A Preliminary Comparison of Tree Encoding Schemes for Evolutionary Algorithms

Eduardo G. Carrano, Carlos M. Fonseca, Ricardo H. C. Takahashi, Luciano C. A. Pimenta and Oriane M. Neto

***Abstract*— This paper presents a comparative study of six encodings which have been used to represent trees in evolutionary algorithms. The study has been divided into two steps: 1) The encoding methods have been evaluated taking into account the time necessary to perform operations such as decoding, crossover and mutation, the feasibility of solutions after those operations, and the corresponding heritability and locality; 2) The encoding methods have been employed in a genetic algorithm to solve three different instances (with 10, 25 and 50 nodes) of the optimal communication spanning tree problem. Finally, the results obtained with each of the encodings are statistically compared using Kruskal-Wallis non-parametric tests and multiple comparisons. The results of this study provide insight into the properties of current encoding schemes for network design problems.**

I. INTRODUCTION

The design of network topologies is an important class of combinatorial optimization problems. Within this class, tree design problems have received special attention, since they are the basis for the modeling of a large range of practical systems, such as telecommunication, gas, electric, computer and transportation networks [1]. In those cases, the use of optimization techniques is very important, because even small improvements in the implemented system often lead to considerable financial savings.

Let $G(V, E)$ be an undirected, weighted, connected graph with $n = |V|$ vertices and $m = |E|$ edges, where V and E are the sets of vertices and edges respectively. By definition, a spanning tree T in G is any subgraph of G which connects all V vertices with only $n-1$ edges (T is necessarily acyclic). In [2], it is proved that a fully connected graph G with n vertices contains a set of n^{n-2} feasible spanning trees ($\mathcal{T}$). Assuming that the cost of a spanning tree is calculated as the sum of its edge weights, it is possible to define the minimum spanning tree (T^*) as the element of $\mathcal{T}$ which has minimum total edge weight. The problem of finding the minimum spanning tree (also called *MST*) can be solved in polynomial time using well-known algorithms, such as those of Prim [3] or Kruskal [4]. However, many variants of the *MST* problem are computationally difficult, such as the optimal communication spanning tree (*OCST*) and the quadratic minimum spanning tree (*QMST*) problems, which are both *NP-hard*. Till now, these variants cannot be solved in polynomial time using exact methods [5].

Evolutionary algorithms (EAs) arise as reasonable methods for solving these variants of the *MST* and other hard network problems. However, it is important to note that, in evolutionary algorithms in general, and in evolutionary network optimization in particular, the choice of representation scheme (encoding) is a critical aspect of the design of the algorithm [6], as the use of inadequate representation schemes often leads to a noticeable reduction in algorithm performance. This is usually due to the resulting search spaces exhibiting undesirable characteristics, such as many local optima and multiple, disconnected, feasible regions, which make the optimization process considerably harder.

It is possible to find other works in the literature which present comparative studies of tree encoding methods [5], [7], [8]. Although these studies are useful to some extent, they are usually aimed at justifying the introduction of yet another encoding scheme, and tend to include only a small number of alternative methods in the comparison. Another limitation of those studies is that the comparisons are based solely on mean values, which can lead to conclusions that are not statistically well supported.

This paper presents a comparison of six classical tree encodings for evolutionary algorithms. The performance of a genetic algorithm (GA) based on each of these encodings, when applied to three *OCST* problem instances, and encoding characteristics generally considered relevant in the literature, are used as merit criteria. The data obtained is analyzed using Kruskal-Wallis non-parametric tests and multiple comparisons [9].

This paper is structured as follows:

- Section II shows the general statement of tree optimization problems. The *OCST* problem is also presented in this section.
- Section III describes a number of characteristics which are usually considered desirable in tree encoding schemes.
- Section IV presents the tree encoding schemes considered in this work.
- Section V presents a practical analysis of some decoding, crossover and mutation operators.

This work was supported by CNPq, CAPES and FAPEMIG - Brazil.

E. G. Carrano, L. C. A. Pimenta and O. M. Neto are with the Department of Electrical Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, Belo Horizonte, MG, 31270-010, Brazil (e-mail: carrano@cpdee.ufmg.br, lucpim@cpdee.ufmg.br, oriane@cpdee.ufmg.br).

