Node-depth Phylogenetic-Based Encoding, a Spanning-Tree Representation for Evolutionary Algorithms. Part I: Proposal and Properties Analysis

Article *in* Swarm and Evolutionary Computation · May 2016

DOI: 10.1016/j.swevo.2016.05.001

6 authors, including:

Telma Woerle Lima
Universidade Federal de Goiás
37 PUBLICATIONS **268** CITATIONS
SEE PROFILE

Alexandre C. B. Delbem
University of São Paulo
223 PUBLICATIONS **2,263** CITATIONS
SEE PROFILE

Anderson Soares
Universidade Federal de Goiás
84 PUBLICATIONS **428** CITATIONS
SEE PROFILE

J.B. Augusto London Jr
University of São Paulo
103 PUBLICATIONS **998** CITATIONS
SEE PROFILE

Node-depth Phylogenetic-Based Encoding, a Spanning-Tree Representation for Evolutionary Algorithms. Part I: Proposal and Properties Analysis

Telma Woerle de Lima[1], Alexandre Cláudio Botazzo Delbem[2], Anderson da Silva Soares[1], Fernando Marques Federson[1], João Bosco Augusto London Junior[2], Jeffrey Van Baalen[3]

[1] *Universidade Federal de Goiás, Instituto de Informática, Brazil*
[2] *University of São Paulo, São Carlos School of Engineering, Brazil*
[3] *University of Wyoming, Computer Science Departament, WY, U.S.A*

Abstract

Representation choice and the development of search operators are crucial aspects of the efficiency of Evolutionary Algorithms (EAs) in combinatorial problems. Several researchers have proposed representations and operators for EAs that manipulate spanning trees. This paper proposes a new encoding called Node-depth Phylogenetic-based Encoding (NPE). NPE represents spanning trees by the relation between nodes and their depths using a relatively simple codification/decodification process. The proposed NPE operators are based on methods used for tree rearrangement in phylogenetic tree reconstruction: subtree prune and regraft; and tree bisection and reconstruction. NPE and its operators are designed to have high locality, feasibility, low time complexity, be unbiased, and have independent weight. Therefore, NPE is a good choice of data structure for EAs applied to network design problems.

Keywords: Network design problems, tree representations, dynamic data structures, evolutionary algorithms

Evolutionary algorithms (EAs) have been applied to the network design problem whose solutions are spanning trees in a graph model. EAs generate and manipulate a set of data structures to represent candidate solutions. Representation choice and the development of search operators applied to it are crucial aspects of the efficiency of EAs in combinatorial problems [1].

The representation is the solution's genotype and the decoded solution is the phenotype. The genotype-phenotype relation should have some properties [2, 1] among them we highlight:

1. Locality: refers to the correspondence between genotypic neighbors to phenotypic neighbors;
2. Bias: refers to the probability of a tree encoding representing (or generating) a solution. Biased encodings and search operators are appropriate if it is known a priori that optimal solutions are similar to over-represented solutions. In contrast, unbiased encodings and search operators should be used when no problem-specific knowledge is available;
3. Feasibility: is true when a representation encodes only feasible solutions and search operators generate only feasible solutions;
4. Time complexity: the time complexities of decoding, evaluating, recombining, and mutating genotypes should be small.

Several representations have been proposed to improve the performance of EAs on network problems. These encodings are classified into direct and indirect representations [1], although some spanning tree representations have characteristics of both types.

In a direct representation, a tree is encoded by the set of edges and the search operators are applied directly to the set of edges. Along with these representations specific operators must be developed. An example of direct representation is Edge-Sets [3, 4]. Edge-Sets has time complexity $O(n)$ and its search operators can be divided into non-heuristic and heuristic [4]. Edge-Sets with heuristic operators yields better results in EAs. However, Tzschoppe et. al. [5] have shown that the heuristic operators of Edge-Sets are biased to the minimum spanning tree, i.e. the solutions generated are close to the minimum spanning tree. Our previous studies [6, 7] have also shown the difficulty of developing unbiased operators for network design problems.

Indirect representations use a genotype-phenotype mapping to represent the solutions. Example indirect representations are Prüfer Numbers [8] ([9, 10]), Network Randon Keys (NetKeys) [11], Characteristic Vector [12, 13] and others [14, 15, 16, 2]. Despite having been largely used to encode trees in EAs, research has shown that Prüfer Numbers have low locality [17]. Characteristic Vector encodes a tree using a binary string where each position indicates whether or not an edge is present in the tree. However, when the

standard search operators are applied, they can produce infeasible solutions, with cycles or disconnects. NetKeys is an alternative to Characteristic Vector that avoids generating infeasible solutions by using a permutation vector. However, the process to insure only feasible solutions has time complexity $O(mlog(m))$, where m is the number of edges in the graph [11].

There are some representations that have characteristics of both types, i.e., a genotype-phenotype mapping and specific search operators, such as the Node Depth Degree Representation (NDDR) [18] and others [19, 20, 21, 22, 23, 24, 25]. NDDR was designed to encode spanning trees and forests in complete and non-complete graphs. The authors have shown that NDDR has time complexity $O(\sqrt{n})$. NDDR encodes a tree using the relations among nodes, their depths, and their degrees in a rooted tree. However, NDDR uses relatively complex procedures in the search operators and encoding to insure time complexity for any spanning tree based problem.

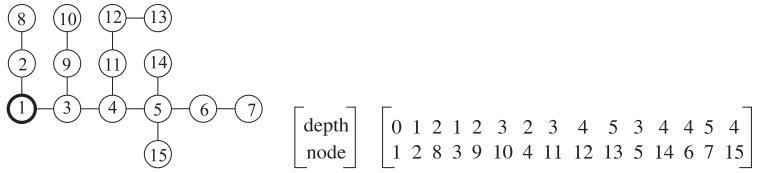
This paper proposes an encoding called Node-depth Phylogenetic-based Encoding (NPE) to represent spanning trees[1] by the relation between nodes and their depths. NPE stores a tree using a simple codification/decodification scheme, as in the usual data structure for binary trees that uses an array [26]. NPE operators work similarly to the tree rearrangement methods used in Phylogenetic Tree Reconstruction (PTR), in order to make the process easier than with NDDR operators. The NPE operators have time complexity $O(n)$, as, for example, Edge-Sets [4]. NPE and its operators are designed to have high locality, be unbiased, be feasible, have low time complexity, and be weighted independently. Therefore, with an unique set of features, NPE is a good choice of data structure for EAs applied to network design problems.

