Information Sciences 179 (2009) 3461–3472

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Evolutionary design of oriented-tree networks using Cayley-type encodings

Sancho Salcedo-Sanz [a,*], Maurizio Naldi [b], Ángel M. Pérez-Bellido [a], Antonio Portilla-Figueras [a], Emilio G. Ortiz-García [a]

[a] Departamento de Teoría de la Señal y Comunicaciones, Universidad de Alcalá de Henares, Campus Universitario, 28871 Alcalá de Henares, Madrid, Spain
[b] Department of Informatics, Systems and Production, Universitá di Roma II, "Tor Vergata", Rome, Italy

ARTICLE INFO

Article history:
Received 20 November 2007
Received in revised form 24 June 2009
Accepted 25 June 2009

Keywords:
Evolutionary algorithms
Tree encodings
Oriented-tree network design problem
Prüfer encoding
Dandelion code

ABSTRACT

This paper introduces the oriented-tree network design problem (OTNDP), a general problem of tree network design with several applications in different fields. We also present several adaptations needed by evolutionary algorithms with Cayley-type encodings to tackle the OTNDP. In particular, we present these adaptations in two Cayley-encodings known as Prüfer and Dandelion codes. We include changes in Cayley-encodings to consider rooted trees. We also show how to use a fixed-length encoding for Cayley codes in evolutionary algorithms, and how to guarantee that the optimal solution is included in the search space. Finally, we present several adaptations of the evolutionary algorithm's operators to deal with Cayley-encodings for the OTNDP. In the experimental part of the paper, we compare the performance of an evolutionary algorithm (implementing the two Cayley-encodings considered) in several OTNDP instances: first, we test the proposed techniques in randomly generated instances, and second, we tackle a real application of the OTNDP: the optimal design of an interactive voice response system (IVR) in a call center.

1. Introduction

Many combinatorial optimization problems related to networks consist of obtaining optimal trees in terms of an objective function and a set of constraints [7,9–15]. In many cases, the tree must be calculated to begin from an initial graph $G(V,E)$. In this case we have a *spanning tree* design problem [24]. Problems involving spanning trees arise in multiple fields, such as telecommunications [23,15,14], economics [8], and transportation system design [11].

Combinatorial problems in which the optimal form of a tree must be obtained have attracted the attention of researchers in the last few years [2,19]. In this case, the constraints do not concern the links that belong to the tree, as in the case of spanning trees, but rather the maximum number of nodes or leaves in the tree.

In this paper tackle a problem of tree network design, the oriented-tree network design problem (OTNDP). This problem looks for the oriented-tree network with the minimum average number of bifurcations. The OTNDP has direct applications to the design of communication networks [20], hydric networks [12], structure of organizations [3], systems design, etc. A number of algorithms and heuristics can be found in the literature for tree design and tree encoding [4,17–21]. Specifically, intensive research has been carried out in the field of evolutionary computation applied to the resolution of problems of tree design [16,21]. In the majority of cases, the evolutionary algorithms (EAs) have been applied to problems involving spanning trees [16,22]. However, several problems require the extension of these algorithms and encodings to the case of oriented-tree design problems, which, to our knowledge, has not been tackled using evolutionary algorithms with Cayley-type encodings.

* Corresponding author. Address: Departamento de Teoría de la Señal y Comunicaciones, Universidad de Alcalá de Henares, Campus Universitario, 28871 Alcalá de Henares, Madrid, Spain. Tel.: +34 91 885 6731; fax: +34 91 624 8749.
E-mail address: sancho.salcedo@uah.es (S. Salcedo-Sanz).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.06.034

In this paper we present several adaptations of EAs using Cayley-encodings for the OTNDP. These adaptations introduce changes in Cayley-encodings to consider rooted trees. We also discuss the possibility of using a fixed-length encoding, guaranteeing that the optimal solution is included in the search space. Finally, we present several adaptations of the EA operators to deal with Cayley-encodings for the OTNDP. The paper considers two different tree encodings strategies based on Cayley-encodings: the Prüfer encoding and the Dandelion code [16]. All the adaptations proposed in this paper are tested in these two encodings. We will show that the Dandelion code outperforms the Prüfer encoding in the OTNDP as expected, producing results quite close to a theoretical lower bound for the problem, also obtained in this work.

The rest of the paper is structured as follows: the next section summarizes the main properties of Cayley encoding for trees, specifying the characteristics of the Prüfer encoding and the Dandelion code. In Section 3 we present the OTNDP definition, and introduce the calculation of a lower bound for the OTNDP based on the optimization of N-ary trees. In Section 4 we describe the specific adaptations that must be included in EAs using Cayley encodings to deal with the OTNDP. Section 5 includes the experimental part of the paper, where experiments in randomly generated OTNDP instances, and a real application in the design of a call center system are presented. Section 6 closes the paper with some final remarks.

2. Background: Cayley-type encodings of trees in evolutionary algorithms

Many combinatorial problems that are to be solved with evolutionary algorithms look for an optimal tree that fulfils a number of constraints [18,9,5]. In addition, in the majority of cases, a given objective function, which represents the quality of the tree for the problem must be optimized. Many tree encoding schemes have been proposed in the literature [21,16], but the most commonly used, the Cayley-type encoding, is used in this paper; it includes the so called *Prüfer encoding* and the recently proposed *Dandelion code*.

In 1889, Cayley proved that in a complete labeled network with n nodes there are n^{n-2} different spanning trees. In 1918 Prüfer gave a constructive proof of Cayley's result, using a bijection between the spanning trees on n vertices and the strings of length $n-2$ over an alphabet of n symbols. Mathematically, for each $n \geqslant 2$, let C_n be the set of strings consisting of $(n-2)$ integers from the set $[1,n] = \{1,2,\ldots,n\}$, with repetitions allowed. For a given n, the strings C_n are known as *Cayley strings*, or *Cayley codes*. For each $n \geqslant 2$ let $\urcorner_n$ be the set of labeled trees on the vertex set $[1,n]$. Cayley's result showed that $|\urcorner_n| = n^{n-2}$, and $|C_n| = |\urcorner_n|$. Thus, the trees in $\urcorner_n$ can be put in one-to-one correspondence with the Cayley strings (C_n), and there are $(n^{n-2})!$ Cayley codes (different possible assignments of trees to Cayley strings).

