Evolutionary Optimization of Service Times in Interactive Voice Response Systems

Sancho Salcedo-Sanz, Maurizio Naldi, *Senior Member, IEEE,* Ángel M. Pérez-Bellido, Jose A. Portilla-Figueras, and Emilio G. Ortíz-García

***Abstract*—A call center is a system used by companies to provide a number of services to customers, which may vary from providing simple information to gathering and dealing with complaints or more complex transactions. The design of this kind of system is an important task, since the trend is that companies and institutions choose call centers as the primary option for customer relationship management. This paper presents an evolutionary algorithm based on Dandelion encoding to obtain near-optimal service trees which represent the structure of the desired call center. We introduce several modifications to the original Dandelion encoding in order to adapt it to the specific problem of service tree design. Two search space size reduction procedures improve the performance of the algorithm. Systematic experiments have been tackled in order to show the performance of our approach: first, we tackle different synthetic instances, where we discuss and analyze several aspects of the proposed evolutionary algorithm, and second, we tackle a real application, the design of the call center of an Italian telecommunications company. In all the experiments carried out we compare our approach with a lower bound for the problem based on information theory, and also with the results of a Huffman algorithm we have used for reference.**

***Index Terms*—Call center design, Dandelion codes, evolutionary algorithms, interactive voice response systems, tree encoding.**

I. Introduction

CALL CENTERS are now an established way of managing the relationship with customers, and they are used by many companies. Though estimating their number is a nearly impossible task, a recently reported estimate is of about 300 000 all over the world, with a continuous growth [1], since it can be envisaged that companies and public institutions will choose call centers as a primary means of customer relationship management.

Manuscript received October 31, 2007; revised November 4, 2008, June 3, 2009, and August 28, 2009. Date of publication March 29, 2010; date of current version July 30, 2010. This work was supported in part by Comunidad de Madrid, Universidad de Alcalá, and the Ministerio de Educación of Spain under Projects CCG08-UAH/AMB-3993 and TEC2006-07010. Á. M. Pérez-Bellido is supported by Junta de Comunidades de Castilla la Mancha, through an FPI fellowship co-funded by the European Social Fund. E. G. Ortiz-García is supported by the Ministerio de Educación of Spain, through an FPU fellowship.

S. Salcedo-Sanz, Á. M. Pérez-Bellido, J. A. Portilla-Figueras, and E. G. Ortíz-García are with the Departamento de Teoría de la Señal y Comunicaciones, Escuela Politécnica Superior, Universidad de Alcalá, Alcalá de Henares 28805, Madrid, Spain (e-mail: sancho.salcedo@uah.es; angel.perezb@uah.es; antonio.portilla@uah.es; emilio.ortizg@uah.es).

M. Naldi is with the Dipartimento di Informatica, Sistemi e Produzione, Universitá di Roma Tor Vergata, Rome 00133, Italy (e-mail: naldi@disp.uniroma2.it).

Digital Object Identifier 10.1109/TEVC.2009.2039142

A typical call center is set up to provide a number of services to customers, which 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 (IVR) system) or require the intervention of a human agent. The customer's satisfaction evaluated over the full service cycle is one of the most important parameters when judging the overall call center's quality [2], [3].

A simplified approach is taken in [4], for the case of call centers managed by human agents, where the service quality is measured along two dimensions: qualitative (psychological) and quantitative (operational). The former relates to the way in which service is provided and perceived (usefulness of answers, friendliness of the agent, etc. [5]). The latter is more related to service accessibility. Thus, the quality of service is typically embodied by a single parameter (the time spent waiting for the agent, for example), though it can also be defined through different parameters, such as the average waiting time or one or two of its percentile values. In this context, the most important effect of the desired quality level is to require a proper level of staffing. In addition, the definition of the correct dimensioning of the system on the basis of the desired waiting time has required a large number of research efforts [4].

However, if the interaction with the customers takes place also through automated systems rather than by human agents alone, we cannot focus just on the time spent waiting for the agents. In this case many answers are provided directly by the IVR, and so the time spent within the IVR becomes relevant for the overall service quality. In today's call centers the IVR is getting more and more important, mainly due to the need to streamline the staff: the usage of IVRs accounted for 38% of all calls handled in 2000, with a growing trend to over 70% in 2005 [6], and is typically higher in some contexts, e.g., in call center for financial services, where the percentage of self-served calls through IVR was 65% already in 1999 [7]. The heavier reliance on IVRs implies that a longer slice of the service time is spent by the customer navigating through the IVR menu [8]. Greater attention should therefore be devoted to the proper design of the IVR in order to minimize the time that a customer is within the system.

A few works have dealt with the optimal design and management of call centers so far. Extensive reviews are contained

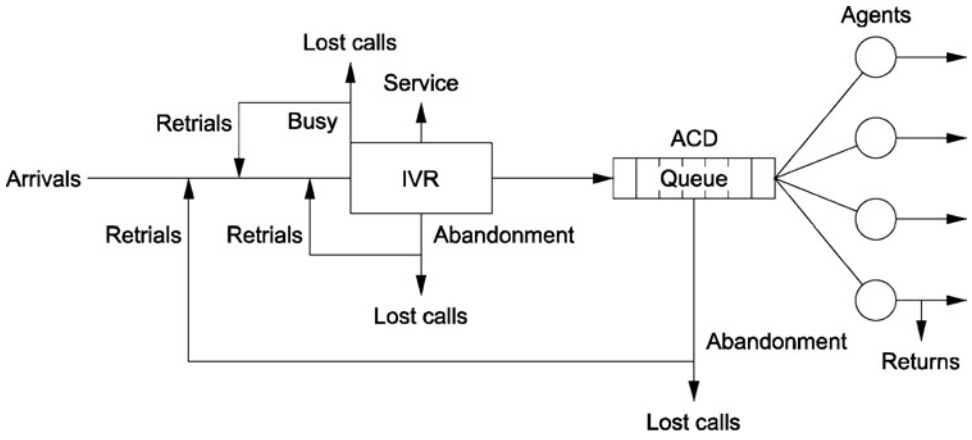

Fig. 1. Service chain in a typical call center.