The following sections are organized as follows. Section 1 introduces NPE. Section 2 describes two tree rearrangement methods used in phylogenetic tree reconstruction and section 3 describes the rearrangement methods as evolutionary operators for NPE. Section 4 analyzes the properties of locality, bias, feasibility, and time complexity of the NPE encoding and, finally, section 5 presents the conclusions and final remarks.

[1] A forest must be rearranged as an unique tree by inserting additional nodes and edges connecting all trees of the forest [26].

1. Node-depth Phylogenetic-based Encoding

The NPE of a graph tree consists of an ordered list of pairs (n_x, d_x), where n_x is a node and d_x is its depth[2]. The pairs are inserted in the order produced by a Depth-First Search (DFS) [26]. When node n_x is visited, pair (n_x, d_x) is inserted in the list.



(a) A spanning tree of G_1.

(b) NPE of the spanning tree.

Figure 1: A spanning tree of complete graph G_1 and the corresponding NPE.

Figure 1 illustrates a spanning tree of a complete graph G_1 (Figure 1(a)) and the corresponding NPE considering node 1 as the root node of the spanning tree (Figure 1(b)).

The process for decoding an NPE traverses the corresponding array from the second position to the last position, and assigns an edge between two nodes n_i and n_j from an NPE N, if $i < j$ and $d_j - d_i = 1$, (where d_i is the depth of node n_i at position i) and $j - i$ is minimal, i.e. n_i is the left nearest node that can connect to n_j. Algorithm 1.1 presents the decoding procedure, which is $O(|N|)$, where $|N|$ is the size of NPE N. The pseudocode uses an auxiliary array $roots_{d_i}$ for storing the last visited node with depth d_i for efficiency in finding n_i from n_j.

2. Phylogenetic Tree Reconstruction - Rearrangement Methods

The main aim of Phylogenetic tree reconstruction (PTR) is to describe evolutionary relationships in terms of relative recency of common ancestry in a tree structure [27]. A phylogenetic tree has the property of a binary tree data structure, i.e. each node has at most two children [26].

There are several methods to reconstruct the phylogenetic tree of a group of alignment sequences. These methods can be classified into three main

[2]The depth of a node "x" in a tree is the length (number of edges) of the unique path from the root node of the tree to the node "x".

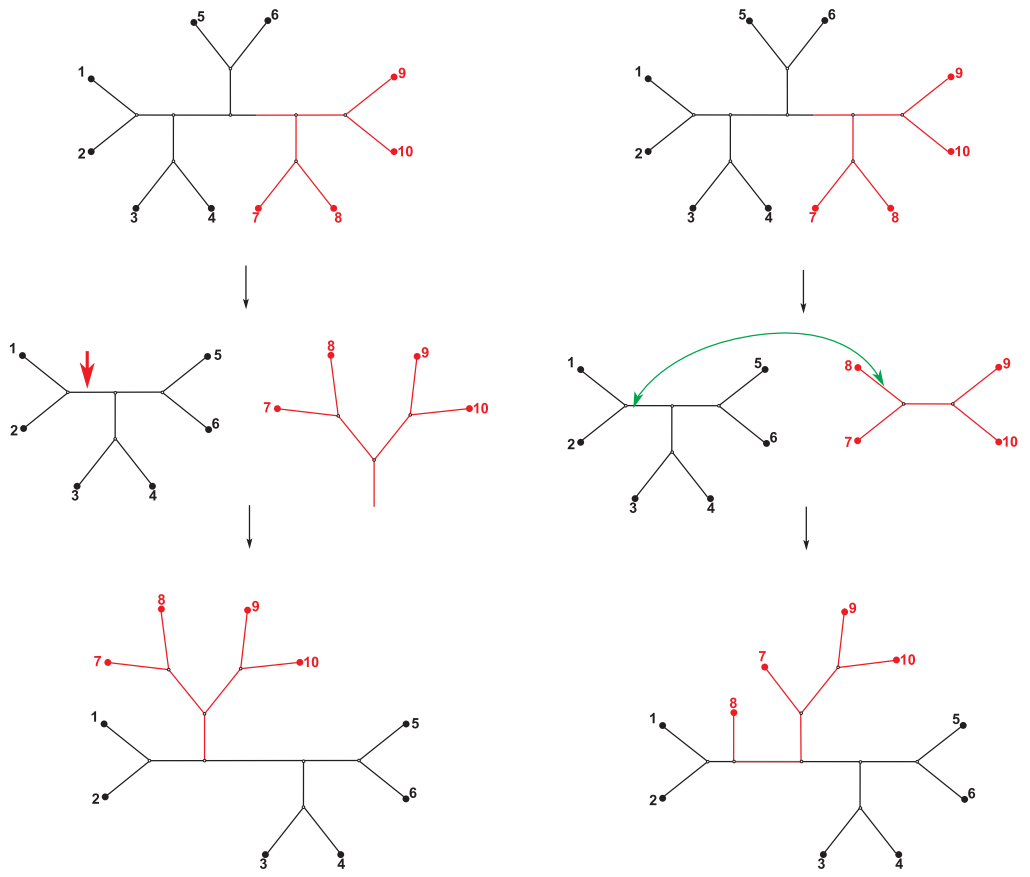
Algorithm 1.1 Pseudocode for decoding NPEs.

Algorithm Decoding
//Let N be an NPE and let N_i be the pair (n_i, d_i) at position i of N
//Let $|N|$ denote the size of N
//Let T be a tree, let $V(T)$ and $E(T)$ be the node and edge sets of T respectively
//Let *roots* be an array of size $|N|$ to store subtree roots
1. $V(T) = \emptyset$
2. $E(T) = \emptyset$
3. if $|N| > 0$
    4. include node n_0 in $V(T)$
5. else return 0
6. $roots_{d_0} = n_0$
7. $j = 1$ //(second position in the NPE of tree T)
8. **While** $j < |N|$ **do**
    9. include node n_j in $V(T)$
    10. $roots_{d_j} = n_j$
    11. $k = d_j - 1$ //(k is the depth of node j minus 1)
    12. include edge $(roots_k, n_j)$ in $E(T)$
    13. increment j

categories: parsimony, distance, and likelihood methods [27]. Most of these methods need to rearrange the generated trees. There are several methods available for rearranging trees in PTR; subtree transfer operations being one example. In this paper we use two of these methods: Subtree Prune and Regraft (SPR) and Tree Bisection and Reconnection (TBR) to formulate the search operators of NPE.

