measures whether small changes in the genotype correspond to small changes in the phenotype. For guided search, as evolutionary algorithms, high locality is necessary. The bias of an encoding describes whether either the genotype-phenotype mapping or the evolutionary search operators like mutation or recombination prefer a specific type of solution and thus, leads a population towards this direction.

The purpose of this paper is to thoroughly investigate the bias of the node-depth encoding. The results show that the initialization operator of the encoding shows a bias towards trees with low diameter and low maximum depth. Therefore, in a second step, the encoding is applied to the bounded-diameter minimum spanning tree problem where the use of the initialization operator is expected to result in high

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

depth	0	1	2	1	2	3	2	3	4	5	3	4	4	5	4
node	1	2	8	3	9	10	4	11	12	13	5	14	6	7	15

Figure 1: A genotype using the node-depth encoding.

EA performance. The results show that for the bounded-diameter MST, EAs using the node-depth encoding perform well in comparison to other EAs [11, 8, 23], but - in contrast to our expectation - using the biased initialization operator does not allow EAs to find better solutions in contrast to an unbiased initialization scheme based on Prüfer numbers.

The following section reviews basic concepts of the node-depth encoding. Section 3 analysis the bias of the encoding. Then, section 4 studies and compares the performance of EAs using the node-depth encoding for the bounded-diameter MST problem. The paper ends with concluding remarks.

2. NODE-DEPTH ENCODING

The node-depth encoding [6] is based on storing the depth of the nodes in an ordered list. For a tree of size n, the nodes v_i ($i \in \{1, \ldots, n\}$) and their corresponding depths d_i are listed in an array as pairs (v_i, d_i). The order of the nodes in the array is important since a construction algorithm traverses this array from the first to the last position determining the edges that compose the phenotype. A depth-first search (DFS) [4] can be used to construct a genotype from a tree by starting from a root node v_1 with depth $d_1 = 0$ and iteratively inserting the pairs (v_i, d_i) in the genotype in the order in which the nodes v_i are visited by the DFS.

The following paragraphs explain the relevant aspects of the encoding, such as initialization, decoding process, and mutation operators. It is important to note that only mutation operators have been proposed for the node-depth encoding, *i.e.* no recombination operator is available in the literature.

2.1 Initialization

Initialization consists of two steps. In the first step, a permutation of n node labels is created. The elements of the permutation correspond to the labels v_i of the n nodes. The permutation is stored in an array of length n and $v_i \in \{1, \ldots, n\}$ corresponds to the label of the node at position i in the array. Often, the first element in the array is fixed and assigned, for example, to node $v_1 = 1$.

In a second step, random depths are assigned to the nodes. To the first node of the list, the depth $d_1 = 0$ is assigned. The second node has depth $d_2 = 1$. The depth of the other nodes $\{v_2, v_3, \ldots, v_n\}$ is iteratively set (starting with $i = 2$) to d_i which is randomly chosen from

$$\{1, \ldots, \max_{j=0,\ldots,i-1}(d_j)\}.$$

Figure 1 presents an example of a genotype of a tree with $n = 15$.

2.2 Decoding

The decoding process is tree-specific and traverses a genotype from left ($i = 1$) to right ($i = n$). Starting with node v_2, each node v_i with the corresponding depth d_i is connected to the node v_j ($j \in \{1, \ldots, i-1\}$) with depth $d_j = d_i - 1$,

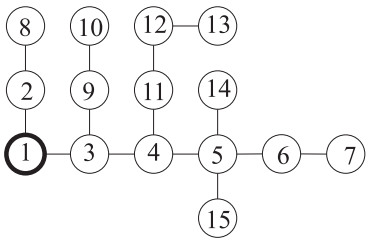


Figure 2: The spanning tree encoded by the genotype of Figure 1

where $i - j$ is minimal. This means, that there is an edge between node v_i with depth d_i and the v_j-nearest preceding node with a depth $d_i - 1$, i.e a depth that is one lower than the depth of d_i. The decoding process is linear in the number of nodes ($O(n)$) and can be implemented using Algorithm 1.