R. H. C. Takahashi is with the Department of Mathematics, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, Belo Horizonte, MG, 31270-010, Brazil (e-mail: taka@mat.ufmg.br).

C. M. Fonseca is with the Centre for Intelligent Systems, Universidade do Algarve, Campus de Gambelas, 8005-139 FARO, Portugal (e-mail: cmfonsec@ualg.pt).

- Section VI discusses the results obtained by the resulting algorithms on the *OCST* problem.

II. PROBLEM STATEMENT

As presented in [7], the *MST* problem can be formulated as an integer programming problem, as described below.

Let $G(V, E)$ be a connected, undirected, weighted graph and $w_{i,j} \in W$ be the weight of connection (i, j). Let $T \subseteq E$ and S be the set of vertices induced by T (i.e., S is the set of vertices connected by the edges in T). The *MST* problem can be formulated as finding T which minimizes the cost:

$$\sum_{i,j \in V} w_{i,j} \cdot t_{i,j} \tag{1}$$

subject to:

$$\begin{array}{lcll} \sum\limits_{i,j \in V} t_{i,j} & = & |V| - 1 & \\ \sum\limits_{i,j \in V} t_{i,j} & \leq & |S| - 1 & \text{for any set } S \\ t_{i,j} \in \{0, 1\} & , & \forall i, j \in V & \end{array} \tag{2}$$

where $(i, j) \in T$ if and only if $t_{i,j} = 1$.

The formulation shown in (1) and (2) may be extended to any variant of the *MST* problem by considering a different objective function and any additional constraints required. In particular, in the case of the *OCST* problem, there are no additional constraints, and the objective function may be written as:

$$\min \sum_{i,j \in V} R_{i,j} \cdot C^X_{i,j} \tag{3}$$

where:
$C^X_{i,j}$ is the sum of weights of edges in path $i - j$.
$R_{i,j}$ is the communication requirement between i and j.

The *OCST* problem can be understood as follows: Let T be a communication system with n nodes and $R_{i,j}$ be the call demand between each pair of nodes i and j. The cost incurred by each pair of nodes (i, j) is calculated by multiplying the total cost of path (i, j), (including all intermediate branches) by the call demand $R_{i,j}$.

This problem has been proved to be *NP-hard*, as shown in [10], and has broad applicability in practice. In the present work, it is used as a benchmark problem in section VI, to evaluate the encoding schemes discussed in section IV.

III. DESIRABLE CHARACTERISTICS OF TREE ENCODING SCHEMES

References [7], [11] state six characteristics which are considered to be desirable in tree encoding schemes:

1) **Coverage**: The encoding should be capable of representing all possible trees;
2) **Unbiasedness**: The encoding should represent all possible trees with the same number of codes ($1 \rightarrow n$: each tree is represented by n different codes). When each tree is represented by only one code ($1 \rightarrow 1$) it is said that the encoding possesses **uniqueness**;
3) **Feasibility**: All possible codes should represent feasible trees, and the result of crossover or mutation operators should always result in feasible trees;
4) **Time**: It should be computationally cheap to go back and forth from code to tree and perform recombination and mutation operations;
5) **Heritability**: The crossover operation should result in trees which combine only the edges of parent trees;
6) **Locality**: A single mutation in a code should represent a small change in the decoded tree.

In principle, all tree encodings used in EAs should have those characteristics. However, most of the available representations fail in at least one of them. The next section presents a short description of six tree encoding methods, considering the first three aspects described above. The analysis of time, heritability and locality of those methods is presented in section V.

IV. TREE ENCODING SCHEMES

Six classical tree encoding schemes are considered in this work:

- ***Characteristic vector***: In *characteristic vector* the trees are represented by binary vectors of dimension m. Each component of the vector represents one possible edge and can assume values 0 (the edge is not present in the tree) or 1 (the edge is present in the tree). This encoding can be derived directly from the problem formulation shown in (1) and (2). This encoding is unbiased ($1 \rightarrow 1$ mapping) and covers all possible solutions. However, many possible codes represent infeasible structures.
- ***Prüfer numbers***: In *Prüfer numbers* [12] the trees are represented by integer vectors of dimension $n-2$, each component of which can assume values from 1 to n. This encoding scheme is based on Cayley's Theorem [2], which states that in a complete graph with n vertices, it is possible to find n^{n-2} trees. The encoding is unbiased ($1 \rightarrow 1$ mapping), covers all possible solutions and all codes represent feasible structures. However, the encoding only works with complete graphs.
- ***Network random keys***: In *network random keys* [1] the trees are represented by real vectors of dimension m. Each component of the vector represents the weight of an edge, usually from 0 to 1. The tree is obtained using an *MST* algorithm, such as those of Prim or Kruskal. The encoding covers all candidate solutions with full feasibility. However, it has high redundancy, since the number of codes which represent the same tree is very high (in theory, it is infinite).
- ***Edge sets***: In *edge sets* [8] the trees are represented by integer vectors of dimension $2n - 2$. Each vector component can assume values in $1, \ldots, n$. The encoding is unbiased, redundant and covers all possible solutions. Although the infeasibility rate is smaller than in characteristic vector, it is still a restrictive aspect of this encoding. Additional repair mechanisms should be employed in incomplete graph instances.

- ***Node biased encoding***: In the *node biased encoding* [11] the trees are represented by real vectors of dimension n. Each vector component represents the weight of a node. Those node weights are used to modify the weights of the original edges. As with network random keys, the trees are obtained by an *MST* algorithm. The encoding is biased (edges with lower weight have a higher chance of being selected) and it may not cover the whole set of alternative solutions in some situations [11]. The decoding always results in feasible structures, despite exhibiting high redundancy.
- ***Link and node biased encoding***: The *link and node biased encoding* [11] is an adaptation of *node biased encoding*, to correct the coverage limitation. In this encoding, the trees are represented by real vectors of dimension $m + n$. Each component of the vector represents the weight of nodes and edges, all of which are used to modify the original edge weights. The other properties described in node biased encoding are valid for this method, too.

V. ANALYSIS OF EVOLUTIONARY OPERATORS

The decoding, crossover and mutation operators were tested on three fully-connected graphs of different sizes (10, 25 and 50 nodes). The nodes were randomly generated according to a uniform distribution on the unit square and the edge weights were set to the Euclidean distance between the corresponding nodes. Four criteria were considered:

1) Time (decoding, crossover and mutation): time necessary to perform the operation;
2) Infeasibility (crossover and mutation): rate of structures which become infeasible after a crossover or mutation operation;
3) Heritability (crossover): number of edges in offspring network which do not belong to the parents;
4) Locality (mutation): number of edges that must be changed in the parent network to obtain the offspring network.

The results obtained for each criterion were analyzed using Kruskal-Wallis non-parametric tests and multiple comparisons [9].

It is important to emphasize that all operators presented in this work (decoding, crossover and mutation) have been implemented with the lowest computational complexity which could be found in the literature. They have been tested on a Pentium IV (Prescott) at 3.2GHz with 1024MB of RAM, using the Matlab 7 environment. Although the times are not comparable with other approaches, since they are strictly dependent of the hardware and software used, the time ratio between the methods can provide useful information about the computational cost of the methods.

A. Decoding

The analysis of the decoding operators for the six encodings was performed as follows:

1) 250 trees were randomly generated using random vectors of Prüfer numbers (for each graph instance), re-encoded according to each encoding scheme considered, and subsequently decoded. The time necessary to decode each tree using each scheme was recorded.
2) To detect differences between the times measured for each encoding scheme, a Kruskal-Wallis hypothesis test was applied to the whole data set, and the corresponding multiple comparison procedure was used to determine which pairs of encodings differed significantly from each other.

The decoding operators were labeled as follows:

Label	Decoding
A	Characteristic vector
B	Prüfer numbers
C	Network random keys
D	Edge sets
E	Node biased
F	Link and node biased

The average decoding time achieved for each encoding scheme, and the corresponding standard deviation are shown in Table I, which also summarizes the results by listing the methods in descending order of performance. In this list, underlining and overlining are used to indicate which methods did not exhibit statistically significant differences from one another.