A new tree is obtained in SPR by pruning a subtree rooted at node x from the original tree and regrafting the pruned subtree into another position of the tree. All available positions in the tree are tested. Then, the position that improves the used metric the most is chosen for regrafting the subtree. Moreover, all possible subtree prunings are tested. The combination of prunning and regrafting positions that improves the used metric the most is chosen. Once the pruned subtree has been regrafted, if necessary, SPR applies a forced contraction, i. e. the removal of an edge and contraction of nodes, in order to maintain a binary tree [27]. Figure 2(a) presents an example of SPR operation.

A new tree is obtained in TBR by the removal of an edge from the original tree yielding the two subtrees (T_1,T_2). Next, the subtrees are reconnected with a new edge between the middle point of some edge in T_1 and an edge in T_2 (see Figure 2(b)). If necessary, a forced contraction, i. e. the removal of



(a) SPR. (b) TBR.

Figure 2: SPR and TBR example.

an edge and contraction of nodes, is used to preserve the binary characteristic of the new tree, i.e. with at most two children [27]. Similarly to SPR, TBR tests all possible reconnections between T_1 and T_2, and then chooses the one that improves the used metric the most. TBR can produce changes in a tree larger than SPR when the created edge connects T_2 in a position different from that of the removed edge.

3. NPE Operators

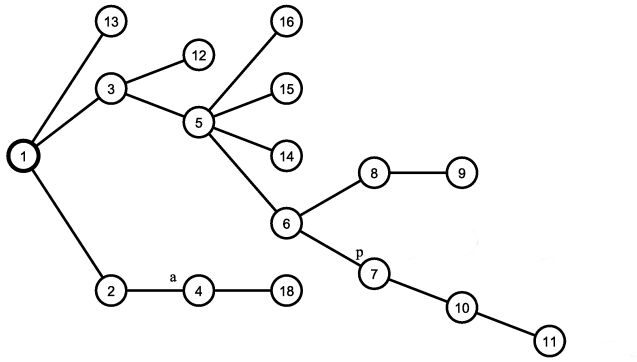
Based on the rearrangement methods summarized in Section 2, we propose two operators to manipulate NPEs. The proposed operators, namely

Subtree Pruning and Regrafting on NPE (SPRN) and Tree Bisection and Reconnection on NPE (TBRN), generate a new spanning tree T' of a graph G from the spanning tree T of G. SPRN and TBRN work like mutation operators in standard EAs. Unlike the reconstruction methods from phylogeny, SPRN and TBRN are able to work with trees having nodes with degree higher than two.

3.1. Subtree Pruning and Regrafting Operator on NPE - SPRN

			a					$\mathbf{R_p}$									
i	0	1	**2**	3	4	5	6	**7**	**8**	**9**	10	11	12	13	14	15	16
n	1	2	**4**	18	3	5	6	**7**	**10**	**11**	8	9	14	15	16	12	13
d	0	1	**2**	3	1	2	3	**4**	**5**	**6**	4	5	3	3	3	2	1

(a) An NPE N for SPRN application.



(b) The spanning tree correspond to NPE N.

Figure 3: Tree origin for SPRN example

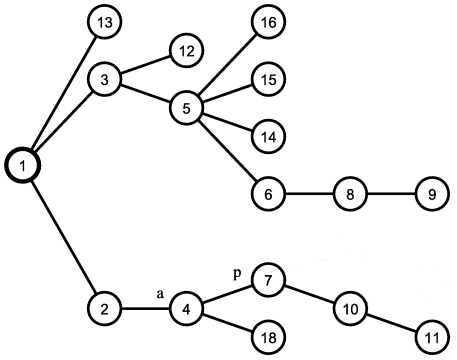
SPRN has two arguments: the prune node p, which is the root of the subtree to be transferred; and an adjacent node a, which is a node where the pruned subtree rooted in p will be regrafted. SPRN is presented in Algorithm 3.1, which assumes:

(i) NPE has been implemented by array N;
(ii) i_p is the index of node p in N and i_a the index of node a in N;
(iii) R_p is the range $(i_p - i_l)$ of indexes in N corresponding to the subtree rooted at node p.

R_p corresponds to node p and to consecutive nodes x in array N such that $d_x > d_p$, where d_x and d_p are the depths of nodes x and p, respectively. The search scans the array N sequentially while $d_x > d_p$ holds;

			a		**R_p**												
i	0	1	**2**	**3**	**4**	**5**	6	7	8	9	10	11	12	13	14	15	16
n	1	2	**4**	**7**	**10**	**11**	18	3	5	6	8	9	14	15	16	12	13
d	0	1	**2**	**3**	**4**	**5**	3	1	2	3	4	5	3	3	3	2	1

(a) NPE N' obtained using SPRN.



(b) The spanning tree corresponding to NPE N'.

Figure 4: Result of SPRN application

Figure 3(a) presents the NPE N of the spanning tree illustrated in Figure 3(b) assuming: node 1 as the root node; node 4 as node a ($i_a = 2$); and node 7 as node p ($i_p = 7$). The Range $R_p = [7, \ldots, 9]$ is also shown in Figure 3(a).

After finding the Range R_p, SPRN copies positions $i = 0, \ldots, 2(i_a)$ from N to N', which will store the new spanning tree. Afterward, positions in R_p are copied to N' on the right side of node a. The depths in R_p are updated using the equation presented in Step 3 of Algorithm 3.1. Finally, positions $i = 3, \ldots, 16$ from N are copied into N' skipping the positions of $R_p = [7, \ldots, 9]$. N', can be seem in Figure 4(a), and the corresponding spanning tree in Figure 4(b).