Cayley-type encodings of trees, and specially Prüfer encoding, have been successfully applied to several problems requiring a tree or a spanning tree of a given graph, like [6,14] or [19]. Prüfer encoding has also sometimes been criticized, due to its poor results in other problems [22]. These results were explained by the poor locality of the mapping (small changes in the representation make large changes in the tree). In order to correct this, changes in the original Prüfer encoding were introduced by Neville in 1953. Much more recently, new Cayley-type encodings have been proposed to be used in evolutionary algorithms. One of these new encodings, known as the *Dandelion code*, has been studied in detail for its use within evolutionary algorithms dealing with spanning trees, [16,22]. The Dandelion code has been reported to exhibit better behavior than the Prüfer encoding in terms of locality (small changes in the representation make small changes in the tree) [16].

The rest of this section summarizes the Prüfer and Dandelion encoding algorithms for trees.

2.1. Prüfer encoding

The algorithm for decoding a Prüfer string is based on adding edges to an initially empty graph, following an order given by the degree of the nodes and the code string. The decoder can be summarized as follows:

- **Input:** A Prüfer code $C = (c_2, c_3, \ldots, c_{n-1})$.
- **Output:** The tree $T \in \urcorner_n$ corresponding to C.
- **Step 1:** Calculate the degree of each node d_i as $d_i = nr_i + 1$, where nr_i counts the number of times that node i appears in the code C.
- **Step 2:** Construct the tree $T \in \urcorner_n$ by adding edges to an initially empty graph, following the order of the minimum degree nodes (vector D) and the code C. When a given edge is included in the graph, then the degrees of the corresponding nodes must be decremented one unit in vector D. The tree is formed when all the edges have been included.

Consider Fig. 1 as an example. In this figure, the Prüfer code $C = (1,2,6,5,1,8)$ is decoded. First, we calculate the vector of degrees, D, which in this case is $D = (3,2,1,1,2,2,1,2)$. Then we start adding edges to an empty graph according to the nodes with minimum value of d_i, which in this case are nodes 3, 4 and 7. Since 3 is the smallest one, and $c_1 = 1$, we add an edge between the nodes (1,3). We decrement the corresponding nodes in vector D, to get $D = (2,2,0,1,2,2,1,2)$. Now we have to choose the minimum between nodes 4 and 7 (values different from 0). The minimum is 4, and $c_2 = 2$, so we add an edge between the pair (2,4). We decrement vector D, to get $D = (2,1,0,0,2,2,1,2)$, and then we choose between nodes 2 and 7. We choose 2, and $c_3 = 6$, so we add an edge between the pair (2,6). We continue applying this algorithm until vector D is a vector of 0s. The final tree can be seen in Fig. 1b.

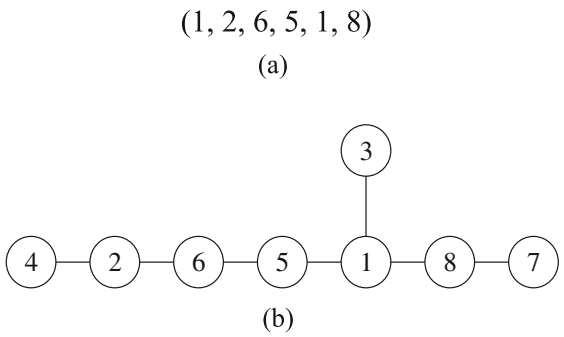

Fig. 1. (a) Example of a Prüfer code and (b) the final tree after the decoding process.

2.2. The Dandelion code

The Dandelion code is a Cayley-type encoding that has recently been described and used for encoding trees in genetic algorithms [16,22]. There are several decoding algorithms (string to tree) for the Dandelion code. In this paper, we use the so called *fast algorithm*, proposed by Piccioto in [17], which has also been used in [22]:

- **Input:** A Dandelion code $C = (c_2, c_3, \ldots, c_{n-1})$.
- **Output:** The tree $T \in \urcorner_n$ corresponding to C.
- **Step 1:** Define the function $\phi_C : [2, n-1] \to [1, n]$ such that $\phi_C(i) = c_i$ for each $i \in [2, n-1]$.
- **Step 2:** Calculate the cycles associated with the function ϕ_C, $Z_1, Z_2, \ldots, Z_k$. Let b_i be the maximum element in cycle Z_i. We assume that the cycles are recorded such that b_i is the rightmost element of Z_i and that $b_i > b_j$ if $i < j$.
- **Step 3:** Form a single list π of the elements in $Z_1, Z_2, \ldots, Z_k$, in the order in which they occur in this cycle list, from the first element of Z_1 to the last element of Z_k.
- **Step 4:** Construct the tree $T \in \urcorner_n$ corresponding to C by taking a set of n isolated vertices (labeled with the integers from 1 to n), creating a path from vertex 1 to vertex n by following the list π from left to right, and then creating the edge (i, c_i) for every $i \in [2, n-1]$ that does not occur in the list π.

We will illustrate this fast algorithm using the example in Fig. 2. This figure proposes the Dandelion code $C = (4, 6, 2, 5, 9, 1, 12, 6, 2, 9)$. Note that there are three cycles in this case: $Z_1 = (6, 9), Z_2 = (5)$ and $Z_3 = (2, 4)$. Also note that the order in which we have recorded these cycles follows the indications in Step 2 of the fast decoder algorithm. We next form the list $\pi = [6, 9, 5, 2, 4]$ and construct the first part of the tree T, starting from vertex 1, ending in vertex 12, and following the numbers in π. The rest of the tree is constructed by creating the corresponding edges (i, c_i) for those i that are not in the list π; in this case they are vertices 7, 3, 11, 10 and 8.

3. Problem formulation and lower bound for the OTNDP

The OTNDP looks for the tree network T with the minimum average number of bifurcations. Let k be the desired number of leaves in a tree with M nodes. Each leaf l_q is associated with the probability $\frac{1}{k}$ (we consider that there is an equal probability of reaching any final node l_q). Mathematically, the OTNDP can be defined using the adjacency matrix of the tree: Let A be the adjacency matrix of the tree, i.e., an $M \times M$ binary matrix, where $A_{ij} = 1$ if there is a link between nodes i and j, and $A_{ij} = 0$

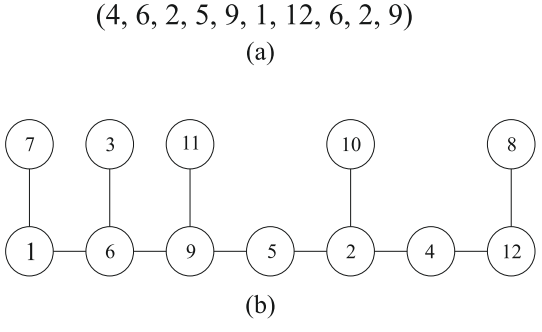

Fig. 2. (a) Example of a Dandelion code and (b) the final tree after the decoding process.

otherwise ($A_{ii} = 0$, we consider no node to be adjacent to itself). Starting from A, we define the path between the root node (r) and each of the tree leaves (l_q). Let N_q be the set of nodes $\{n_{q,i}\}$ that form this path:

$$N_q = \{n \in T / A(n_{q,i}, n_{q,i-1}) = A(n_{q,i}, n_{q,i+1}) = 1, n_{q,0} = r, n_{q,t_q} = l_q\}, \tag{1}$$

where $t_q = |N_q| - 1$.