TABLE I
DECODING - TIME (ms)

	10 nodes		25 nodes		50 nodes	
Lab.	avg	sd	avg	sd	avg	sd
A	0.031	0.001	0.036	0.000	0.053	0.002
B	0.253	0.003	0.645	0.012	1.323	0.011
C	0.704	0.085	2.154	0.353	4.887	0.838
D	0.057	0.002	0.098	0.001	0.169	0.002
E	1.466	0.139	8.118	1.912	29.745	2.388
F	1.607	0.123	8.681	0.666	32.538	2.012

Order (best to worst):
10, 25 and 50 nodes:
$\underline{A}$ $\underline{D}$ $\underline{B}$ $\underline{C}$ $\underline{E}$ $\underline{F}$

In this case, all methods differ significantly in decoding time and, thus, the under/overlining covers only one letter at a time. If two or more methods were underlined (or overlined) together, it would mean that the difference observed between those methods was not statistically significant.

Decoding operators A and D were the fastest methods for all instances. On the other hand, the time necessary to perform decoding in E and F was observed to increase considerably with instance size.

B. Crossover

Two crossover operators were used for each of the six encoding methods:

1) Single point crossover;
2) Uniform crossover.

Additionally, one crossover operator which performs the operation in solution space (space of the networks), was formulated and tested. It operates in the following way:

- Let P_1 and P_2 denote the two parent trees;
- Create a graph $S = P_1 \bigcup P_2$;
- Generate two sets of random weights, W_1^S and W_2^S, for the edges of S;
- Find two offspring trees, O_1 and O_2, by applying the Kruskal algorithm to S using each set of weights, W_1^S and W_2^S.

It is straightforward to note that, by construction, this operator always generates feasible networks, and provides full heritability (only the edges of parent networks are used to build the offspring networks). The *Kruskal* algorithm was implemented using a Union-Find node labeled data structure and path compression [8]. In the remaining of this work, this operator will be referred to as *Kruskal crossover*.

Various crossover operators were compared using the same methodology described in Decoding section, and were labeled as follows:

Label	Decoding	Crossover
A	Characteristic vector	single point
B	Characteristic vector	uniform
C	Prüfer numbers	single point
D	Prüfer numbers	uniform
E	Network random keys	single point
F	Network random keys	uniform
G	Edge sets	single point
H	Edge sets	uniform
I	Node biased	single point
J	Node biased	uniform
K	Link and node biased	single point
L	Link and node biased	uniform
M	-	Kruskal

Tables II and III present the results obtained for infeasibility and heritability, respectively.

TABLE II
CROSSOVER - INFEASIBILITY

	10 nodes	25 nodes	50 nodes
Lab.	avg	avg	avg
A	0.7820	0.9320	0.9720
B	0.9480	1.0000	1.0000
G	0.8240	0.9340	0.9920
H	0.9120	1.0000	1.0000

Order (best to worst):
10 and 25 nodes:
$A \quad G \quad H \quad B$
50 nodes:
$A \quad G \quad H \quad B$

Operators C, D, E, F, I, J, K, L and M were not considered in the infeasibility test since, by construction, they present full feasibility. It is noticeable that the infeasibility of operators A, B, G and H is very large, even for small instances. In practice, it is necessary to use some kind of repair mechanism together with those operators. In comparison with single-point crossover, uniform crossover exhibited higher infeasibility regardless of the underlying encoding.

TABLE III
CROSSOVER - HERITABILITY

	10 nodes		25 nodes		50 nodes	
Lab.	avg	sd	avg	sd	avg	sd
C	1.688	1.281	6.566	3.248	15.326	7.021
D	2.784	1.432	12.932	2.462	31.894	3.490
E	0.838	0.902	2.276	1.630	4.922	2.847
F	1.382	1.017	4.350	1.946	9.216	2.970
G	0.226	0.419	0.482	0.500	0.448	0.498
H	2.514	1.429	9.854	2.381	21.838	3.197
I	1.122	1.266	3.748	3.134	7.658	6.135
J	1.344	1.293	5.322	3.284	12.520	6.208
K	1.758	1.340	6.756	3.545	15.796	8.721
L	2.162	1.296	8.886	2.466	21.472	3.670

Order (best to worst):
10 nodes:
$G \quad E \quad I \quad J \quad F \quad C \quad K \quad L \quad H \quad D$
25 nodes:
$G \quad E \quad I \quad F \quad J \quad C \quad K \quad L \quad H \quad D$
50 nodes:
$G \quad E \quad I \quad F \quad J \quad C \quad K \quad L \quad H \quad D$

Operators A, B and M were excluded from the heritability test, since, by construction, they possess full heritability. Among the other methods, G exhibited the highest heritability (i.e., lowest number of non-parent edges in the offspring). As reported in the literature ([7], [11]), the *Prüfer numbers* presented low heritability. It is still noticeable that the uniform crossover generally reduces the heritability for all encodings.