3.2. Tree Bisection and Reconstruction Operator (TBRN)

TBRN has three arguments: the prune node p, the root node r of the subtree to be transferred, and an adjacent node a. Observe that from TBRN the subtree will be reconnected by inserting a new edge (r, a) in the spanning tree.

Algorithm 3.1 -Pseudocode for SPRN

//Let N be an NPE and $|N|$ the size of N
//Let R_p be the range in N of indexes corresponding to the subtree pruned at p
//Let N' be the new NPE
//Let p be the pruned node and i_p its index in N
//Let a be the regrafted node and i_a its index in N
//Let d_i be the depth of node i
//Determine the range R_p in N corresponding to the subtree pruned at p
1. Traverse N from i_p until the first position $i_x > i_p$, where $d_x \leq d_p$, has been found
2. Copy the positions from N from $i = 0$ to i_a, skipping R_p if $i_p < i_a$, and store them into N'
//Insertion of R_p at the right of node a in N'
3 Recalculate the depths of nodes in R_p as follows:

$$d_x = d_x - d_p + d_a + 1$$

4 Copy the positions from N from $i_a + 1$ to $|N| - 1$ into N' skipping R_p

TBRN is presented in Algorithm 3.2, which assumes:

(i) i_p, i_r, and i_a are the indexes of nodes p, r, and a in N, respectively;
(ii) R_p is the range $(i_p - i_l)$ of indexes in N corresponding to the subtree rooted at node p (as previously defined);
(iii) f_x is the "father" of node x, i.e. the adjacent node of x with $d_{f_x} = d_x - 1$ and that $i_x - i_{f_x}$ is minimal.

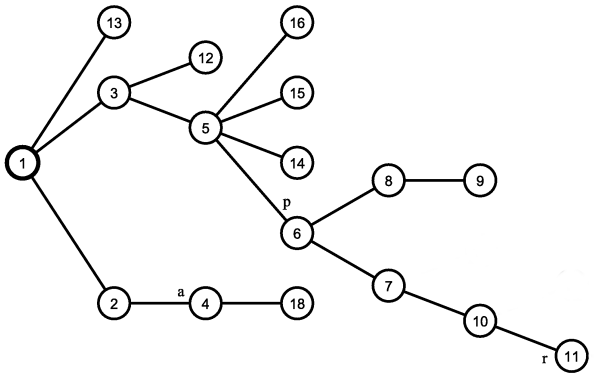
Observe that SPRN maintains the root of the pruned subtree from N, when this subtree is grafted into N'. Also, TBRN changes the root of the transferred subtree, that is, it reorganizes the pruned subtree so that r becomes its root in N'. In other words, TBRN changes the flow in the pruned subtree and SPRN keeps the flow the same. For some network design problems, like vehicle routing and energy restoration problems, the change of the flow influences the network properties and evaluation. Algorithm 3.2 shows the steps of TBRN.

Figure 5(a) presents the NPE N of the spanning tree illustrated in Figure 5(b) assuming: node 1 as the root node; node 6 as node p ($i_p = 6$); node 11 as node r ($i_r = 9$); and node 4 as node a ($i_a = 2$). The Range $R_p = [6, \ldots, 11]$ is also presented in Figure 5(a).

After finding the Range R_p, TBRN reorganizes the subtree R_p so that r becomes the new root of this subtree (Step 2 of Algorithm 3.2). Then, the reorganized R_p, called R_r, is copied to the temporary NPE array N_{tmp}, and

			a				**R_p**										
i	0	1	**2**	3	4	5	**6**	**7**	**8**	**9**	**10**	**11**	12	13	14	15	16
n	1	2	**4**	18	3	5	**6**	**7**	**10**	**11**	**8**	**9**	14	15	16	12	13
d	0	1	**2**	3	1	2	**3**	**4**	**5**	**5**	**4**	**5**	3	3	3	2	1

(a) NPE N to TBRN application.



(b) The spanning tree represented by NPE N.

Figure 5: Tree origin for TBRN example.

the node depths are recalculated by the equations defined on Steps 3 and 4 of Algorithm 3.2. The resulting N_{tmp} can be seen in Figure 6(a). Afterwards, positions $i = 0, \ldots, 2$ are copied from N to N'. Then, N_{tmp} is copied to N' on the right side of node a. Finally, positions $i = 3, \ldots, 16$ are copied from N into N', avoiding the positions of $R_p = [6, \ldots, 11]$. N', can be seen in Table 6(b), and the corresponding spanning tree in Figure 6(c).

3.3. Determination of nodes p, r, and a

To find node p, we randomly select an index i from $i = 1 \ldots (|N| - 1)$. Then, we call i of i_p and p the node of this position in the NPE. To find r, we need to detect R_p with Step 1 of the algorithm. Then we randomly select a position from R_p and we call it i_r. The selection of node a, which must be adjacent to p (SPRN) or r (TBRN), is performed as follows. In complete graphs we randomly choose a position i_a from $[1 \ldots (|N| - 1)] - [R_p]$. With non-complete graphs we need a special strategy to find node a that avoids generating infeasible solutions. A possible strategy is to verify whether the randomly selected node is adjacent to node p/r. If node a is infeasible, a limited number of attempts is made to choose a new set of nodes that is

Algorithm 3.2 -Pseudocode for TBRN.

//Let N be an NPE and $|N|$ the size of N
//Let R_p be the range in N of indexes corresponding to the subtree rooted at p
//Let N_{tmp} be an auxiliary NPE to store the subtrees pruned
//Let N' be the new NPE
//Let p be the pruned node and i_p its index in N
//Let r be the new root of the pruned subtree and i_r its index in N
//Let a be the reconstructed node and i_a its index in N
//Let d_i be the depth of node i
//Determine range R_p in N corresponding to the subtree pruned at p
1 Traverse N from i_p until the first position $i_x > i_p$, where $d_x \leq d_p$, has been found
//Reorganization of R_p so that r is the root of the subtree
2 Determine range R_r of nodes rooted at r (as step 1 for $p = r$)
3 Copy the nodes of R_r from N to N_{tmp} and recalculate the depths as follows:

$$d_i = d_i - d_r + d_a + 1$$

4 Let $x = r$, copy R_{f_x} without R_x, where f_x is the father of x in R_x, to the end of N_{tmp}. Recalculate the depths as follows:

$$d_i = d_i - d_{f_x} + d_x + 1$$

5 Repeat step 4 for $x = f_x$ until $f_x = p$.
6 Copy the positions from N in N' starting at $i = 0$ until i_a, skipping R_p if $i_p < i_a$
7 Insert N_{tmp} on the right of node a in N'
8 Copy the positions from N from $i_a + 1$ to $|N| - 1$ in N' avoiding R_p

feasible.