in [4] and [19]. Among the most recent works we can cite [9], [10], and [20], which examine different queuing models for call center staffing. On the other hand, in [8] the tree structure of the service menu proposed by IVRs is highlighted, and its relationship with quality-of-service parameters is considered.

In this paper, we propose an evolutionary algorithm based on Dandelion encoding [12], to optimally design IVR systems. The Dandelion encoding has been recently proposed as a powerful method to encode trees in evolutionary algorithms, with good properties of locality under evolutionary operators [13]. Several modifications to the basic Dandelion decoding algorithm are needed in order to adapt it to the specific characteristics of IVR design systems, and to improve the performance of the algorithm. These modifications are included due to the following facts: first, the IVR design problem needs to encode rooted trees, whereas the basic Dandelion encoding manages trees without root, so a small modification is needed to solve this point, and second, we introduce two search space size reductions by using equivalent trees to the specified by the Dandelion string. This improves considerably the performance of the proposed evolutionary algorithm. Our Dandelion encoded evolutionary algorithm has been tested on the design of several call centers based on IVR, involving from 50 up to 300 services, which may be considered as large and very large systems. Also, a real application, the design of the call center of the Italian company Telecom Italia Mobile (TIM) has been carried out.

The rest of the paper has the following structure. The next section provides an overview of how to design IVR systems, and shows that this problem is equivalent to the design of the rooted tree of minimum average service time, in which the leaves are the services. Section III specifically defines the problem, and relates it with the source coding theorem which has been formulated in the context of digital communication systems. In this section, a lower bound based on the entropy of the systems is presented, and the Huffman algorithm is proposed as reference algorithm. Section IV presents the Dandelion encoded evolutionary algorithm we propose in this paper. Also the case of rooted trees and the search space size reduction procedures are introduced at this point. In Section V, we test the performance of our approach by solving different IVR design instances, with different number and probability distributions of services. This section also analyzes other aspects of the algorithm, such as the performance of the search space reductions, the performance of the algorithm with different operators and a detailed analysis of the computation time of the approaches analyzed in the paper. Finally, Section VI gives some final conclusion on the work carried out and some lines for future research.

II. Service Trees in Call Centers

A typical call center employs an IVR system as a front-end for its customers. The use of an IVR allows to reduce the number of human operators needed to manage the call center. This results in savings on the overall workforce costs, which represent the major part of the costs incurred in call center operations. 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 a human intervention. The range of services amenable to be directly provided by an IVR is therefore quite wide. In some cases the IVR is employed not only to provide an automatic response, but also to select the most appropriate set of operators for the service requested. Hence, of all the services provided by the call center, some will be provided by an automatic system, while others will require a talk with a human operator.

The IVR typically represents the first stage in the customer-call center interaction, so that the human intervention is configured as the last chance. The complete service chain, including both the IVR and the operator-assisted portions, is reported in Fig. 1 [4].

After dialing the call center phone number, the customer is driven through a menu listing a number of options (leading to different services) and providing a code to signal the customer choice. At each stage the users can choose (by dialing the appropriate code) among a number of options, which may either provide them with the required service (in the form of an answer uttered by a synthetic voice or by the connection to

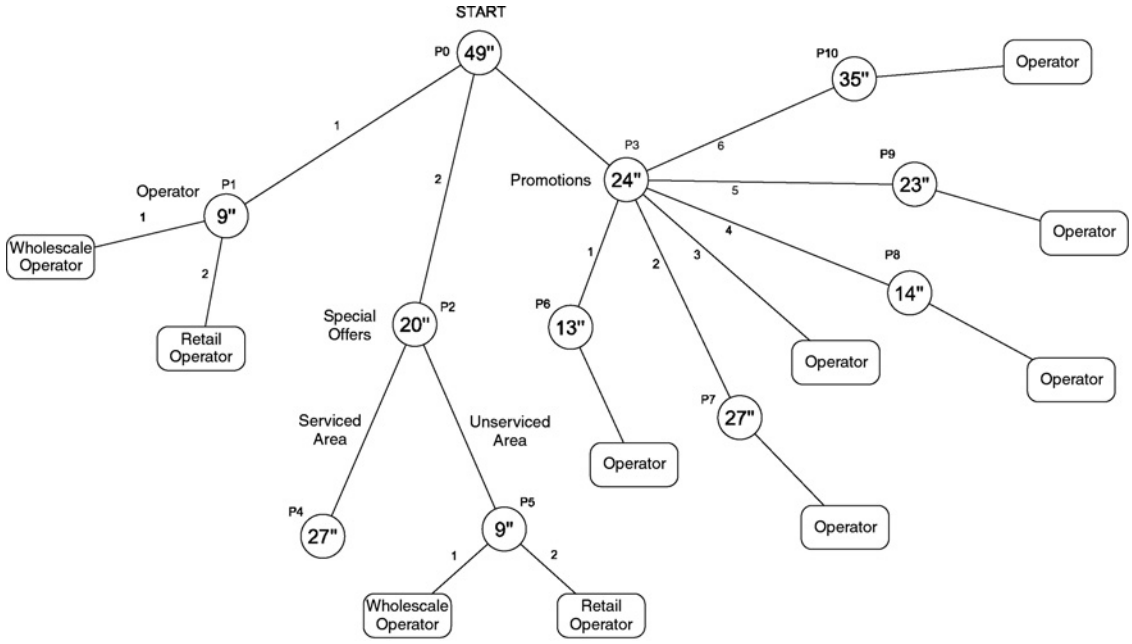

Fig. 2. Example of service tree.