Although the time required to perform crossover was initially considered an important aspect, the experiments showed that the time differences between operators A to L were very minor from a practical point of view. Kruskal crossover (operator M), on the other hand, was up to 250 times slower than the other operators. This may simply be due to the fact that the MATLAB implementation of the Kruskal algorithm required the use of an interpreted for loop, whereas the remaining operator implementations could be fully vectorized.

C. Mutation

Two mutation operators were employed for each of the six encoding methods:

1) Point mutation;
2) Swap mutation.

As in the crossover study, one mutation operator based on the Kruskal algorithm was formulated. It can be described as follows:

- Let P denote the parent tree;
- Select an edge p_1 from $E - P$, at random;
- Create a new graph S by adding edge p_1 to P;
- Generate a random set of weights in $]0, 1]$ for the edges of S and set the weight corresponding to p_1 to 0, to obtain W^S;
- Find the offspring network O by applying the *Kruskal* algorithm to S using W^S as the set of edge weights.

As in *Kruskal crossover*, it is easy to see that this operator possesses full locality (all possible offspring trees may differ from the parent tree by a single edge replacement), and never generate non-tree graphs.

The mutation operators considered were labeled as follows:

Label	Decoding	Mutation
A	Characteristic vector	swap
B	Prüfer numbers	point
C	Prüfer numbers	swap
D	Network random keys	point
E	Network random keys	swap
F	Edge sets	point
G	Edge sets	swap
H	Node biased	point
I	Node biased	swap
J	Link and node biased	point
K	Link and node biased	swap
L	-	kruskal

Tables IV and V show the results obtained for infeasibility and locality.

TABLE IV
MUTATION - INFEASIBILITY

	10 nodes	25 nodes	50 nodes
Lab.	avg	avg	avg
A	0.6200	0.8280	0.8520
F	0.4480	0.4600	0.4640
G	0.4400	0.4600	0.4760

Order (best to worst):
10, 25 and 50 nodes:
$F \quad G \quad A$

TABLE V
MUTATION - LOCALITY

	10 nodes		25 nodes		50 nodes	
Lab.	avg	sd	avg	sd	avg	sd
B	1.456	1.486	2.792	3.253	6.504	6.904
C	2.060	1.169	3.452	2.287	5.928	5.378
D	0.712	0.454	0.860	0.348	0.936	0.245
E	0.780	0.415	0.900	0.301	0.972	0.165
F	0.164	0.371	0.060	0.238	0.028	0.165
G	0.788	0.410	0.924	0.266	0.940	0.238
H	1.044	0.937	1.596	2.062	2.008	2.525
I	1.716	1.559	2.420	2.655	3.100	4.657
J	0.808	0.548	0.912	0.499	1.008	0.634
K	0.812	0.683	0.840	0.418	0.932	0.559

Order (best to worst):
10 nodes:
$F \quad D \quad E \quad G \quad J \quad K \quad H \quad B \quad I \quad C$
25 nodes:
$F \quad K \quad D \quad E \quad J \quad G \quad H \quad I \quad B \quad C$
50 nodes:
$F \quad K \quad D \quad G \quad E \quad J \quad H \quad I \quad C \quad B$

Once again, the methods which guarantee offspring feasibility (B, C, D, E, H, I, J, K and L) were excluded from the feasibility test. The edge sets representation exhibited significantly lower infeasibility than the binary representation, regardless of the type of mutation (point or swap).

By construction, methods A and L present full locality and do not need to be considered in the locality test. As for heritability, the *edge sets* representation exhibited high locality. The *Prüfer numbers* representation showed poor locality, as discussed in [7], [11].

Regarding computation times, the same considerations made for crossover apply.

VI. ALGORITHM RESULTS AND ANALYSIS

The operators described in the previous sections were used to build genetic algorithms for the optimization of the *OCST* problem. Single point crossover was used in all versions of the GA, since it seemed to possess better properties than uniform crossover. The resulting algorithms were evaluated based on two criteria:

1) Best function value: Value of the objective function for the best solution found;
2) Computation time.