The first choice of nodes p, r and a can also violate some problem restrictions. In this case, a problem specific strategy must be chosen to select the set of adequate nodes. It is important that this strategy preserves the average time complexity of SPRN and TBRN.

4. NPE Properties

In this section we evaluate NPE according to some criteria used in the literature to analyze EA encodings: bias, locality, feasibility and time complexity.

4.1. Evaluation of bias of NPE operators

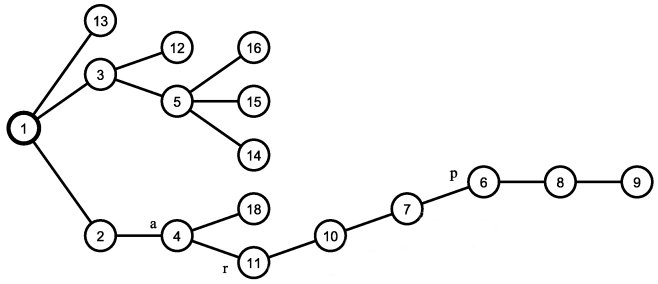
In an unbiased encoding with unbiased operators, all possible phenotypes are encoded and generated uniformly. Moreover, the application of the search

	$\mathbf{R_{11}}$	$\mathbf{R_{10}}$	$\mathbf{R_7}$	$\mathbf{R_6}$		
i	0	1	2	3	4	5
n	11	10	7	6	8	9
d	3	4	5	6	7	8

(a) NPE N_{tmp}.

			a	$\mathbf{R_p}$													
i	0	1	2	**3**	**4**	**5**	**6**	**7**	**8**	9	10	11	12	13	14	15	1
n	1	2	4	**11**	**10**	**7**	**6**	**8**	**9**	18	3	5	14	15	16	12	13
d	0	1	2	**3**	**4**	**5**	**6**	**7**	**8**	3	1	2	3	3	3	2	1

(b) NPE N' obtained using TBRN.



(c) Spanning tree corresponding to N'.

Figure 6: Result of TBRN application.

operator without selection does not modify the statistical properties of the population [5].

We analyze the bias of NPE operators for tree problems with $n = 20$, 50, 100 and 250 nodes. We use the first instances of Euclidean graphs available for the Euclidean Steiner tree problem in the *OR-Library*[3]. To investigate the bias, the following measures are used: distance from the Minimum Spanning Tree (MST), distance from star trees, and depth on average. Tzschoppe et. al. [5] used the distance from the MST to evaluate the bias on the Edge-Sets representation. And de Lima et. al. [6] used distance from the MST, distance from star trees, depth and diameter as metrics to evaluate the bias of the tree representation. In addition to these metrics being used in other research, they can indicate, in case of bias, that a proposed encoding is more adequate. For example, when the representation has a bias towards MST it

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

is more suitable for problems like the optimal communication spanning tree whose solution is close to the MST [5]. If the encoding is biased to small depths, it can produce better solutions for problems with depth constraints, like wireless backhaul network design [28].

The distance between two spanning trees is calculated by the Hamming distance of trees [29]:

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

where $l_{u,v}$ is 1 if an edge from u to v exists in tree i , and 0 if it does not exist. The distance from the MST is calculated using the Hamming distance with tree j being the MST of the graph.

The minimum distance to a star is calculated as

$$dist_{i,STAR} = min(dist_{i,star_j}).$$

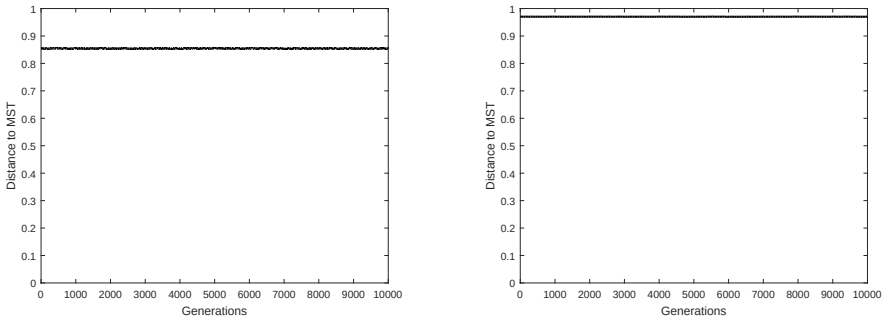
where $j = 1 \ldots n$ and $dist_{i,star_j}$ is the Hamming distance from an arbitrary tree to a star tree with center in node j.

Depth measures the maximum depth of the tree, considering the root node to have depth 1. To analyze potential bias of the NPE operators, we measured the following metrics, with K the number of individuals.

- Average depth $de_{pop} = \frac{1}{K} \sum_{i=1}^{K} depth_i$;
- Average distance from MST $dist_{pop-MST} = \frac{1}{K} \sum_{i=1}^{K} dist_{i,MST}$;
- Average minimum distance to stars $dist_{pop-STAR} = \frac{1}{K} \sum_{i=1}^{K} .dist_{i,STAR}$

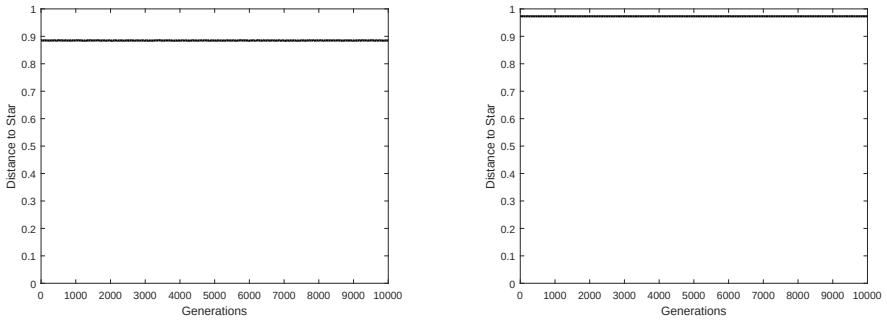
The initial population with 1,000 individuals was randomly generated with an unbiased method (using the Prüfer Numbers). In the experiments to investigate the bias of NPE operators, we execute 30 trials for each graph with 10,000 generations for each operator. All individuals in the population generate a new solution. As no selection operator was used, no selection pressure pushed the population to high-quality solutions.