a human operator) or lead them to a further option choosing stage. The service therefore requires the users to go through a number of stages where they have to exercise a choice. The guided navigation through the service menu can be described by a graph in the form of a *service tree*, where each node represents a message announcing the various options (the node's cost is the message duration) and the links represent the possible options that exist, leading either to the service provision or to a new option menu further down the service tree. The tree leaves represent the services finally delivered to the user.

An example of service tree is presented in Fig. 2. It describes the current service tree of the Customer Service of a prominent Italian telecommunications operator, as reported in [8]. The circles representing the nodes include the message duration, expressed in seconds. The links include the code to be dialed to choose the corresponding option. In this tree, the number of leaves (including the recurring leaves, which have not been included for the sake of readability) are 25.

As can be seen, the number of options announced within a message can be quite variable (in the example of Fig. 2 it goes from a minimum of 2 in the node *Operator* to a maximum of 6 in the node *Promotions*). In the following, we refer to the case where the number of options in each announcement is constant (say k) as a regular k-ary tree. In the general case of an IVR with M services (i.e., a service tree with M leaves), we indicate by p_i the probability of the ith service to be requested and by t_i the time needed to get it. It is important to point out that this time t_i is given by the sum of the durations of the announcements to be listened in order to get the service i, which in our examples are considered to be equal for all the announcements, so differences in t_i are given by the number of messages to be listened in a given route to a service. With these definitions, the most relevant figure of merit of the IVR is the average service time, defined as $\sum_{i=1}^{M} t_i p_i$, similarly to the widespread usage in call centers (see, e.g., [14] and [15]). Therefore, the optimal design of an IVR system consists of, given a set of services with the corresponding probabilities, obtaining a service tree (T) which minimizes the following objective function:

$$f(T) = \sum_{i=1}^{M} t_i p_i. \tag{1}$$

III. The Service Tree Design Problem and Its Relation to Source Coding

Since the number of services is typically set in advance, the tree design problem is basically the search for a grouping of services that minimizes the resulting average service time given by (1); if we move a service up in the tree, we lower the duration of the message (the *donor* node) announcing that service (and therefore reduce the time needed to get the services announced in the nodes child to the donor node), but at the same time we add to the duration of the message (the *recipient* node) where the service is now announced (and therefore increase the time needed to get the services announced in the nodes child to the recipient node). The final result may be positive or negative depending on the tree structure and on the probability values associated with the different services. Note that the problem is similar to assign codewords to a set of messages to be transmitted (digital transmission in communications networks) so that the mean length of the codeword is minimized. The parallel can be made more precise if we consider a binary tree, e.g., in Fig. 3.

If we assign to either of the branches leaving each node the symbol 0, and to the other branch the symbol 1, the path from the root message to the final service can be described by a codeword whose length represents the number of tree levels traversed. For example, if we assign the symbol 0 to

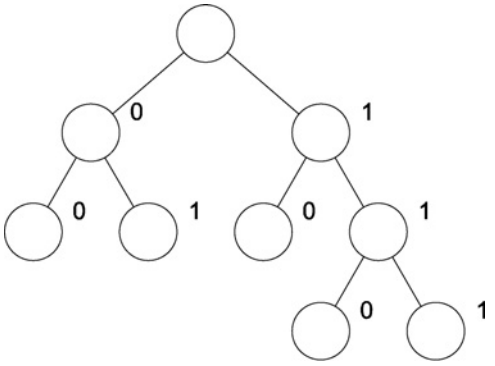

Fig. 3. Regular binary service tree.

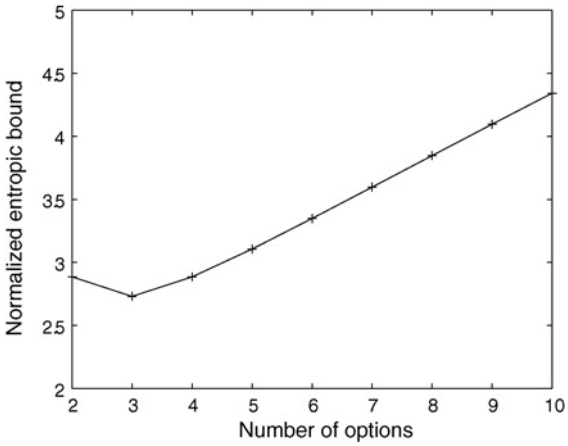

Fig. 4. Entropic bounds for the service time.