The following parameters were used in the simulations:

- Number of runs: 30 runs per method;
- Population size: 50 individuals;
- Crossover probability: 0.80;
- Mutation probability: 0.01;
- Linear ranking and roulette-wheel selection;
- Generational replacement with elitism;
- Stop criterion: 100 generations without improvement.

The algorithms were labeled as follows:

Label	Decoding	Crossover	Mutation
A	Characteristic vector	single point	swap
B	Prüfer numbers	single point	point
C	Prüfer numbers	single point	swap
D	Network random keys	single point	point
E	Network random keys	single point	swap
F	Edge sets	single point	point
G	Edge sets	single point	swap
H	Node biased	single point	point
I	Node biased	single point	swap
J	Link and node biased	single point	point
K	Link and node biased	single point	swap
L	-	kruskal	kruskal

A. OCST Problem

Tables VI and VII show the results (convergence time and best function value) obtained on three instances of the *OCST* problem. The instances were based on the fully-connected graphs used previously, and the communication requirements between different nodes were generated uniformly at random.

Algorithms H, I, F, G and L have achieved remarkable convergence results (Table VII). Those algorithms are based on *node biased* and *edge sets* encodings, and on the *Kruskal* operators, respectively.

On the other hand, algorithms D, E, B and C, which are based on *Network random keys* and *Prüfer numbers* encodings, have achieved the worst convergence results. In the case of the algorithms based on *Prüfer numbers*, this performance can be explained by the poor heritability and locality associated with this representation. The low performance of *Network random keys* may have been caused by the high redundancy and neutrality of the encoding.

From Table VI, it is possible to note that the algorithms based on *Prüfer numbers*, which had poor convergence, was

TABLE VI

OCST - BEST FUNCTION VALUE

	10 nodes		25 nodes		50 nodes	
Lab.	avg	sd	avg	sd	avg	sd
A	3.65e5	7.25e3	2.58e6	1.09e4	1.19e7	3.52e5
B	3.67e5	1.24e4	2.90e6	2.01e5	1.32e7	7.53e5
C	3.74e5	1.62e4	2.85e6	2.00e5	1.30e7	8.96e5
D	3.75e5	1.59e4	2.93e6	2.81e5	1.42e7	1.36e6
E	3.74e5	1.57e4	2.80e6	1.59e5	1.42e7	1.64e6
F	3.64e5	0.00e0	2.57e6	9.89e3	1.16e7	2.28e5
G	3.64e5	0.00e0	2.60e6	2.13e4	1.25e7	7.30e5
H	3.64e5	0.00e0	2.56e6	6.24e3	1.11e7	2.53e4
I	3.64e5	1.65e3	2.58e6	1.02e4	1.12e7	3.97e4
J	3.64e5	1.44e3	2.64e6	3.37e4	1.17e7	2.98e5
K	3.64e5	9.32e2	2.62e6	3.16e4	1.17e7	3.07e5
L	3.64e5	0.00e0	2.57e6	5.60e3	1.20e7	4.99e5

Order (best to worst):

10 nodes:

$F \; G \; H \; L \; K \; I \; J \; A \; B \; E \; C \; D$

25 nodes:

$H \; L \; F \; I \; A \; G \; K \; J \; E \; C \; B \; D$

50 nodes:

$H \; I \; F \; K \; J \; A \; L \; G \; C \; B \; E \; D$

TABLE VII

OCST - COMPUTATION TIME (ms)

	10 nodes		25 nodes		50 nodes	
Lab.	avg	sd	avg	sd	avg	sd
A	1.956e4	2.628e3	1.325e5	2.235e4	7.317e5	2.262e5
B	8.984e3	7.426e2	3.009e4	7.329e3	1.917e5	4.363e4
C	8.578e3	1.655e3	2.246e4	4.737e3	1.272e5	4.821e4
D	1.492e4	2.255e3	7.437e4	1.828e4	3.376e5	1.046e5
E	1.648e4	3.692e3	7.306e4	2.400e4	3.194e5	1.132e5
F	1.516e4	1.225e3	6.471e4	9.189e3	4.280e5	9.074e4
G	1.638e4	2.266e3	7.306e4	2.147e4	3.412e5	1.241e5
H	1.648e4	8.892e2	7.474e4	1.331e4	5.401e5	1.531e5
I	1.682e4	4.310e2	5.902e4	1.546e4	3.409e5	9.116e4
J	1.773e4	1.085e3	1.103e5	2.824e4	7.909e5	2.169e5
K	1.913e4	1.785e3	1.321e5	4.066e4	8.369e5	2.968e5
L	2.001e4	1.745e3	9.654e4	1.227e4	6.972e5	1.792e5