Algorithm 1 Decoding process for node-depth encoding

```
//Let N be a genotype and N_i be a pair (v_i, d_i) at position i ∈ {1, n}
//Let T be the phenotype; let V(T) and E(T) be the nodes and edge sets of T, respectively
//Let r be an array of size n to store node labels
E(T) = ∅
V(T) = v_0
r_0 = v_0
i = 1
while i < |N| do
  V(T) = V(T) ∪ v_j
  r_{d_i} = v_i
  E(T) = E(T) ∪ (r_{d_i - 1}, v_i)
  i = i + 1
end while
```

We want to give a brief example. Assuming the genotype shown in Figure 1, Algorithm 1 starts by adding nodes $v_1 = 1$ and $v_2 = 2$ as well as edge $(1, 2)$ to the tree. Node $v_3 = 8$ has depth $d_3 = 2$ and it is thus connected to node $v_2 = 2$ with $d_2 = d_3 - 1$. Node $v_4 = 3$ has depth $d_4 = 1$ and thus edge $(3, 1)$ is added. Repeating this process for the subsequent nodes, we obtain the tree shown in Figure 2.

2.3 Mutation Operators

This section presents two mutation operators used by the node-depth encoding: **PAO** (Preserve Ancestor[1] Operator) and **CAO** (Change Ancestor Operator), presented in the next subsection. PAO and CAO generate new solutions by changing a pruned subtree from one node to another node in the tree.

PAO requires a set with two nodes previously determined: the pruned node p, which indicates the root of the subtree to be transferred and the adjacent node a. CAO requires a set with three nodes: the pruned node p, adjacent node a, and a randomly chosen node r of the subtree. We briefly illustrate the functionality of both operators and assume that the required set of nodes has been previously determined [6].

[1]The root of a tree (or subtree) can be seen as an ancestor of other nodes in the tree (or subtree).

depth	0	1	2	1	2	3	4	5	5	6	5	2	3	4	5
node	1	2	8	3	9	10	5	14	6	7	15	4	11	12	13

Figure 3: The node-depth encoding obtained after the application of PAO.

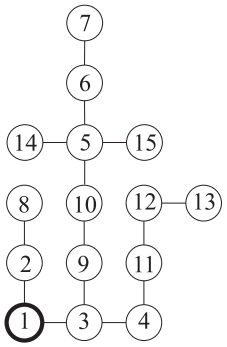


Figure 4: The spanning tree encoded by the genotype of Figure 3.

2.3.1 Preserve Ancestor Operator

The application of PAO to a tree is equivalent to transferring a subtree rooted at node p to another node a. PAO causes small changes in the tree. The following steps describe the application of PAO, where i_p and i_a are the positions of nodes p and a in the genotype.

1. Choose a random node p and find the subtree that should be moved.
2. Create a new tree T' by inserting the subtree rooted at p after node a. In the genotype the positions of the corresponding nodes are moved to the new positions and their depths are set according to the new position.

We want to give a brief example of the functionality of PAO using the tree presented in Figure 2, whose node-depth encoding is in Figure 1. We randomly choose node $p = 5$ and node $a = 10$. The range of nodes 5, 14, 6, 7, 15 corresponds to the subtree that should be moved. Then, PAO inserts it after node a generating a new genotype (see Figure 3) representing the tree presented in Figure 4.

2.3.2 Change Ancestor Operator

In CAO, the transferred subtree will have a new root (any node of the subtree rooted at p different from the original root p) and this node will be named r. CAO causes larger modifications in the tree. The differences between PAO and CAO are in the composition of pruned subtrees. CAO changes the order of the subtrees rooted in the nodes that are in the path between p and r. We assume three nodes p, r, and a previously determined [6].

1. Choose a random node p and find the subtree that should be moved. Determine a node r in this subtree to be the new root.
2. Copy the subtree of r to a temporary array. Consider the nodes in the path from r to p as roots of subtrees. Copy the subtree rooted at it node in the path without the subtree rooted in the previous node. Store the resultant subtrees in the temporary array.

depth	3	4	5	6	6
node	7	6	5	14	15

Figure 5: The node-depth encoding of the temporary array.

depth	0	1	2	3	4	5	6	6	1	2	3	2	3	4	5
node	1	2	8	7	6	5	14	15	3	9	10	4	11	12	13

Figure 6: The node-depth encoding obtained after the application of CAO.

3. Create a new tree T' inserting the nodes of the temporary array to position $i_a + 1$ updating the depths according the position.