the left branch, the service 4 (assigning numbers to services starting from the leftmost leaf of the tree) is represented by the codeword 110, whose length is 3. If the duration of each message is constant, the codeword length is proportional to the service time. The problem of minimizing the average service time is therefore equivalent to minimizing the average codeword length. Of course, the binary codes case (binary trees) is readily extended to k-ary codes (k-ary trees). This analogy allows us to use the well-known results obtained in the past for coding. In particular, we can use the noiseless coding theorem [11] to establish a lower bound for the average codeword length, and therefore the average service time. That theorem states that the average codeword length $\bar{l}$ using an alphabet of k symbols is always larger than the uncertainty measure represented by the entropy of the system. For a set of M services (whose probabilities are $p_1 \leq p_2 \leq \ldots \leq p_M$) we have

$$\bar{l} \geq -\sum_{i=1}^{M} p_i \log_k (p_i). \tag{2}$$

In our case, we can derive the lower bound for the average service time for a regular k-ary tree with a portfolio of M services (whose probabilities are again $p_1 \leq p_2 \leq \ldots \leq p_M$)

$$\overline{T} = \bar{l} k d_{\text{ann}} \geq -k d_{\text{ann}} \sum_{i=1}^{M} p_i \log_k (p_i) \tag{3}$$

where d_{ann} is the common duration of the announcements for each option. It can be seen that the lowest lower bound is reached for $k = 3$, which is therefore an absolute benchmark. In fact, we can write the lower bound in the following form:

$$\overline{T} \geq -k d_{\text{ann}} \sum_{i=1}^{M} p_i \log_k (p_i) = -d_{\text{ann}} \frac{k}{\ln (k)} \sum_{i=1}^{M} p_i \ln (p_i) \tag{4}$$

where the dependence on the tree order (i.e., on the number of options for each message) is through the function $k/\ln(k)$ which is reported in Fig. 4; this function represents the *entropic bound* for the service time, normalized to the entropy expressed in the base e and to the announcement duration. Of course, this does not mean that a ternary tree is always a better choice than any other regular k-ary tree, or a non-regular tree. The optimal value of k for a given design procedure will depend on the probability values of the set of services.

Since our service tree design problem has been shown to be analogous to source coding, we can resort to the existing set of entropy-based source coding algorithms for which:

a) an alphabet of k symbols is used;
b) the probability of occurrence of each message (service) is known.

A natural choice in this case is the Huffman coding, which, though being more than 50 years old [16], provides near-optimal performance. In fact, its inefficiency, i.e., the difference between the expected service time and the optimum, given by the entropy, is bounded by [17]

$$\Delta T = \sigma_k + p_M \frac{k}{\ln k} \tag{5}$$

where

$$\sigma_k = \log_k (k-1) + \log_k \left(\log_k (e)\right) - \log_k (e) + \frac{1}{k-1}. \tag{6}$$

For the sake of completeness, we report here the Huffman coding methodology, adapted to our case, through the list of its steps.

1) List the services.
2) Order the services starting with the most likely one.
3) Choose the number k of options for each message.
4) Create a parent node for the least likely services by grouping the k least likely services (if it is the first grouping and $(M-1)/(k-1)$ is not an integer then group under the new node the $M \bmod (k-1)$ least likely services).
5) Assign to the new node a probability being the sum of the probabilities of its sons.
6) Redefine an ordered list formed by the so far ungrouped services and the grouped services as represented by their parent nodes.
7) Go back to step 4 until no ungrouped services are left.

IV. Evolutionary Design of Service Trees

In this paper, an evolutionary algorithm based on a recently proposed tree encoding (Dandelion encoding) is presented

for near-optimal design of services trees in a call center. In the next subsection we present the different parts of the algorithm, including the encoding using the Dandelion code, and a modification of the algorithm to manage rooted trees, two search space size reductions, and the main operators and parameters used in the core of the evolutionary algorithm.

A. Evolutionary Algorithm Encoding: The Dandelion Code

The Dandelion code is a Cayley-like encoding which has been recently described and used for encoding trees in evolutionary algorithms [12], [13]. 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 the 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 $n \geq 2$, 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, C_n are known as *Cayley strings*, or *Cayley codes*. For each $n \geq 2$ let $\top_n$ be the set of labeled trees on the vertex set $[1, n]$. The Cayley result showed that $|\top_n| = n^{n-2}$, and $|C_n| = |\top_n|$. Thus, the trees in $|\top_n|$ 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).

In [12], [13], it is shown that the Dandelion encoding is a suitable way of encoding a tree in an evolutionary algorithm, better than other types of encodings like Prüfer encoding for example, since Dandelion code ensures locality (i.e., small changes in the code, make small changes in the tree), whereas Prüfer encoding or other types of tree representation do not have this important feature, so their performance is usually worse.

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 [18], which has been also used in [13].

1) Input: a Dandelion code $C = (c_2, c_3, \ldots, c_{n-1})$.
2) Output: the tree $T \in \top_n$ corresponding to C.
3) Step 1: define the function ϕ_C: $[2, n-1] \rightarrow [1, n]$ such that $\phi_C(i) = c_i$ for each $i \in [2, n-1]$. Note that the value of this function ϕ_C in i corresponds to the ith position of the encoding (c_i).
4) Step 2: calculate the cycles associated to the function ϕ_C, $Z_1, Z_2, \ldots, Z_L$ (see [12] for an example). 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 $i > j$.
5) Step 3: form a single list π of the elements in Z_1, $Z_2, \ldots, Z_L$, in the order they occur in this cycle list, from the first element of Z_1 to the last element of Z_L.
6) Step 4: construct the tree $T \in \top_n$ corresponding to C in the following way: take a set of n isolated vertices (labeled with the integers from 1 to n), create a path from vertex 1 to vertex n by following the list π from left to right, and then create the edge (i, c_i) for every $i \in [2, n-1]$ which does not occur in the list π.