Order (best to worst):

10 nodes:

$C \; B \; D \; F \; G \; E \; H \; I \; J \; K \; A \; L$

25 nodes:

$C \; B \; F \; E \; G \; D \; I \; H \; L \; J \; A \; K$

50 nodes:

$C \; B \; E \; D \; G \; F \; I \; H \; L \; A \; J \; K$

also the fastest algorithms for all instances. On the other hand, the algorithms with good convergence performance, such as F, G, H, I, J, K and L, were clearly slower than algorithms B and C. This reflects the stopping criterion adopted, which only allows the algorithm to proceed as long as improvement is being made. Overall, the algorithms based on *edge sets* and *node biased* encodings would seem to be the most suitable methods for the *OCST* problems considered.

The algorithms based on the *link and node biased* encoding also exhibited good performance, but they were usually outperformed by the simpler *node biased* encoding. Finally, the *binary* and the *network random keys* encodings showed poor performance, and do not seem appropriate for the *OCST* problem, despite exhibiting good heritability and locality.

VII. CONCLUDING REMARKS

This paper has presented a preliminary comparison of six classical tree encoding methods. The encodings have been compared in a two step analysis: 1) Study of encoding characteristics (feasibility, heritability, locality and computation time); and 2) Study of the performance of encodings on a *NP-hard* problem.

The results suggest the *node biased* and *edge sets* representations as the most suitable representations for the OCST problem. This is in line with the results of previous comparative studies [8], [11], although neither included both the *node biased* and the *edge sets* representations.

A. Future Work

Since many aspects of tree encodings have not been addressed in this study, the following extensions are planned for this work:

- Study of encoding schemes proposed more recently, such as *PrimPred-based encoding*, *Dandelion code* and *Edge-window encodings*;
- Evaluation of performance of encoding schemes in other *NP-hard* problems, such as *DC-MST* and *QMST*;
- Evaluation of performance of encoding schemes in other types of instances, such as non-Euclidean graphs;
- Constraint handling; and
- Incremental function evaluation.

REFERENCES

[1] F. Routhlauf, D. Goldberg, and A. Heinzl, "Network random key - A tree network representation scheme for genetic and evolutionary algorithms," *Evolutionary Computation*, vol. 10, pp. 75–97, 2002.

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

[3] R. Prim, "Shortest connection networks and some generalizations," *Bell. Syst. Tech. Jour.*, vol. 36, pp. 1389–1401, 1957.

[4] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling sallesman problem," *Proc. of Amer. Math. Soc.*, vol. 7, pp. 48–50, 1956.

[5] S. Soak, D. W. Corne, and B. Ahn, "The edge-window-decoder representation for tree-based problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 124–144, 2006.

[6] H. Chou and G. P. C. Chu, "Genetic algorithms for communication network design - An empirical study of the factors that influence performance," *IEEE Trans. Evol. Comput.*, pp. 236–249, 2001.

[7] L. Lin and M. Gen, "Node-based genetic algorithm for communication spanning tree problem," *IEICE Trans. Commun.*, vol. E89-B, pp. 1091–1098, 2006.

[8] G. R. Raidl and B. A. Julstrom, "Edge-sets: An effective evolutionary encoding of spanning trees," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 225–239, 2003.

[9] W. J. Connover, *Practical Nonparametric Statistics*, 3rd ed. Wiley, 1999.

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

[11] C. C. Palmer and A. Kershenbaum, "Representing trees in genetic algorithms," in *Proc. IEEE Conference on Evolutionary Computation*, Bristol, USA, 1994, pp. 379–384.

[12] H. Prüfer, "Neuer Beweis eines Satzes über Permutationen," *Arch. Math. Phys*, vol. 27, pp. 742–744, 1918.