We also present a short example of a CAO application. Considering, the tree of Figure 2 and its node-depth encoding in Figure 1. The triple of nodes is chosen as node $p = 5$, $a = 8$, and $r = 7$. The resulting offspring is shown in Figure 6 and Figure 7 shows the corresponding genotype.

3. ANALYZING THE BIAS OF THE NODE-DEPTH ENCODING

In an unbiased encoding, all possible phenotypes are encoded uniformly. Analogously, unbiased search operators do not over-represent specific solutions, and the application of the search operator without selection does not modify the statistical properties of the population. Biased encodings and search operators can be used if it is known a priori that optimal solutions are similar to the over-represented solutions [22]. In contrast, unbiased encodings and search operators should be used if no problem-specific knowledge is available.

The following sections analyze the bias of the node-depth encoding for tree problems with $n = 15$, 20, 25, 50 and 100 nodes. We randomly create 10 problem instances of each size, where $w_{i,j}$ are uniformly distributed between $[0 \dots 1]$. We investigate the bias of the initialization and mutation operators.

In order to analyze the bias, the following measures are used: distance from the MST, distance from stars, diameter and depth. The distance between two spanning trees T_i

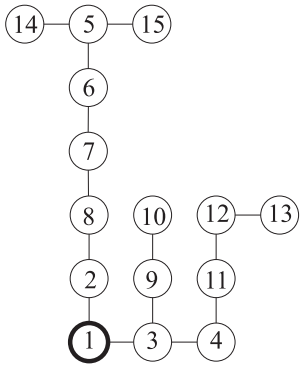


Figure 7: The spanning tree encoded by the genotype of Figure 6.

Table 1: Properties of solutions generated using either, the initialization of the node-depth encoding, or the unbiased Prüfer numbers. We show average depth, diameter, distance from MST and minimum distance from stars.

	depth				diameter				distance from MST				min distance from stars			
	node-depth		Prüfer		node-depth		Prüfer		node-depth		Prüfer		node-depth		Prüfer	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
$n = 15$	4.4	0.9	7.7	1.6	5.7	1.1	8.1	1.6	12.1	1.1	12.1	1.2	7.4	1.6	9.9	0.6
$n = 20$	4.6	0.9	9.2	1.9	6.3	1.2	9.8	1.9	17.1	1.2	17.1	1.2	10.7	2.1	14.6	0.7
$n = 25$	4.9	0.9	10.5	2.2	6.8	1.2	11.2	2.3	22.1	1.2	22.1	1.2	14.0	2.6	19.4	0.7
$n = 30$	5.0	0.9	11.6	2.4	7.2	1.2	11.5	2.6	27.1	1.2	27.1	1.3	17.2	3.0	24.2	0.7
$n = 50$	5.4	0.9	15.6	3.4	8.1	1.2	16.9	3.5	47.0	1.3	47.0	1.3	29.9	4.2	43.8	0.7
$n = 100$	5.9	0.9	22.8	5.4	9.14	1.2	24.7	5.4	97.0	1.3	97.0	1.3	61.6	6.1	93.3	0.7

and T_j is measured using the Hamming distance that is the number of different edges in the two trees, *i.e.* the number of edges that the trees do not share. This distance is calculated by

$$d_{i,j} = \frac{1}{2} \sum_{u,v \in V, u<v} |l^i_{u,v} - l^j_{u,v}|,$$

where l^i_{uv} is 1 if an edge from u to v exists in T_i, and 0 if it does not exist. The minimum distance to a star is calculated as

$$d_{i,STAR} = min(d_{i,star_j}),$$

where $j = 1 \dots n$. The diameter ($diam(i)$) is the maximum path in the tree and the depth ($de(i)$) measures the maximum depth of the tree, considering node 0 as the root of the tree.

3.1 Initialization

To investigate the bias in the initialization proposed in Section 2.1, we generated 10.000 random solutions, for each problem instance, using the initialization described in Section 2.1 and then, we compared their properties to 10.000 solutions that were generated by using an unbiased encoding (Prüfer Numbers).