We will illustrate this fast algorithm using the example in Fig. 5. This figure proposes the Dandelion code

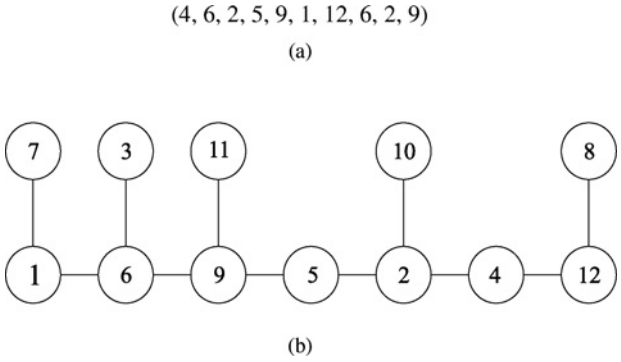

Fig. 5. (a) Example of a Dandelion code. (b) Final tree after the decoding process.

$C = (4, 6, 2, 5, 9, 1, 12, 6, 2, 9)$. Note that there are three cycles in this case, $Z_1 = (6, 9)$, $Z_2 = (5)$, and $Z_3 = (2, 4)$. Note also that the order in which we have recorded these cycles follows the indications in step 2 of the fast decoder algorithm. We form then 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 i which are not in the list π, in this case the vertices 7, 3, 11, 10, and 8.

B. The Oriented-Tree Case

Dandelion code considers the case of un-oriented trees, in which there is not a root node to start the tree. In the design of call centers a rooted tree is required, i.e., one node acts as root node, and the rest are directly attached to it or as subbranches in lower nodes. Thus, a given encoding for a Dandelion code represents $n - M$ possible directed trees (note that M of the tree leaves cannot be considered as root nodes). After the decoding of the Dandelion string, we evaluate the $n-M$ possible different directed trees, and keep the best tree in terms of the objective function as represented by the Dandelion string, (see Fig. 6 as an example).

Another important point to consider in the oriented-tree case is that M nodes of the oriented-trees must be leaves. Thus, a procedure that ensures this fact is necessary. We implement a procedure based on a property of the Dandelion codes: the leaves of a tree in a Dandelion code are the nodes whose numbers do not appear in the Dandelion string. As an example, Fig. 5(a) 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. 5(b)]. In this way, the procedure consists of not using the first M nodes numbers (1 to M) in the random generation of the initial codes (at the beginning of the evolutionary algorithm), nor in the mutation operator. Note that this implies that elements in the Dandelion codes used in the proposed evolutionary algorithm (EA) are in the range $[M + 1, n - 2]$.

In our evolutionary algorithm, we use a fixed-length tree encoding, i.e., trees with a given constant number of nodes equal to $n = 2M - 1$. Note that $2M - 1$ is the worst case, that is, any tree with larger number of nodes than $2M - 1$ can be transformed into a better fitness tree with less than

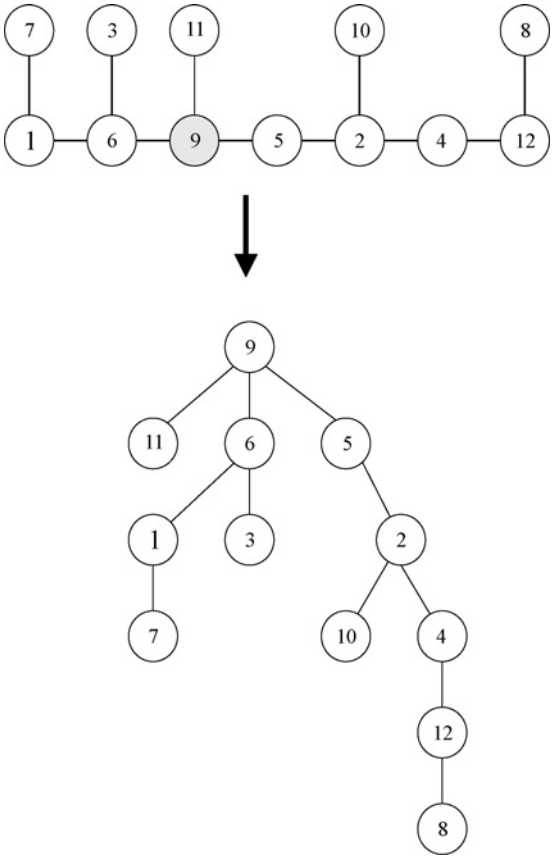

Fig. 6. Example of root selection in a directed tree; it is supposed that the best fitness is obtained when the node number 9 is considered the tree root.

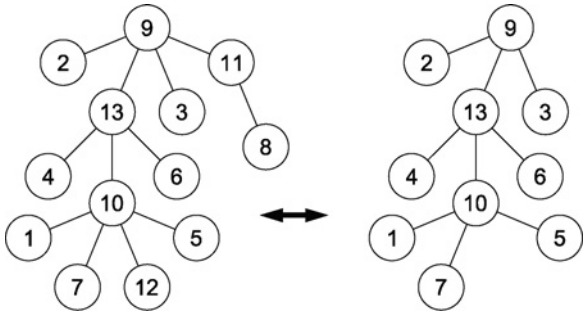

Fig. 7. Example of leaves' removal in an unfeasible tree.