Given these N_q, we look for the tree such that

$$\min_T g(T) = \left(\frac{\sum_{q=1}^{k} \left(\sum_{i=0}^{t_{q-1}} \left(\sum_{j=1}^{M} A(n_{q,i}, j) - 1 \right) + 1 \right)}{k} \right). \tag{2}$$

3.1. An example

Consider the small example tree given in Fig. 3. A tree with $k = 3$ leaves and $M = 5$ nodes must be evaluated. The adjacency matrix of this example is

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Since $k = 3$, there are 3 paths from the root to the leaves of the tree: $N_1 = \{1, 2, 4\}, N_2 = \{1, 2, 5\}$ and $N_3 = \{1, 3\}$. With these parameters, we can calculate the value of the function $g(T)$, $g(T) = \frac{10}{3}$.

3.2. A lower bound for the OTNDP based on N-ary trees

Consider an oriented-tree type network. Following the OTNDP formulation, we look for a tree network with the minimum average number of bifurcations between the root node and the leaves of the tree. In the case that the same probability is associated to each tree leaf, the network topology that minimizes the number of bifurcations is a N-ary tree (from each non-leaf node in the tree there are N links to the next level; see Fig. 4). Thus, at the ith tree level, the number of nodes is N^i. If k is the number of leaves and L is the tree depth, they are related by the following relationship:

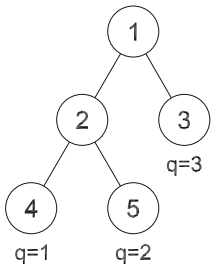

Fig. 3. Example of a tree with $k = 3$ (leaves) and $M = 5$ (nodes).

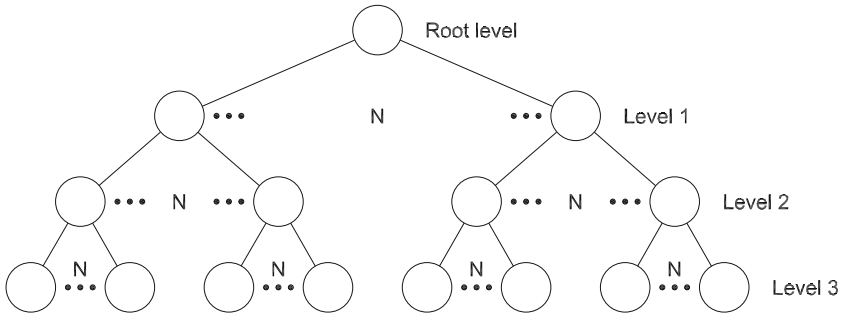

Fig. 4. Example of N-ary tree.

$$L = \frac{\ln k}{\ln N}. \tag{3}$$

This expression gives us the number of levels (tree depth) in a N-ary tree with k leaves. The number of bifurcations produced by each tree leaf in this case is

$$S = N \cdot L = N\frac{\ln k}{\ln N}. \tag{4}$$

If we wish to minimize the number of bifurcations with respect to N, we get:

$$\frac{dS}{dN} = \ln k \frac{\ln N - 1}{(\ln N)^2} = 0 \Rightarrow N = e. \tag{5}$$

Note that $N = e$ gives us a minimum for S:

$$\left.\frac{d^2S}{dN^2}\right|_{N=e} = \left.\frac{\frac{1}{N}(\ln N)^2 - (\ln N - 1)2\ln N}{(\ln N)^4}\right|_{N=e} = \frac{1}{e} > 0. \tag{6}$$

Therefore, the optimal value of N is $N = e$, that produces the minimum number of bifurcations

$$S = e \ln k. \tag{7}$$

But note that in a N-ary tree, the number of edges leaving from a node (N) must be a natural number, so we have calculated an ideal, unfeasible, tree network topology, which can be considered as a lower bound for the OTNDP.

3.3. A real application of the OTNDP

The general OTNDP can be specified in several applications, such as the design of computer or telecommunication networks, and other applications in which the solution is specified as a tree. One of these applications is the optimal design of interactive voice response systems (IVR) in call centers [1]. A typical call center is set up to provide a number of services to customers, that may vary from providing simple information to gathering and dealing with complaints or more complex transactions. In a call center, an interaction takes place between the customer and the system, which has to provide answers to the customers' queries. The interaction may be managed by a fully automatic system (the so called interactive voice response system) or require the intervention of a human agent.

A typical call center employs an IVR system as a front-end for its customers. The use of an IVR reduces the number of human operators needed to manage the call center. This results in savings in the overall workforce costs, which represent the majority of the call center costs. Note that many of the services provided by the call center require standardized answers (such as the answers to Frequently Asked Questions) or answers that result from database query searches (such as the balance on a bank account or the state of a purchase order) and therefore do not need to involve human intervention.

An IVR system can be shown as a tree network in which the leaves are the final services to users. Fig. 5 shows an example of an IVR system in which the nodes are notated as P_x, and the average time of the users in each node is also shown.

Note that an IVR system is an example of a tree network that can be optimized by the OTNDP if all the services are considered to be of equal probability. In this case, the lower the number of bifurcations in the tree, the lower the average time of users in the IVR.

4. Proposed improvements in EAs with Cayley-type encodings for the OTNDP

The OTNDP can be solved using evolutionary algorithms with Cayley-type encodings. However, several adaptations are necessary in these encodings the OTNDP successfully. In this section, we present and discuss these adaptations, which are the main objective of this paper.

The first adaptation of Cayley-type encodings to the OTNDP is related to the root node included in the OTNDP. Note that the Prüfer and Dandelion codes consider the case of un-oriented trees, in which there is no root node to start the tree. In the OTNDP, however, one node acts as the root node, and the rest are attached to it directly or as subbranches in lower nodes. Thus, a given encoding of a Prüfer or Dandelion code represents $n - k$ possible directed trees (the k tree leaves cannot be considered as the root node). To solve this point, after decoding the Cayley string (coming from a Prüfer or Dandelion code), we evaluate the $n - k$ possible different directed trees and keep the best tree in terms of the objective function represented by the Cayley string; see Fig. 6 as an example.