Table 1 shows the mean and the standard deviation of depth, diameter, minimum distance towards stars, and distance towards MST. We compared results for the node-depth initialization with an unbiased tree initialization routine using Prüfer numbers. The results for the depth and diameter show that the proposed initialization has a bias to solutions with small depths and diameters. Analyzing the results about the distances to the MST and to stars we can conclude that the proposed initialization does not have a bias towards MSTs, but a bias towards stars.

3.2 Mutation Operators

To investigate whether the mutation operators of node-depth encoding have a bias and over-represent some tree structure we randomly generated an unbiased population of 300 individuals (using Prüfer numbers) and subsequently applied mutation operators to each individual. As no selection operator was used, no selection pressure pushed the population to high-quality solutions. The mutation operators are unbiased if the statistical properties of the population did not change by applying mutation alone.

To analyze a potential bias of the mutation operators, we measured the average depth $de_{pop} = \frac{1}{N} \sum_{i=1}^{N} de(i)$, average diameter $diam_{pop} = \frac{1}{N} \sum_{i=1}^{N} diam(i)$, average distance to the MST $d_{mst-pop} = \frac{1}{N} \sum_{i=1}^{N} d_{i,MST}$, and average minimum distance to stars $d_{star-pop} = \frac{1}{N} \sum_{i=1}^{N} \min(d_{i,STAR})$ of all 300 individuals in the population. If de_{pop} increases or decreases the mutation operators are biased to high (low) depths. If $diam_pop$ increases or decreases the mutation operators are biased to high (low) diameters. If $d_{mst-pop}$ decreases, the mutation operators are biased towards the MST and if $d_{star-pop}$ decreases, the mutation operators are biased towards stars. If all the measures remain constant, the mutation operators are unbiased.

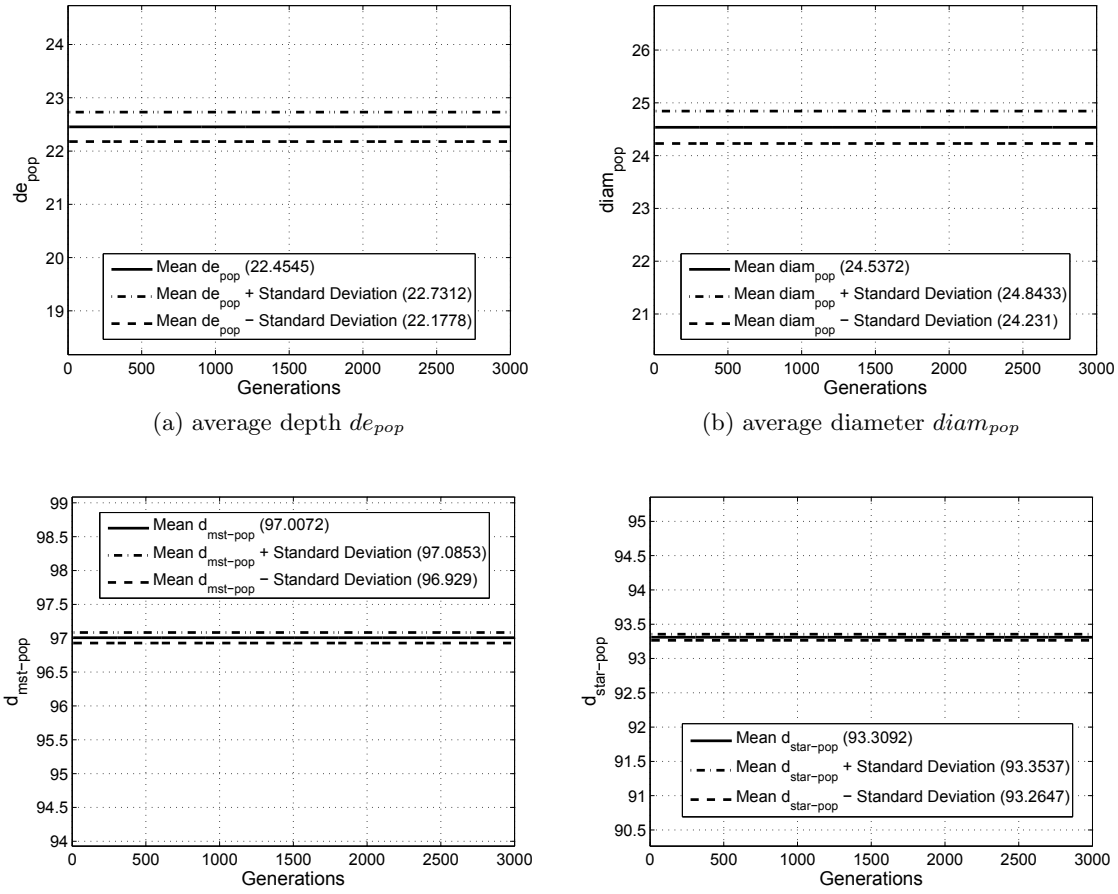
Figure 8 presents results for the bias in PAO mutation operator and shows the mean and standard deviation of de_{pop} (Fig. 8(a)), $diam_{pop}$ (Fig. 8(b)), $d_{mst-pop}$ (Fig. 8(c)), and $d_{star-pop}$ (Fig. 8(d)). Results for CAO are analogously to the results presented here and are omitted due to space restrictions. Both mutation operators are unbiased and do not modify the statistical properties of the population (de_{pop}, $diam_{pop}$, $d_{mst-pop}$ and $d_{star-pop}$ remain constant).