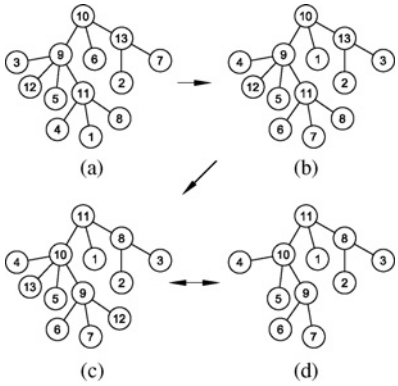

Fig. 8. Search space reductions example. (a) Original tree. (b) Equivalent tree after applying Reduction 1. (c) Equivalent tree after applying Reduction 2. (d) Final tree, after removing all useless leaves.

or equal to $2M - 1$. Moreover, these constant-length codes allow us to perform a search for trees of any number of nodes. That is because a tree which has some leaves with associated numbers larger than M (and consequently a tree with more than M leaves) is equivalent to another tree without these leaves (both trees produce the same value of the fitness function, because we consider that only the M first leaves of the tree are relevant to calculate the average number of bifurcations). See an example of this in the tree of Fig. 7. In this example $M = 7$, thus the nodes 8, 11, and 12 (all of them larger than M) which appear in the tree on the left-hand side do not influence the number of bifurcations, so the tree on the right-hand side is equivalent.

C. Search Space Size Reductions

Two search space size reductions can be considered when the Dandelion code is applied to obtain the optimal service tree. Both reductions are related to the position of the tree leaves and the intermediate nodes. The objective of these search space reductions is to eliminate redundant trees in the search space. Note that these search space reductions do not eliminate the optimal tree from the search space.

1) *Search Space Reduction 1:* There is one quite intuitive space size reduction that may be applied in order to save computation time in the algorithm. Recall that in the correspondence between codewords and service trees that we employ in the use of the Huffman algorithm, each codeword represents a service in the call center, and the length of each codeword is related to the time a given user needs to reach that service, or, equivalently, the number of bifurcations from the root node to the leaf (service). The idea is that the most frequently requested services must be associated to the smallest times (smallest number of bifurcations), as the Huffman algorithm does. In order to force this, we sort the services in increasing order of probability, and the positions where these services (leaves) can be placed in decreasing order of time to reach them, and then match the elements of both list in an one-to-one fashion. Note that this procedure always provides a solution at least as good as the initial one, so the resulting search space will contain the global optimum to the problem. Fig. 8(a) and (b) shows an example of this search space reduction procedure. This example considers a service tree with $M = 7$ services, with the service 1 being the most probable, the service 2 the second most probable, and so on.

2) *Search Space Reduction 2:* The second search space size reduction deals with the reallocation of the $n - M$ last nodes. Recall that the first M numbers in nodes identify the tree services, but there are still $n - M$ nodes in the tree which do not represent services, and have influence in the encoding of the tree. The idea is that these $n - M$ nodes must be reallocated as a function of the M leaves of the tree, in such a way that two trees with the leaves in the same position (same fitness), have the same encoding. Note that this procedure reduces $(n - M)!$ trees, all them equivalent (same fitness), into just one, and therefore part of the existing encoding redundancy is eliminated. Fig. 8(b) and (c) shows an example of this point.

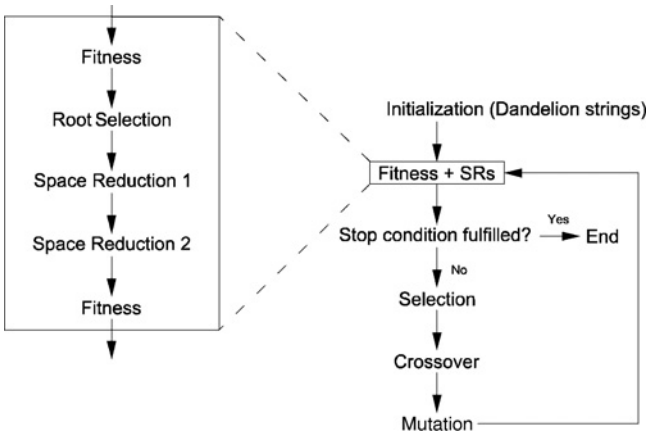

Fig. 9. Outline of the evolutionary algorithm proposed in this paper, including the search space reduction (SR) procedures.

Fig. 8(d) shows a tree equivalent to the one given in (c), a process similar to the one shown in Fig. 7.

D. Evolutionary Algorithm Operators and Parameters

Our evolutionary algorithm is structured in the traditional form, with procedures of selection, crossover, and mutation. The encoding of the proposed EA is the Dandelion encoding described in Section IV-A, specific for the direct tree case, as described in Section IV-B. Once the root is selected, the two local searches for reducing the search space are applied. At this stage, the individual is ready for the calculation of its final fitness value. Several operators of selection and crossover are used in the experiments carried out, whereas the mutation operator is maintained unchanged. We describe the election of selection and crossover in each experimental subsection. The mutation operator is a little bit different than in a classical evolutionary algorithm. Recall that we reserve the first M 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 from $[M+1, n-2]$ (range of the elements of the Dandelion codes). An outline of the proposed EA is shown in Fig. 9.

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 100 and the generations number has been fixed to 750, after which we keep the best tree encountered so far.

V. Experiments and Results

In order to test the performance of the algorithms proposed in this paper we have tackled the design of a number of call centers, structuring the experiments in synthetic instances and a real application.