If de_{pop} increases SPRN or TBRN is biased to high depths, and if it decreases the operators are biased to low depths. If $dist_{pop-MST}$ decreases, SPRN or TBRN are biased towards the MST. And if $dist_{pop-STAR}$ decreases, the operators are biased towards stars. If all the measures remain constant, the operators are unbiased.



(a) 20 nodes graph and SPRN. (b) 100 nodes graph and TBRN.

Figure 7: Average Distance to the MST ($dist_{pop-MST}$).



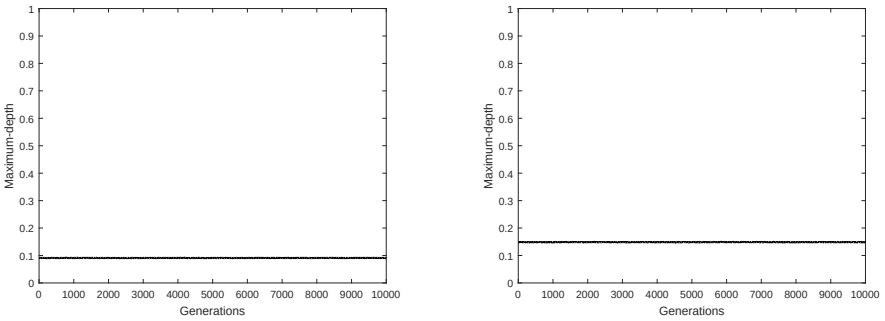
(a) 50 nodes graph and SPRN. (b) 250 nodes graph and TBRN.

Figure 8: Average Mininum Distance to Stars($dist_{pop-STAR}$).

Figures 7(a), 8 and 9 present the results normalized by the number of nodes in the graph for the properties $dist_{pop-MST}$, $dist_{pop-STAR}$ and de_{pop} with SPRN and TBRN. Results for the other graphs are analogous to the results presented here. Both operators are unbiased and do not modify the statistical properties of the population ($dist_{pop-MST}$, $dist_{pop-STAR}$ and de_{pop} remain constant).

4.2. Evaluation of NPE locality

Previous studies have shown that an encoding with high locality does not change the problem complexity, however, an encoding with low locality can [1]. In other words, low locality can change a deceptive problem into an



(a) 100 nodes graph and SPRN. (b) 250 nodes graph and TBRN.

Figure 9: Average of Maximum Depth (de_{pop}).

easy one or can change a easy problem into a deceptive one but high locality will keep a problem easy or difficult to solve. For the analysis of the locality of an encoding, a distance measure between individuals must be defined for both phenotypic and genotypic spaces. For the NPE locality study, we use Hamming distance between two trees for the phenotypic space [29] and the distance between two NPE arrays (genotypic space) is the number of pairs (n_x, d_x), where the node or the depth is different.

In order to analyze NPE locality we use the measurement d_m proposed by Rothlauf [1]:

$$d_m = \sum_{d^g_{x,y}=d^g_{min}} d^p_{x,y} - d^p_{min}$$

where $d^g_{x,y}$ is the genotypic distance between x^g (genotypic of individual x) and y^g (genotypic of individual y); $d^p_{x,y}$ is the phenotypic distance between x^p (phenotype of individual x) and y^p (phenotype of individual y) and d^g_{min} and d^p_{min} are the minimum distances between two neighbor genotypes and phenotypes, respectively. $d^p_{x,y} - d^p_{min}$ is the difference between the distance of a new tree to its ancestor and the minimal distance between trees. In d_m only trees resulting from the minimal change in the genotype are included in the sum. This means that when d_m is greater than zero there is at least one tree resulting from a minimal change in the genotype that is not a neighbour of its ancestor in genotypic space. When $d_m = 0$, all neighbour's genotypes correspond to neighbour's phenotypes, and, the locality is perfect. The smaller the value of d_m, the higher the locality of the representation. When d_m has a high value it means that small steps in the genotype correspond to large

steps in the phenotype and the locality is low.

The smallest distance in phenotypic space between two different spanning trees is one. The minimal modification in genotypic space for an NPE is one. This minimal difference in an NPE array corresponds to a change in the depth of one position. Therefore, $d^p_{min} = 1$ and $d^g_{min} = 1$. Moreover, $d^p_{x,y} = 1$, since the modification in the depth of a node p is the result of pruning and regrafting this node. In other words, this modification is the result of a SPRN with node p in position k and node a in position $k-1$ of the NPE. Analyzing SPRN and TBRN we can observe that both operators consist of the removal of one edge and the insertion of a new one. Thus, $d_m = \sum_{d^g_{x,y}} 1 - 1 = 0$ for NPE. This means that a small step in the genotypic space always corresponds to a small step in the phenotypic space. Therefore, NPE has high locality.

4.3. Evaluation of NPE feasibility

Since an NPE encodes a solution as an array of pairs (n_x,d_x) arranged in proper order without repetition, it corresponds only to solutions representing a tree if the NPE is decoded by Algorithm 1.1. The decoding process guarantees connected trees since the proper order does not allow a node inserted at a depth d without a previous node in depth $d-1$ that is a father of the node.

The NPE operators are specific to generating a new spanning tree by the pruning and regrafting of a subtree. Therefore they produce only trees. When NPE is applied to complete graphs, the regraft node a is always a feasible one. Moreover, when dealing with non-complete graphs or problems with restrictions, a proper validation must be done on the method for determining the nodes p, r and a. The validation procedure in case of non-complete graphs must verify that node a is adjacent to node p in SPRN or to node r in TBRN. When the problem has constraints, the validation procedure is responsible to insure that nodes p, r and a respect it. Thus, the validation procedure insures that SPRN and TBRN will be applied only when the nodes result in feasible solutions. In this sense, NPE has the feasibility property, i.e. it encodes and generates only feasible solutions.