4. NODE-DEPTH ENCODING FOR THE BDMSTP

In a graph, the distance from a vertex v to a vertex t is the number of edges of the shortest path between v and t. The longest distance from a vertex v to any other vertex is called the eccentricity of the vertex v. The diameter of a graph is the maximum eccentricity of its vertices, *i.e.* the number of edges in the longest path of the graph.

Given an undirected connected weighted graph and a positive integer k, the bounded-diameter minimum spanning tree problem (BDMSTP) searches a spanning tree of the graph with smallest weight among all spanning trees of the graph, whose diameter is at most k edges. Formally, let $G = (V, E)$ be an undirected, connected graph, where V denotes the set of vertices and E denotes the set of edges. Given non-negative edge weights (w_e) and a positive integer k, the BDMSTP searches a spanning tree $T \subseteq E$ that minimizes $w_T = \sum_{e \in T} w_e$, subject to the constraint that the diameter of T is at most k edges ($diam(T) \leq k$). This problem is also known as the diameter-constrained minimum spanning tree problem or shallow light spanning tree problem.

As described by [23], the BDMSTP has been shown to be $\mathcal{NP}$-hard when $k \geq 4$ [7] and it has many practical applications in such areas as telecommunication networks and linear light wave network design, distributed mutual exclusion and bit compression for information retrieval.



(a) average depth de_{pop}

(b) average diameter $diam_{pop}$

(c) average distance to MST $d_{mst-pop}$

(d) average minimum distance to stars $d_{star-pop}$

Figure 8: Statistical properties of the mutation operators of the node-depth encoding. We show results for a population size $N = 300$, 3.000 iterations, and problems with $n = 100$.

For the BDMSTP instances with $k = 2$, $k = 3$, $k = n - 1$ and all edge weights the same, polynomial time algorithms are known. In the 1990s, some exact and heuristic algorithms were proposed to the BDMSTP, however, due to the exponential time complexity of these algorithms, just small instances can be solved [3]. Abdala et al. [1] proposed two new heuristics for the BDMSTP, the first heuristic calculates the MST and then transforms the MST in the BDMST, however, this heuristic is computationally expensive to small k. The second heuristic, called one-time tree construction (OTTC), imitates Prim's algorithm and subsequently chooses edges of low weight so that the insertion of an edge does not violate the constraint. The problem with OTTC is related to Euclidean instances, where it can find solutions that differs from the optimal solution because it chooses only the lowest-weight edge available when it constructs the backbone of the tree so, it is misled. A good heuristic for the BDMSTP will not prefer lower-weight edges as it builds the backbone.

Raidl and Julstrom, in 2003, proposed new technique named a randomized greedy heuristic (RGH) [18] used in a steady state evolutionary algorithm whith edge-sets (RJ-ESEA) as well as a steady state evolutionary algorithm using a permutation encoding (RJ-PEA) [11] for the BDMSTP. Both algorithms can find good solutions to the BDMSTP for large problem instances. RJ-PEA obtains better results than the RJ-ESEA. However, RJ-PEA was several times slower. Later, Julstrom [10] proposed two generational EAs. One uses random keys to represent solutions and employs two-point crossover and uniform mutation, whereas the other uses permutation coded and employs the C1 crossover proposed by Reeves [19] and swap mutation. However, these two generational evolutionary algorithms perform worse than the others for large instances.