Another adaptation to consider in the oriented-tree case as well as in the OTNDP, is that k of the nodes of the oriented-trees must be leaves. Thus, we need to ensure this point in the evolutionary algorithms. We implement a procedure based on a property of the Cayley-type encodings: the leaves of a tree in the Cayley encoding are the nodes whose numbers do not appear in the string. As an example, Fig. 2a shows a Dandelion code in which numbers 7, 3, 8, 10 and 11 are not included, and consequently the nodes with these numbers become the tree leaves (Fig. 2b). In this way, the procedure consists of not using the first k node numbers (1 to k) in the random generation of the initial codes (at the beginning of the evolutionary algorithm), nor in the mutation operator.

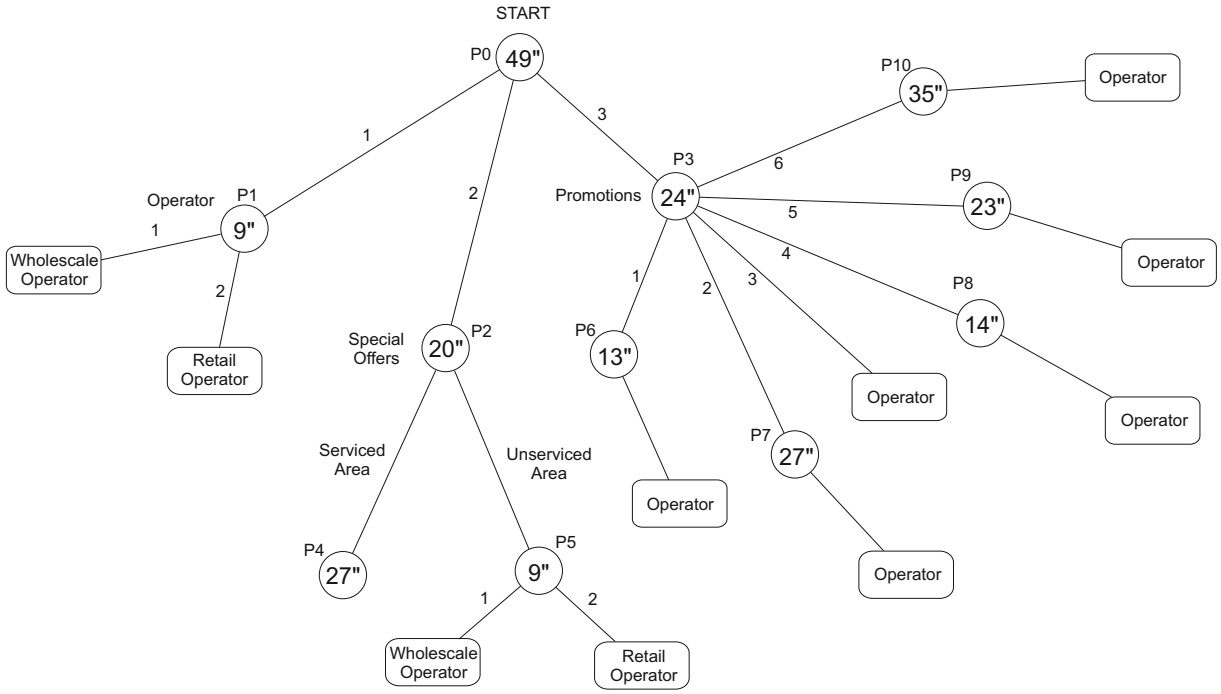

Fig. 5. Example of an IVR system in a call center. In each node we show the average time (in seconds) that a given user spends listening to option messages. Also, in each link we show the number of key that the user must press in order to advance to the required service.

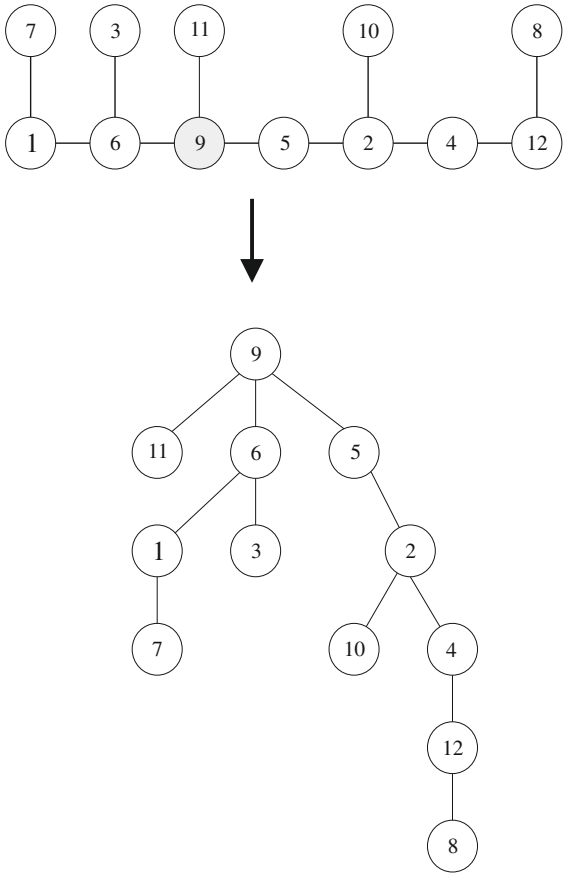

Fig. 6. Example of the encoding of an oriented-tree using the Dandelion code, assuming that the tree with root in node 9 is the best one.

In the evolutionary algorithm we use fixed-length tree encodings (trees with a given constant number of nodes). However, these constant-length codes allow us to perform a search for trees with any number of nodes (under a given maximum, of course). That is because a tree that has some leaves with associated numbers larger than k (and consequently a tree with more than k leaves) is equivalent to another tree without these leaves (both trees produce the same value of the fitness function because only the first k leaves of the tree are relevant to calculate the average number of bifurcations). An example of this is in the tree of Fig. 7. In this example $k = 7$; thus, the nodes 8, 11 and 12 (all larger than k), which appear in the tree on the left side, do not influence the number of bifurcations, so the tree on the right side is completely equivalent.