4.4. Analysis of NPE time complexity

An upper bound for the time complexity of NPE procedures can be derived as follows for complete graphs. All steps of SPRN and TBRN are

performed traversing array N a constant number of times. Thus, SPRN and TBRN (see Section 3) are $O(|N|)$, where $|N|$ is the cardinality of N.

The determination of nodes p, r, and a (see procedures in Section 3.3) is also $O(|N|)$. The first step can be performed in $O(1)$, since it requires only the selection of a random position. The next step is executed in $O(|N|)$. The construction of R_p traverses array N once. For TBRN, the next step determines node r and requires time complexity $O(1)$, since it selects a random position from R_p.

The last step determines node a by removing the positions of R_p from N so that only feasible values for node a are obtained. This task can be performed in $O(|N|)$, since R_p is composed of consecutive and ordered positions from N and those positions can be removed from N by traversing N once and copying the valid positions into an auxiliary array. The algorithm randomly selects a position from this auxiliary array and calls it i_a, the position of node a. Therefore, the time complexity of NPE operators is $O(|N|)$.

This analysis is of the standard operation in complete graphs and considers no problem constraints. For example, in the degree constrained minimum spanning tree problem (dc-MSTP) [30], nodes a and r can not violate a degree upper bound [31]. Using NPE in this problem, we fixed a number k_n of trials in the process of searching for a node that satisfies the degree constraint. This limit is established for each node (p, r and a) search. Note that k_n is a relatively small number that maintains the time complexity equal to $O(|N|)$.

In the worst case SPRN has $O({k_n}^2)O(|N|)$ time complexity, where $O({k_n}^2)$ results from verifying at most k_n values for a possible node p, as well as at most k_n value verifications for a node a for each p. Therefore ${k_n}^2$ pairs of nodes are evaluated in the worst case and the resulting time complexity is $O(|N|)$.

A stress test was carried out to empirically demonstrate that k_n can be a small value. First, the test generates a solution F with the maximum possible number of nodes with degree equal to the constraint of the dc-MSTP. Then, it generates a new solution using SPRN based on F and counts the number of trials until SPRN has reached a valid set of nodes p and a for a specific k_n. Parameter k_n varies from 1 to 5. Figure 10 shows the results using an average of 50 trials for each k_n value. The stress test also evaluated graphs with different numbers of nodes obtaining the same behavior for all graphs. Figure 10 presents the average number of SPRN trials and the corresponding standard deviation (std) from a total of 50 trials. We can observe that an

upper bound for verifications (k_n) equal to or larger than four is needed for applying SPRN without failure. If $k_n < 4$, it is possible that SPRN or TBRN cannot be applied because the determination process results in an infeasible set of nodes.

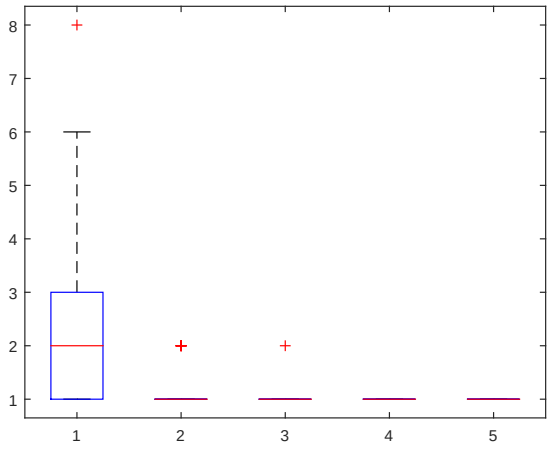


Figure 10: Average number of SPRN trials when the upper bound k is set to a relatively small value for the degree-constrained minimum spanning tree problem with degree constraint 3.

When dealing with non-complete graphs the strategy for choosing nodes p, r and a is similar to that used in the dc-MSTP problem. In this case, we need to check in the adjacency matrix of the graph if node a is adjacent to node p for SPRN (or to node r for TBRN). Thus, with a small value of k_n it is possible to keep NPE's time complexity for non-complete graphs the same as for the complete graph case.

5. Conclusions

This paper presented an encoding and its operators for EAs applied to problems involving spanning trees, called Node-depth Phylogenetic-based Encoding (NPE). The operators of NPE are based on the phylogenetic tree rearrangement operators TBR and SPR, called SPRN and TBRN that simplifies NDDR operators for network design problems represented by spanning trees instead of spanning forests.

The analyses and simulation results presented in this paper demonstrate that NPE has desirable properties for EA encodings. NPE has high locality, since the change of a value in the NPE array corresponds to the change of a few edges in the phenotype. Considering SPRN, TBRN, and the decode process described, all encodings of NPE represent feasible solutions. The bias of NPE and its operators was evaluated by the measure of some tree properties in an evolutionary process. These measures showed that NPE, TBRN, and SPRN do not present bias to any special type of tree topology. Also, we demonstrated that NPE operators have average time complexity of $O(n)$, where n is the number of nodes in the tree. Since NPE operators (called SPRN and TBRN) do not depend on graph weights, NPE is adequate for working with a large variety of graphs, as Random Graphs, Misleading Graphs, Euclidean, or even for graphs with weights depending on the spanning tree topology. Based on the NPE properties demonstrated in this paper we can say that, using NPE, a larger variety of relevant Network Design Problems can potentially be solved by EAs.

As future work we propose case studies applying NPE in evolutionary algorithms to network design problems. In addition, we would like to study the development of biased search operators in NPE for specific problems, such as a weight heuristic. Also, the development of EAs with NPE for other network design problems.

References

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

[2] C. C. Palmer, A. Kershenbaum, Representing trees in genetic algorithms, in: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, 1994, pp. 379–384 vol.1. doi:10.1109/ICEC.1994.349921.