Latterly, Singh and Gupta [23] proposed improved versions of RGH (called RGH-I), the center based tree construction (CBTC-I) which in its original form was proposed by Julstrom as a modification of RGH and the permutation coded steady state evolutionary algorithm (PEA-I). They showed that these improved versions can find better solutions for the BDMST and need less time than the first versions.

In this work an EA with the node-depth encoding (EAN) is applied to the BDMSTP in order to examine the effects of the bias in the initialization process to this problem.

4.1 Results

The EA using the node-depth encoding was implemented in C. All experiments have been performed under Linux on a Dual Intel Xeon processor with 4GB of memory. For all experiments, we used a population size of $N = 20$. The EA terminates if the best solution in the population has not been improved in the last 100.000 search steps as in the tests proposed in the literature. We study two variants of the EA using either the initialization proposed in Section 2.1 (EANR) or generating the initial solutions unbiased using Prüfer numbers (EANP).

We use the same BDMST test instances as Julstrom and Raidl [11, 18, 10]. The test problems are originally designed for the Euclidean Steiner Tree problem and are available in Beasley's OR-library[2]. For each instance, the diameter bound k is set to 5, 7, 10, 15, 25, 35, 50, 70 and 100. For $n = 50$, the maximum diameter bound is 49 and for $n = 100$, it is 99. To compare the obtained results with results presented in the literature for $n = 500$, we also use the bound $k = 20$. Results are presented for all 25 Euclidean instances with five instances for each value of $n = 50$, 100, 250, 500, and $1,000$ nodes.

Table 2 compares the performance of ENAR, JR-PEA, PEA-I, and JR-ESEA for the different test instances. We reported the best result found by each algorithm and the average number of search steps. Data for JR-PEA, PEA-I, and JR-ESEA are taken from Singh and Gupta [23]. The results show that EANR performs slightly worse than JR-PEA, PEA-I, and JR-ESEA in all instances but the number of search steps is much lower in comparison to the other approaches.

Figure 9 compares the performance of EANR and EANP over the bound k for one particular test instance (instance 3) for different problem sizes ($n = 50$, 100, 250, 500, 1000). The results for the other test instances are analogous. The plots show that only for very low bounds ($k < 15$), EANP shows higher performance than EANR. However, with increasing k, the biased initialization approach of the node-depth encoding allows EAs to find slightly better solutions.

In general, the bias of the initialization scheme of the node-depth encoding towards small diameters does not systematically increases the performance of evolutionary algorithm for the BDMSTP. This is due to the fact that the population is very small and a large number of search steps per individual is performed. Therefore, the structure of the initial solution has low impact only on the performance of the search, and the EA can not make proper use of the fact that initial solutions are biased towards low depths.

5. CONCLUSIONS

This work investigates the bias of the node-depth encoding and examines its performance for the bounded-diameter minimum spanning tree problem (BDMSTP). The node-depth encoding can be classified as a combination of direct and indirect representations for trees. It uses an explicite genotype-phenotype mapping as well as problem specific mutation operators.

The analysis of the bias shows that the initialization method of the node-depth encoding is biased towards trees with low diameters and low depths. Furthermore, it over-represents star-like structures. In contrast, the mutation operators are unbiased. When applying EAs using the node-depth encoding to BDMSTPs, the bias of the initialization method towards trees with low diameters does not lead to better results in comparison to using an unbiased initialization method.

[2] `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

6. ACKNOWLEDGMENTS

The authors acknowledgments are to FAPESP by the research scholarship and the Department of Information System and Business Administration in the University of Mainz, where part of this work was developed.

7. REFERENCES

[1] A. Abdalla and N. Deo. Random-tree diameter and the diameter-constrained mst. *Int. J. Comput. Math.*, 79(6):651–663, 2002.

[2] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees. In *SAC '94: Proceedings of the 1994 ACM symposium on Applied computing*, pages 242–246, New York, NY, USA, 1994. ACM Press.

[3] N. Achuthan, L. Caccetta, P. Caccetta, and G. J.F. Algorithms for the minimum weight spanning tree with bounded diameter problem. *Optimization: techniques and applications*.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, McGraw-Hill Book Company, 2000.