In the case of synthetic instances we test different aspects of the algorithm using different number and probability distribution of services. Specifically, we have solved synthetic instances from 50 to 300 services, in steps of 50. In these instances we test the differences in performance of the algorithm when the search space reduction is applied, we analyze the computational time required and several possibilities to reduce it, and finally the performance of the algorithm when different evolutionary operators are applied. To offer a direct comparison of the results obtained in the different experiments, we used the Huffman algorithm's results and the entropic bound described in Section III (see Table I). Regarding the real application, it corresponds to the design of a call center to the Italian mobile communications company TIM. In all the experiments carried out in this paper a number of 30 independent runs for each instance are considered.

TABLE I
Lower Bound and Huffman Algorithm's Results (in Seconds) in the Different Synthetic Instances Tackled in the Paper

Number of services	50	100	150	200	250	300
Zipf distribution						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781
Uniform distribution						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821

Used for reference and comparison in the rest of the paper.

In the experiments carried out on synthetic problems instances we use two models for the probability of services: a classical uniform probability distribution and a Zipf probability distribution. According to the latter model, the probability of a given service is related to its popularity rank. Assigning ranks in order of decreasing popularity the probability that the ith ranked service in a group of M is chosen is inversely proportional to its rank, i.e.,

$$P[X = i] = \frac{\frac{1}{i}}{\sum_{i=1}^{M} \frac{1}{i}} \tag{7}$$

where X is the random variable expressing the choice. This expression represents the simplest form of the original formulation by Zipf [21] where the probability of interest is inversely proportional to a power of the rank (though the power was close to unity in the original model). Though arising in the field of linguistics, to model the frequency of appearance of words in a written text, the model has quickly established itself to describe rank-frequency relationships in a number of application fields. Unfortunately, there are no studies, as far as the authors know, on popularity models for the services offered by a call center, but the Zipf model has been widely used in similar and related contexts. Among the most relevant we can cite the popularity of web documents [22], information dissemination [23], queries in P2P systems [24], web services on personal digital assistants (PDAs) [25], and wide-area infrastructure services [26]. We can therefore consider the Zipf model as a valid candidate to represent the popularity of call center services.

A. Experiments to Test the Proposed Search Space Reductions

In the first round of experiments, we show the effect of introducing the search space reductions described in

TABLE II

COMPARISON OF THE RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA (MULTI-POINT CROSSOVER, ROULETTE WHEEL) WITH AND WITHOUT SEARCH SPACE REDUCTIONS, WHEN SERVICE PROBABILITIES FOLLOW A ZIPF DISTRIBUTION

Number of services	50	100	150	200	250	300
Dandelion EA (NSPR)						
Best value	8.9034	10.3261	11.1720	11.7129	12.3673	12.9145
Mean	8.9830	10.4681	11.3710	12.0588	12.8042	13.2107
Standard deviation	0.0523	0.0801	0.0862	0.1521	0.1619	0.1875
Dandelion EA (SPR1)						
Best value	8.8534	10.1535	10.9125	11.4586	11.8786	12.2175
Mean	8.8678	10.1734	10.9388	11.4895	11.9047	12.2492
Standard deviation	0.0119	0.0270	0.0204	0.0198	0.0200	0.0234
Dandelion EA (SPR2)						
Best value	8.8862	10.2806	11.0708	11.6603	12.2267	12.7219
Mean	8.9423	10.3623	11.2684	11.8892	12.5441	12.8941
Standard deviation	0.0385	0.0615	0.0694	0.1077	0.1149	0.1493
Dandelion EA (SPR)						
Best value	8.8532	10.1498	10.9000	11.4312	11.8441	12.1791
Mean	8.8639	10.1558	10.9106	11.4439	11.8592	12.1953
Standard deviation	0.0083	0.0066	0.0114	0.0075	0.0118	0.0118
LB and Huffman value						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781

NSPR: EA without any search space reduction; SPR1: EA working only with the first space reduction; SPR2: EA working only with the second space reduction; SPR: EA working with both space reductions proposed.

TABLE III

COMPARISON OF THE RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA (MULTI-POINT CROSSOVER, ROULETTE WHEEL) WITH AND WITHOUT SEARCH SPACE REDUCTION, WHEN SERVICE PROBABILITIES FOLLOW A UNIFORM DISTRIBUTION

Number of services	50	100	150	200	250	300
Dandelion EA (NSPR)						
Best value	10.5558	12.6693	13.9872	14.5584	15.3126	16.0303
Mean	10.7694	12.9346	14.0071	14.8908	15.6012	16.3907
Standard deviation	0.0799	0.2356	0.1300	0.1200	0.1472	0.2292
Dandelion EA (SPR1)						
Best value	10.2901	12.0899	13.2636	14.1509	14.6537	15.2246
Mean	10.2975	12.1088	13.3093	14.2337	14.7668	15.3489
Standard deviation	0.0081	0.0194	0.0518	0.0619	0.0735	0.0953
Dandelion EA (SPR2)						
Best value	10.4606	12.4422	13.7476	14.4740	15.0689	15.7187
Mean	10.6110	12.6443	13.7638	14.7078	15.3249	16.1467
Standard deviation	0.0554	0.1910	0.1100	0.1078	0.1310	0.1804
Dandelion EA (SPR)						
Best value	10.2782	12.0884	13.2566	14.1330	14.6099	15.1614
Mean	10.2812	12.0930	13.2580	14.1453	14.6356	15.1804
Standard deviation	0.0028	0.0089	0.0026	0.0226	0.0253	0.0110
LB and Huffman value						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821

NSPR: EA without any search space reduction; SPR1: EA working only with the first space reduction; SPR2: EA working only with the second space reduction; SPR: EA working with both space reductions proposed.