4.1. Bounding the individuals' encoding length

Previous adaptations of EAs for the OTNDP can be extended by means of setting a bound for the length of the encoding (or equivalently the number of nodes of the tree) that guarantees the optimal solution is included. For this we use the properties of the objective function given by Eq. (2). Note that this function depends on the length of the routes between the root node and each leaf node, in such a way that the shorter the routes, the lower the objective function and the better the associated individual. Consider, therefore, Fig. 8. Note that due to the presence of node 8, nodes 3 and 4 the left tree have routes to the root node larger than the corresponding nodes in the right tree. Thus, the right tree solution is better than the left one in terms of the objective function given by Eq. (2). We refer to any node in a tree with the same structure as node 8 in Fig. 8 as a *dummy node*. Following this example, the idea is that given a tree containing dummy nodes, it is always possible to find a better tree in terms of the objective function by removing these dummy nodes.

Consider again Fig. 8. Note that the final tree, obtained when the dummy node is removed, is formed by $2k - 1$ nodes (recall that k is the number of leaves in the tree), which is 7 in this case. The following theorem generalizes this result:

Theorem 1. *Given a rooted tree with k leaves, the maximum number of nodes in the tree that does not contain dummy nodes is* $2k - 1$.

Proof. Let us suppose that the number of nodes in a tree with k leaves is N, so the number of links in the tree is $N - 1$. To ensure none of the nodes are dummy nodes, each node that is not a leaf must have at least two nodes hanging from it. The total number of nodes that are not leaves in the tree is denoted as B. Therefore, the following expression holds in such a tree:

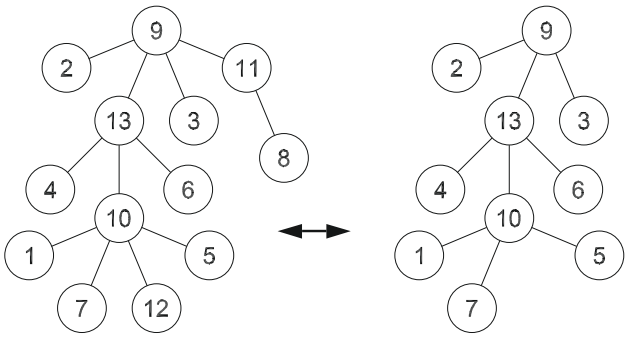

Fig. 7. Example of leaves' removal in an unfeasible tree of 7 leaves.

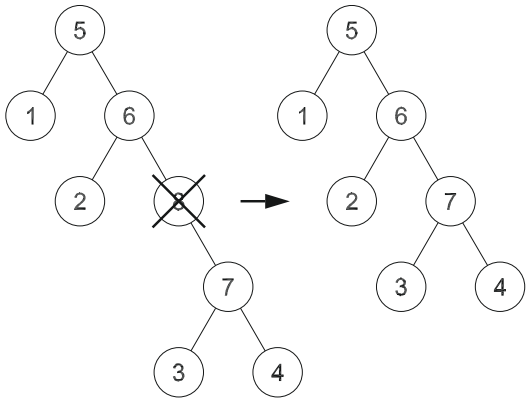

Fig. 8. Example of a *dummy node* (node 8 in this case) in a tree, and the improved tree obtained when it is removed.

$$B \leqslant \left(\frac{N-1}{2}\right). \tag{8}$$

Note that if this expression is not fulfilled, there would not be enough links to fulfil the condition that at least two nodes must hang from any node that is not a leaf. Note that:

$$N = B + k \leqslant \left(\frac{N-1}{2}\right) + k, \tag{9}$$

so,

$$2N \leqslant N - 1 + 2k, \tag{10}$$

and finally

$$N \leqslant 2k - 1. \quad \square \tag{11}$$

Note that we use Theorem 1 in order to obtain the minimum possible encoding length in our algorithms, and at the same time we ensure that the optimal solution is included in the search space.

4.2. Evolutionary algorithm operators and parameters

Finally, we briefly comment on small changes carried out in the evolutionary operators (mainly the mutation operator) for the OTNDP and also on the parameters used by the EAs tested in the experimental section. The EA with Cayley encoding proposed for the OTNDP is structured in the traditional form of a classic genetic algorithm [10], with procedures for Selection, Crossover and Mutation. The Selection procedure is the standard roulette wheel, in which the probability of survival of a given individual is proportional to its fitness value. A two-point Crossover operator is applied, where the parents are randomly chosen among the individuals in the population. The Mutation operator is slightly different than in a standard evolutionary algorithm. Recall that we reserve the first k numbers to represent the leaves of the tree, so they cannot appear in the encoding. Thus, the Mutation operator chooses one point in the Dandelion string and randomly replaces it with another integer number randomly chosen in $[k+1, n-2]$. Regarding the parameters of the algorithm, we have implemented the standard values of Crossover ($P_c = 0.6$) and Mutation ($P_m = 0.01$) operators. The population size is 50, and the number of generations has been fixed to 500, after which we keep the best tree encountered so far.

5. Experiments and results

The experimental part of the paper has been structured in two different parts. First, we test the improvements proposed in this paper over the Cayley-type encodings described in Section 2 (Prüfer encoding and the Dandelion code) to handle the OTNDP. Several randomly generated OTNDP instances have been tackled, and a discussion of the results is carried out. Second, a real application of the OTNDP is shown in the optimal design of an IVR structure for an Italian telecommunications company (TIM).

5.1. Experiments in randomly generated instances

To test the performance of the Cayley-type encodings for the OTNDP, we have implemented both encodings within an evolutionary algorithm. The design of 30 OTNDP instances from 10 to 300 leaves has been carried out using the evolutionary algorithm with both encodings. We have compared the results obtained with the theoretical lower bound given in Section 3.2. For each instance (with a given number of leaves) we have launched the evolutionary algorithm with Prüfer and Dandelion encodings 30 times each, keeping the best and mean value in each case.

Fig. 9 shows the best objective function value obtained by the evolutionary algorithm with the Prüfer and Dandelion encodings, and the lower bound for each instance. It is interesting that both encodings produce very similar results in small size instances, up to 100 nodes. However, when the size of the instances grows, the Dandelion encoding produces better results than the Prüfer one. Fig. 10 shows the mean value of the 30 experiments for each instance. The differences in mean values are more significant than in the best value. If we compare the results with the theoretical lower bound, both encodings produce solutions of good quality. In order to perform this comparison, we have obtained the values of the parameters $\Delta_D = \frac{D-C}{C}$, and $\Delta_P = \frac{P-C}{C}$, where D stands for the mean value obtained with the evolutionary algorithm using the Dandelion encoding, P stands for the mean value obtained with the evolutionary algorithm using the Prüfer encoding and C stands for the lower bound value. Both parameters are given as percentages of increment, that provide an estimation of how far the evolutionary algorithm's solution (with different encodings) is from the lower bound. Fig. 11 shows the differences between the evolutionary algorithm with Prüfer encoding and Dandelion code, taking the lower bound as reference. It is easy to see that in instances up to 100 nodes, the differences between the encodings are not remarkable. However, the evolutionary algorithm with the Dandelion code outperforms the Prüfer encoding when the size of the problem grows; for the largest problem attempted (300 leaves), the difference is about 4%.