[5] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[6] A. Delbem, A. de Carvalho, C. A. Policastro, A. K. Pinto, K. Honda, and A. C. Garcia. Node-depth encoding for evolutionary algorithms applied to network design. In K. Deb and et al., editors, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, Lecture Notes in Computer Science, pages 678–687, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

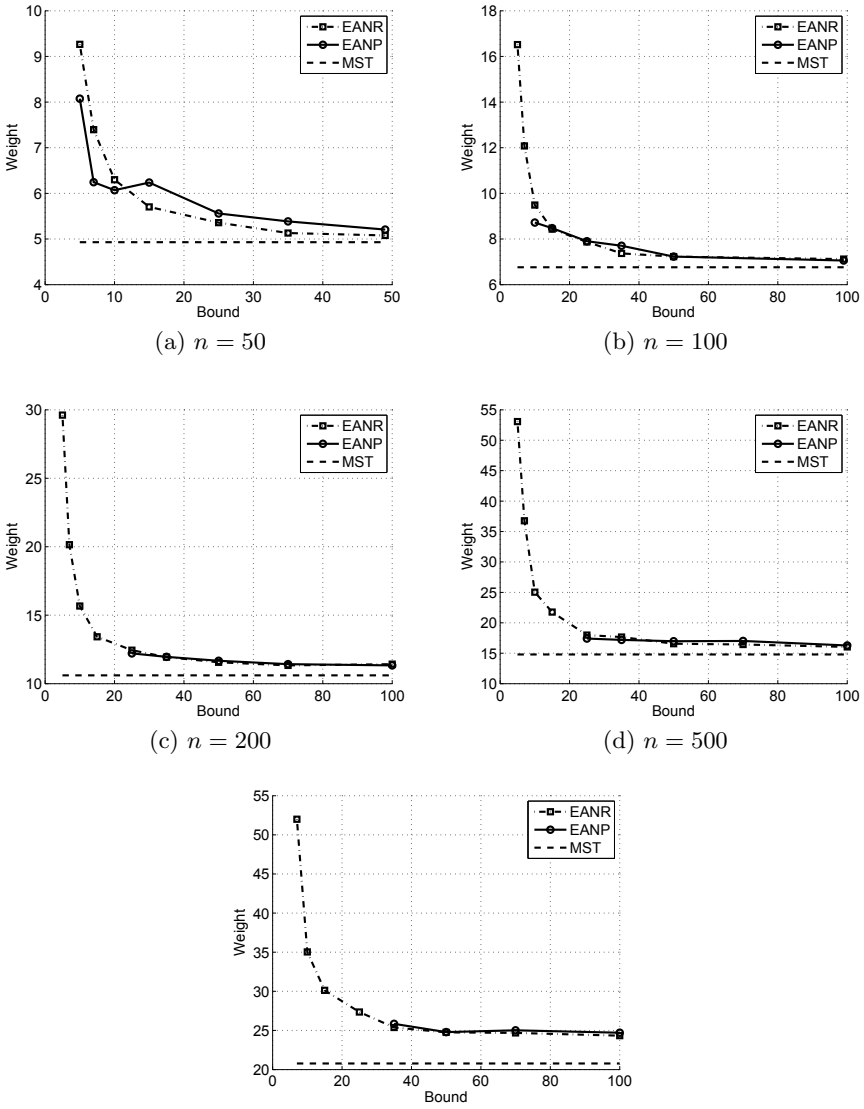
[7] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[8] M. Gruber, J. van Hemert, and G. Raidl. Neighborhood searches for the bounded diameter minimum spanning tree problem embedded in a vns, ea, and aco. volume 2, pages 1187–1194, Seattle, USA, 2006. ACM.

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

[10] B. A. Julstrom. Encoding bounded-diameter spanning trees with permutations and with random keys. In K. Deb and et al., editors, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102, pages 1272–1281, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[11] B. A. Julstrom and G. R. Raidl. A permutation-coded evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In A. M. Barry, editor, *GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 2–7, Chigaco, 11 July 2003. AAAI.



(a) $n = 50$ (b) $n = 100$

(c) $n = 200$ (d) $n = 500$

(e) $n = 1000$

Figure 9: Performance of EANR and EANP for Euclidean instance 3 and different problem sizes.