[3] G. Raidl, An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem, in: Proceedings of the 2000 Congress on Evolutionary Computation CEC00, IEEE Press, La Jolla Marriott Hotel La Jolla, California, USA, 2000, pp. 104–111.
URL citeseer.ist.psu.edu/raidl00efficient.html

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

[5] C. Tzschoppe, F. Rothlauf, H.-J. Pesch, The edge-set encoding revisited: On the bias of a direct representation for trees, in: K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Eds.), Genetic and Evolutionary Computation – GECCO-2004, Part II, Vol. 3103 of Lecture Notes in Computer Science, Springer-Verlag, Seattle, WA, USA, 2004, pp. 258–270. doi:doi:10.1007/b98645.
URL http://link.springer.de/link/service/series/0558/bibs/3103/31030258.htm

[6] T. W. de Lima, F. Rothlauf, A. C. Delbem, The node-depth encoding: Analysis and application to the bounded-diameter minimum spanning tree problem, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08, ACM, New York, NY, USA, 2008, pp. 969–976. doi:10.1145/1389095.1389279.
URL http://doi.acm.org/10.1145/1389095.1389279

[7] T. W. de Lima, A. C. B. Delbem, F. Rothlauf, Analysis of properties of recombination operators proposed for the node-depth encoding, in: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 137–138. doi:10.1145/2001858.2001936.
URL http://doi.acm.org/10.1145/2001858.2001936

[8] H. Prüfer, Neuer beweis eines satzes uber permutationen, Arch. Math. Phys.

[9] G. Zhou, M. Gen, A note on genetic algorithms for degree-constrained spanning tree problems, Networks (USA) 30 (2) (1997) 91–95.

[10] M. Gen, G. Zhou, An approach to the degree-constrained minimum spanning tree problem using genetic algorithm, Tech. rep., Ashikaga Intitute of Technology (1995).

[11] F. Rothlauf, D. E. Goldberg, A. Heinzl, Network random keys: A tree representation scheme for genetic and evolutionary algorithms, Evolutionary Computation 10 (1) (2002) 75–97. arXiv:http://www.mitpressjournals.org/doi/pdf/10.1162/106365602317301781, doi:10.1162/106365602317301781.
URL http://www.mitpressjournals.org/doi/abs/10.1162/106365602317301781

[12] C. C. Palmer, An approach to a problem in network design using genetic algorithms, Ph.D. thesis, Brooklyn, NY, USA (1994).

[13] L. Davis, D. Orvosh, A. Cox, Y. Qiu, A genetic algorithm for survivable network design, in: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, pp. 408–415.

[14] B. A. Julstrom, The blob code is competitive with edge-sets in genetic algorithms for the minimum routing cost spanning tree problem, in: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM Press, New York, NY, USA, 2005, pp. 585–590. doi:http://doi.acm.org/10.1145/1068009.1068108.

[15] C. C. Palmer, A. Kershenbaum, An approach to a problem in network design using genetic algorithms, Networks 26 (1995) 151–163.

[16] T. Paulden, D. Smith, From the dandelion code to the rainbow code: a class of bijective spanning tree representations with linear complexity and bounded locality., IEEE Transactions Evolutionary Computation 10 (2) (2006) 108–123.

[17] J. Gottlieb, B. A. Julstrom, G. R. Raidl, F. Rothlauf, Prüfer numbers: A poor representation of spanning trees for evolutionary search, in: L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann, San Francisco, California, USA, 2001, pp. 343–350.
URL citeseer.ist.psu.edu/article/gottlieb01prfer.html

[18] A. C. B. Delbem, T. W. Lima, G. P. Telles, Efficient forest data structure for evolutionary algorithms applied to network design, IEEE Transactions on Evolutionary Computation 99 (2012) 1–28.

[19] F. N. Abuali, R. L. Wainwright, D. A. Schoenefeld, Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem, in: L. J. Eshelman (Ed.), Proc. of the Sixth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, pp. 470–477.
URL citeseer.ist.psu.edu/abuali95determinant.html

[20] P. M. S. Carvalho, L. A. F. M. Ferreira, L. M. F. Barruncho, On spanning-tree recombination in evolutionary large-scale network problems - application to electrical distribution planning., IEEE Trans. Evolutionary Computation 5 (6) (2001) 623–630.

[21] S.-M. Soak, D. Corne, A. Byung-Ha, A new encoding for the degree constrained minimum spanning tree problem., in: KES, 2004, pp. 952–958.

[22] S.-M. Soak, D. Corne, B.-H. Ahn, A new evolutionary algorithm for spanning-tree based communication network design, IEICE Trans Commun E88-B (10) (2005) 4090–4093. arXiv:http://ietcom.oxfordjournals.org/cgi/reprint/E88-B/10/4090.pdf, doi:10.1093/ietcom/e88-b.10.4090.
URL http://ietcom.oxfordjournals.org/cgi/content/abstract/E88-B/10/4090

[23] G. Zhou, M. Gen, A genetic algorithm approach on tree-like telecommunication network design problem, J. of the Operational Research Society 54 (3) (2003) 248–254.

[24] B. A. Julstrom, Encoding bounded-diameter spanning trees with permutations and with random keys, in: K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Eds.), Genetic and Evolutionary Computation – GECCO-2004, Part I, Vol. 3102 of Lecture Notes in Computer Science, Springer-Verlag, Seattle, WA, USA, 2004, pp. 1272–1281. doi:doi:10.1007/b98643.

URL http://link.springer.de/link/service/series/0558/bibs/3102/31021272.htm

[25] G. Raidl, C. Drexel, A predecessor coding in an evolutionary algorithm for the capacitated minimum spanning tree problem, in: C. Armstrong (Ed.), Proceeding of 2000 Genetic and Evolutionary Computation Conference, 2000, pp. 309–316.

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

[27] J. Felsenstein, Inferring phylogenies, Sinauer, Sunderland, Massachusetts, 2004.

[28] C. E. Andrade, M. G. Resende, W. Zhang, R. K. Sinha, K. C. Reichmann, R. D. Doverspike, F. K. Miyazawa, A biased random-key genetic algorithm for wireless backhaul network design, Applied Soft Computing 33 (2015) 150 – 169. doi:http://dx.doi.org/10.1016/j.asoc.2015.04.016.
URL http://www.sciencedirect.com/science/article/pii/S1568494615002410

[29] R. W. Hamming, Coding and Information Theory, Prentice-Hall, 1980.

[30] M. R. Garey, D. S. Johnson, Computers and intractibility (1979).

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