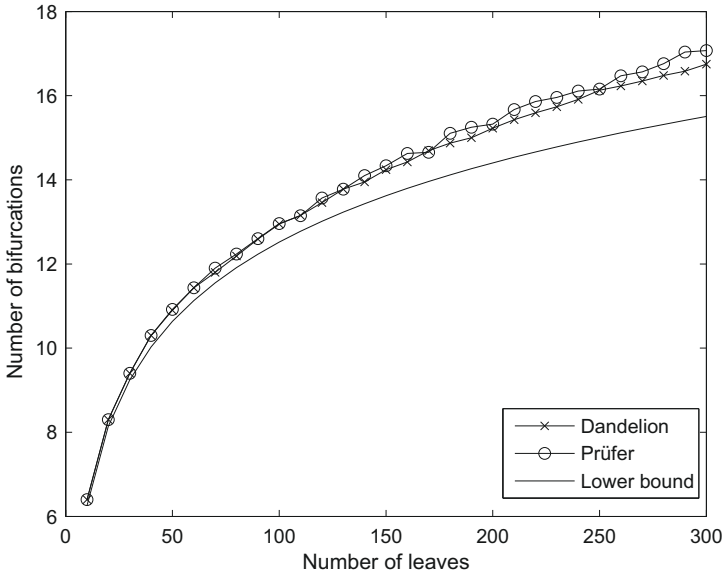

Fig. 9. Comparison of the best values obtained using the evolutionary algorithm with Dandelion code and the Prüfer encoding. The lower bound for the problem is depicted as a continuous line.

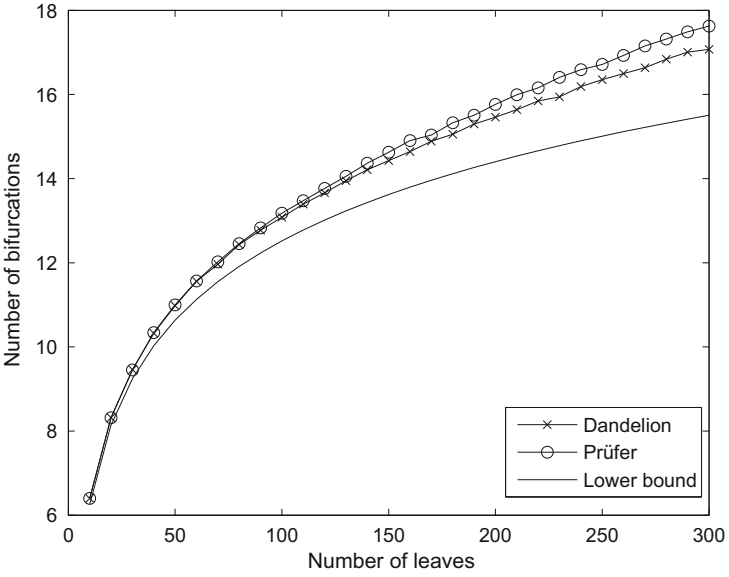

Fig. 10. Comparison of the mean values (over 30 runs) obtained using the evolutionary algorithm with Dandelion code and the Prüfer encoding. The lower bound for the problem is depicted as a continuous line.

Finally, to check whether the observed differences between the two encodings are statistically significant, a Mann–Whitney test (also known as a U-test [13]) was performed on the data obtained by the two algorithms. The choice of a non-parametric test such as the U-test allows us to remove the hypothesis of normality and even of symmetry for the probability density function of the data. As the average data in Fig. 11 suggest, we compared the null hypothesis that the two encodings perform the same against the alternative hypothesis that the Dandelion encoding performs significantly better. The ties have been managed using the average rank criterion. The resulting test statistics are shown in Fig. 12, where the horizontal line shows the critical region bound at the 5% significance level (i.e., the 95% Gaussian percentile), so that the null hypothesis is rejected if the test statistic is larger than that level. The visual impression provided by the average values of Fig. 11 is confirmed, since the difference between the two encodings (in favor of Dandelion) is significant when the number of nodes gets larger than 60 (with the exception of the case for 80 nodes, where the null hypothesis would be rejected with a 13%

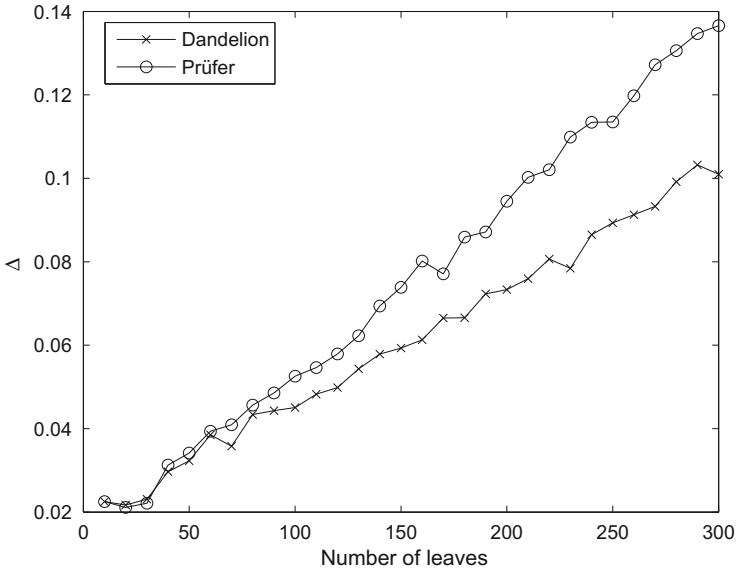

Fig. 11. Comparison of the mean values (over 30 runs) obtained using the evolutionary algorithm with Dandelion code and the Prüfer encoding, referred to the lower bound.