Table 2: Results of EANR, JR-PEA, PEA-I, JR-ESEA, and Average number of evaluations required by the algorithms.

Instance			EANR		JR-PEA		PEA-I		JR-ESEA	
n	k	Number	Best	Evaluations	Best	Evaluations	Best	Evaluations	Best	Evaluations
50	5	1	8.81	108,678	7.60	8,862,240	7.60	7,503,360	7.60	8,147,280
50	5	2	8.62	110,208	7.75	6,709,207	7.75	3,201,280	7.68	8,736,720
50	5	3	7.98	110,125	7.25	7,614,000	7.25	7,821,760	7.24	6,700,560
50	5	4	8.52	107,717	6.62	9,768,720	6.62	3,784,000	6.59	7,531,680
50	5	5	8.72	107,790	7.39	8,248,560	7.39	7,976,640	7.32	8,381,760
100	10	1	8.66	215,816	7.77	21,319,440	7.76	33,043,840	8.00	45,366,240
100	10	2	9.50	198,888	7.88	19,814,400	7.85	26,686,080	8.10	49,413,840
100	10	3	9.43	195,152	7.93	21,232,080	7.90	43,832,000	8.22	42,250,320
100	10	4	9.36	209,813	8.00	22,318,320	7.98	36,442,560	8.27	39,154,080
100	10	5	9.41	202,294	8.16	21,536,400	8.16	36,610,240	8.48	39,516,240
250	15	1	13.74	947,338	12.41	66,956,160	12.24	123,583,040	12.93	113,232,720
250	15	2	13.72	943,624	12.20	74,723,760	12.04	117,557,120	12.86	111,851,280
250	15	3	13.40	986,246	12.12	74,327,760	12.03	130,052,480	12.69	111,508,320
250	15	4	13.96	941,691	12.64	73,230,000	12.42	136,202,560	13.22	106,187,040
250	15	5	13.74	1,002,782	12.37	75,841,440	12.28	112,577,280	13.02	119,388,000
500	20	1	18.58	3,278,443	17.10	170,882,640	16.96	261,824,640	18.33	126,635,760
500	20	2	18.83	3,065,966	16.97	176,861,040	16.81	266,788,800	18.17	156,482,160
500	20	3	18.93	3,145,114	17.13	179,102,160	16.89	285,520,640	18.33	121,035,600
500	20	4	18.57	3,221,440	17.09	178,135,440	16.96	279,102,400	18.32	157,169,040
500	20	5	18.70	3,086,630	16.79	169,275,120	16.58	271,399,680	17.80	155,555,520
1,000	25	1	26.81	8,462,703	-	-	23.97	565,930,240	26.13	120,251,280
1,000	25	2	26.44	9,062,438	-	-	23.70	488,745,920	26.14	111,455,280
1,000	25	3	26.09	8,592,669	-	-	23.61	570,712,640	25.47	133,104,240
1,000	25	4	26.12	8,354,454	-	-	24.04	553,896,000	26.13	121,929,120
1,000	25	5	26.15	8,970,823	-	-	23.75	537,930,880	25.91	155,476,320

[12] J. D. Knowles and D. Corne. A new evolutionary approach to the degree-constrained minimum spanning tree problem. *IEEE Trans. Evolutionary Computation*, 4(2):125–134, 2000.

[13] C. C. Palmer. *An approach to a problem in network design using genetic algorithms*. PhD thesis, Brooklyn, NY, USA, 1994.

[14] S. Picciotto. *How to Encode a Tree*. PhD thesis, University of California, 1999.

[15] H. Prüfer. Neuer beweis eines satzes uber permutationen. *Arch. Math. Phys.*, 1918.

[16] 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*, 13(4):441–475, 2005.

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

[18] G. R. Raidl and B. A. Julstrom. Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 747–752, New York, NY, USA, 2003. ACM.

[19] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Comput. Oper. Res.*, 22(1):5–13, 1995.

[20] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag, 2006.

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

[22] F. Rothlauf and C. Tzschoppe. On the bias and performance of the edge-set encoding, 2005.

[23] A. Singh and A. K. Gupta. Improved heuristics for the bounded-diameter minimum spanning tree problem. *Soft Comput.*, 11(10):911–921, 2007.