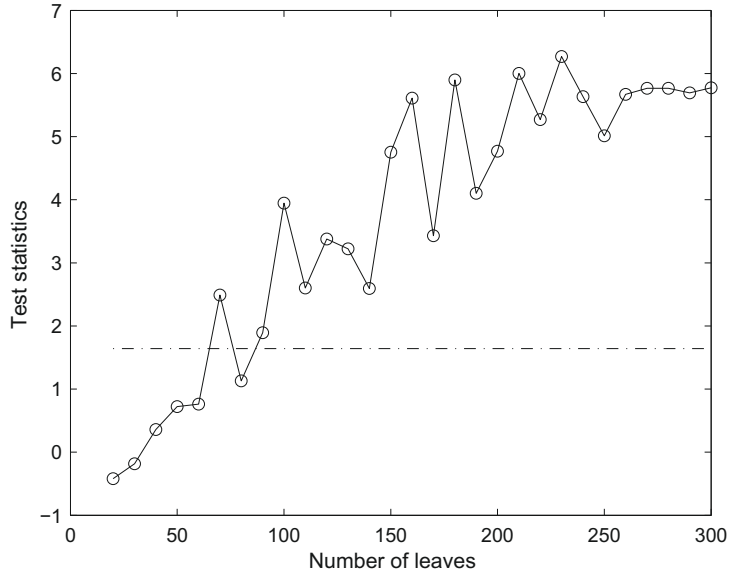

Fig. 12. Results of the U-test over the data obtained by the compared algorithms.

significance level). Thus, the Dandelion encoding produces a more scalable algorithm, which is able to obtain good results as the size of the instances grows.

5.2. Experiments in the optimization of a real IVR system

In order to show the performance of our approach in a real case, we apply it for the design of a call center (the IVR part) for the Italian mobile operator TIM. In this paper we examine the redesign of the customer care service call center, offered by TIM through its "green" number 119. The IVR used by TIM consists of 50 different services, structured as shown in Fig. 13. This initial service tree is representative of a call center managing a very large number of customers each day, with a very wide range of services. The average number of bifurcations (given by Eq. (2)) of the non-optimized IVR system used by TIM is 14,860 bifurcations. Note that the lower bound for this real system is 10,634 bifurcations (Eq. (7)). We have applied the best

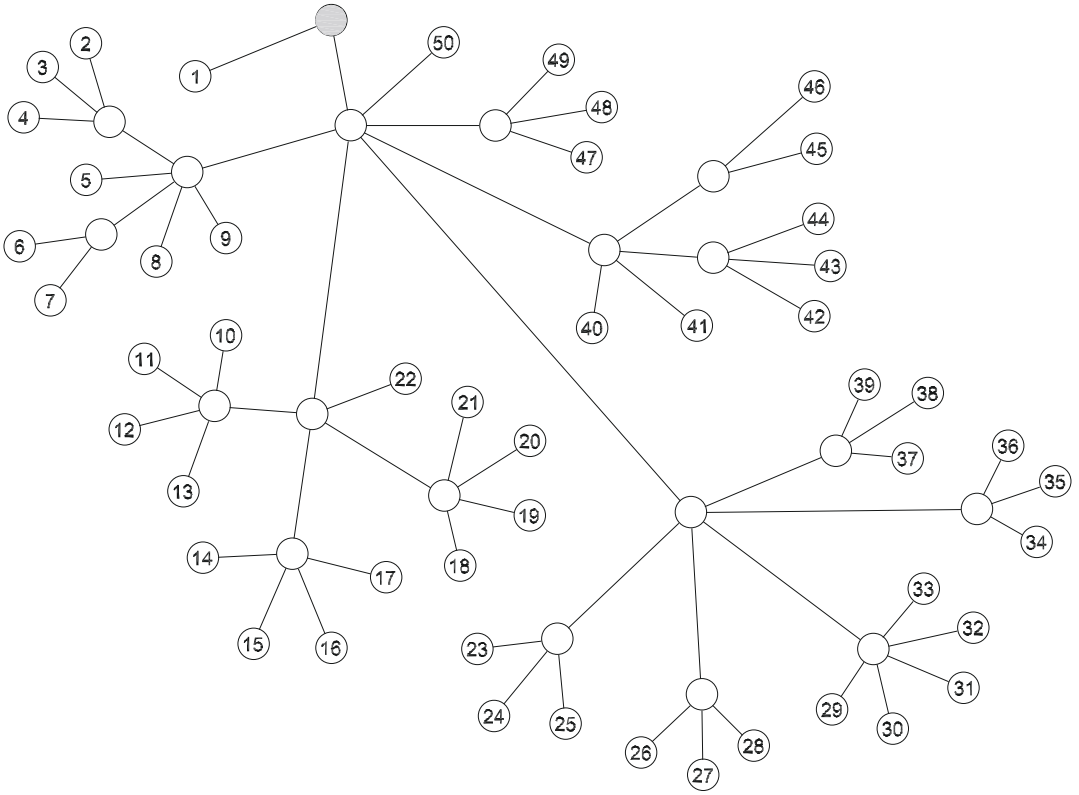

Fig. 13. Initial service tree used by TIM (not optimized).

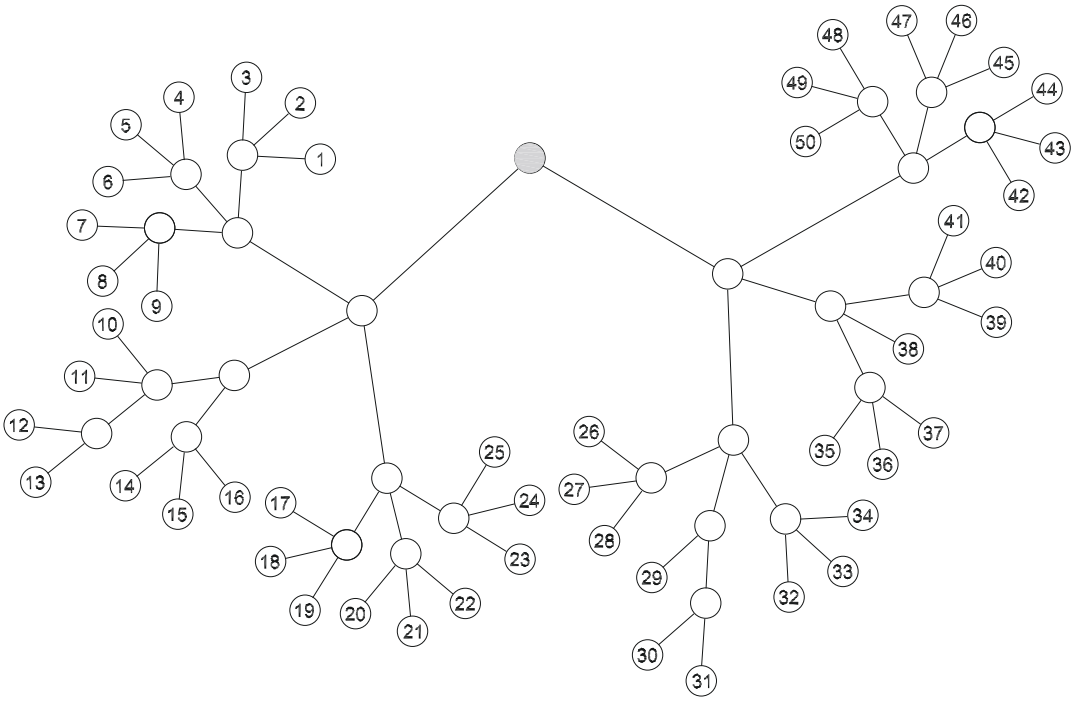

Fig. 14. Optimized tree with the Dandelion EA for the TIM call center.

proposed algorithm for the OTNDP (EA with Dandelion encoding) to the optimization of this system. Fig. 14 shows the best tree obtained for the TIM call center with the proposed approach. The average number of bifurcations in the optimized system (objective function value) is 10,900 bifurcations, which is quite close to the system's lower bound and better than the number of bifurcations of the IVR used by TIM.

6. Conclusions

A problem of oriented-tree design (OTNDP) is solved in this paper using evolutionary algorithms (EA) with two different tree encoding strategies, both of them using Cayley codes: the Prüfer encoding and the Dandelion code. In this paper we have shown how to adapt an EA with Cayley encoding to the OTNDP, how the Dandelion code with the corresponding adaptations is a powerful tool for solving this problem, and also a real application of the proposed algorithms in the design of a call center. The OTNDP is a general problem that can be used to define other important problems in several fields, mainly in the design of communication networks. The adaptations presented in this paper allow the successful application of EAs to these problems as powerful solution methods.

Acknowledgements

The authors would like to thank Editor in Chief and anonymous reviewers for their comments and hints to improve this paper. This work has been partially supported by Comunidad de Madrid, Universidad de Alcalá and Ministerio de Educación, through Projects CCG08-UAH/AMB-3993 and TEC2006/07010. E.G. Ortiz-García is supported by Universidad de Alcalá, under the University F.P.I. Grants program. Á.M. Pérez-Bellido is supported with a doctoral fellowship by the European Social Fund and Junta de Comunidades de Castilla la Mancha, in the frame of the Operating Programme ESF 2007-2013.

References

[1] J. Anton, The past, present and future of customer access centers, International Journal of Service Industry Management 11 (2) (2000) 120–130.
[2] I. Averbakh, O. Berman, Algorithms for path medi-centers of a tree, Computers and Operations Research 26 (1999) 1395–1409.
[3] J.J. Bartholdi III, D. Eisenstein, Y.F. Lim, Bucket brigades on in-tree assembly networks, European Journal of Operational Research 168 (2006) 870–879.
[4] S. Caminiti, E.G. Fusco, R. Petreschi, A bijective code for k-trees with linear time encoding and decoding, in: Proceedings of the International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE'07), Hangzhou, China, 7–9 April, 2007.
[5] G. Chen, S. Chen, W. Guo, H. Chen, The multi-criteria minimum spanning tree problem based genetic algorithm, Information Sciences 177 (22) (2007) 5050–5063.
[6] S. Choi, B. Moon, A graph-based Lamarckian–Baldwinian hybrid for the sorting network problem, IEEE Transactions on Evolutionary Computation 10 (2) (2005) 105–114.
[7] S.A. Choudum, R. Indhumathi, On embedding subclasses of height-balanced trees in hypercubes, Information Sciences 179 (9) (2009) 1333–1347.
[8] C. Eom, G. Oh, S. Kim, Deterministic factors of stock networks based on cross-correlation in financial market, Physica A: Statistical Mechanics and its Applications 383 (1) (2007) 139–146.
[9] A. Fatemi, K. Zamanifar, N. NematBakhsh, A new genetic approach to construct near-optimal binary search trees, Applied Mathematics and Computation 190 (2) (2007) 1514–1525.
[10] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[11] J.F. Guan, H. Yang, S.C. Wirasinghe, Simultaneous optimization of transit line configuration and passenger line assignment, Transportation Research Part B: Methodological 40 (10) (2006) 885–902.
[12] J.M. Latorre, S. Cerisola, A. Ramos, Clustering algorithms for scenario tree generation: application to natural hybro inflows, European Journal of Operational Research 181 (2007) 1339–1353.
[13] J.P. Lecoutre, P. Tassi, Statistique Non Parametrique et Robustesse, Economica Press, Paris, 1987.
[14] C. Lo, W. Chang, A multiobjective hybrid genetic algorithm for the capacitated multipoint network design problem, IEEE Transactions on Systems, Man, and Cybernetics, Part B 30 (3) (2000) 461–470.
[15] T. Öncan, Design of capacitated minimum spanning tree with uncertain cost and demand parameters, Information Sciences 177 (20) (2007) 4354–4367.
[16] T. Paulden, D.K. Smith, From the Dandelion code to the rainbow code: a class of bijective spanning tree representations with linear complexity and bounded locality, IEEE Transactions on Evolutionary Computation 10 (2) (2006) 108–123.
[17] S. Piccioto, How to Encode a Tree, Ph.D. Dissertation, University of California, San Diego, 1999.
[18] F. Rothlauf, D.E. Goldberg, A. Hinzl, Network Random Keys – A Tree Network Representation Scheme for Genetic and Evolutionary Algorithms. IlliGAL Internal Report, No. 2000031, 2000. Available: <http://www.illigal.ge.uiuc.edu>/.
[19] K. Sawada, R. Wilson, Models of adding relations to an organization structure of a complete K-ary tree, European Journal of Operational Research 174 (2006) 1491–1500.
[20] S. Shioda, K. Ohtsuka, T. Sato, An efficient network-wide broadcasting based on hop-limited shortest-path trees, Computer Networks 52 (17) (2008) 3284–3295.
[21] S. Soak, D.W. Corne, B. Ahn, The edge-window-decoder representation for tree-based problems, IEEE Transactions on Evolutionary Computation 10 (2) (2006) 124–144.
[22] E. Thompson, T. Paulden, D.K. Smith, The Dandelion code: a new coding of spanning trees for genetic algorithms, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 91–100.
[23] J. Urrutia, Local solutions for global problems in wireless networks, Journal of Discrete Algorithms 5 (3) (2007) 395–407.
[24] B.Y. Wu, K.M. Chao, Spanning Trees and Optimization Problems, Chapman & Hall, London, UK, 2004.