Section IV-C. For these experiments we consider the proposed EA with a multi-point crossover operator and roulette wheel selection, the rest of parameters of the EA are the ones described in Section IV-D.

Table II shows the results obtained by our approach with search space size reductions: NSPR stands for the case of the EA working without any search space reduction, SPR1 stands for the first search space reduction, SPR2 stands for the second search space reduction, and SPR refers to the application of the two search space reductions. This table shows the results for the Zipf probability distribution of services. Table III shows similar results for the case of uniform distribution of services. Note that in all cases considered the proposed EA works much better when the two SPR are applied. The EA working with SPR1 shows better performance than the EA working with SPR2. Regarding the comparison with the Huffman algorithm (reference values also provided in Tables II and III), note that the EA with SPR obtains better solutions in terms of the objective function for both cases (Zipf and uniform distribution of services), however, the EA without SPR is not able to obtain better results than the Huffman approach. It is interesting that the EA with the

TABLE IV
RESULTS (IN SECONDS) OBTAINED BY THE PROPOSED EVOLUTIONARY ALGORITHM WITH DIFFERENT PERCENTAGE OF NODES TESTED AS ROOT OF THE TREE (SERVICES FOLLOWING A ZIPF PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Dandelion EA (best value)						
1%	8.8853	10.3204	11.2163	11.8283	12.4154	12.8592
25%	8.8793	10.2565	10.9669	11.6173	12.0339	12.5029
50%	8.8565	10.1574	10.9023	11.4511	11.8663	12.1999
75%	8.8544	10.1498	10.9043	11.4312	11.8560	12.1879
100%	8.8532	10.1493	10.9000	11.4280	11.8441	12.1791
Dandelion EA (mean value)						
1%	8.9917	10.5850	11.6305	12.2701	13.0976	13.5997
25%	8.9592	10.4250	11.3734	12.0437	12.5653	13.0832
50%	8.8996	10.2487	11.0504	11.6845	12.2973	12.7432
75%	8.8764	10.1806	10.9481	11.5633	12.1477	12.3909
100%	8.8639	10.1558	10.9106	11.4439	11.8592	12.1953
Dandelion EA (standard deviation)						
1%	0.0871	0.1780	0.2506	0.2525	0.3685	0.4277
25%	0.0788	0.1388	0.2509	0.2611	0.3043	0.3651
50%	0.0492	0.1264	0.1934	0.2163	0.3571	0.4135
75%	0.0432	0.0456	0.0983	0.2422	0.3721	0.3361
100%	0.0028	0.0089	0.0026	0.0226	0.0253	0.0110
LB and Huffman value						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781

SPR1 is able to improve the results of the Huffman algorithm, but the EA with the SPR2 is only able to obtain better results than the Huffman approach for the first (smallest) instances. The improvement of performance of the EA with SPR (both reductions of the search space included) is evident in these experiments. On the other hand, note how close the results obtained by the EA with SPR are to the entropic bound.

As conclusion for these experiments, we stress the good performance of the algorithm when the two proposed procedures for reducing the search space size are applied. The reason for this good performance is, of course, that the algorithm needs to look for the optimal tree in a smaller search space. In order to measure this reduction, we need to estimate the number of trees remaining after applying the search space reductions in a problem with M services. First, note that if the reduction is not applied, the number of trees of n nodes is given in [12] as n^{n-2}. Recall that $n = 2M - 1$ and also that the first M elements are reserved for the tree services, so the total number of trees (search space size) is in this case $(M-1)^{2M-3}$.

In the case that the first search space reduction is applied, there are $M!$ ways of sorting M services, among which we select the best one. Therefore, the search space size in this case yields

$$\frac{(M-1)^{2M-3}}{M!}. \tag{8}$$

In the case that the second search space reduction is applied, there are $(M-1)!$ ways of enumerating those nodes which are not services, among which we select one. Therefore, the search space size in this case yields

$$\frac{(M-1)^{2M-3}}{(M-1)!}. \tag{9}$$

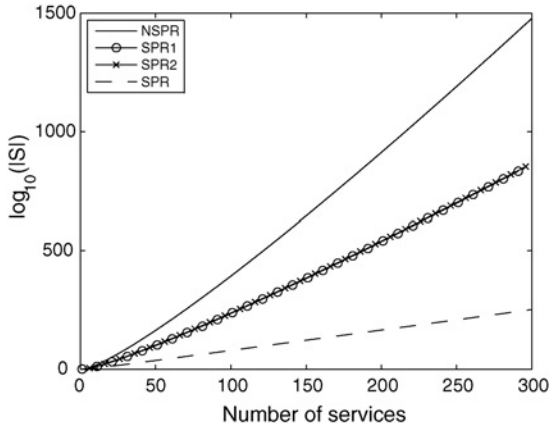

Fig. 10. Search space size (logarithmic scale) with and without search space size reductions.

Also, finally, if the two proposed search space size reductions are applied, the minimum possible number of trees yields

$$\frac{(M-1)^{2M-3}}{M!(M-1)!}. \tag{10}$$

Fig. 10 shows the number of trees (search space size in logarithmic scale) versus the number of services in the tree, with the different search space size reductions and without reduction. Note that the reduction of the search space introduced by SPR1 and SPR2 is similar (they only differ in a scale factor M). Note also the very large reduction of search space size obtained, including the two proposed search space size reductions for large values of M (about 200 to 300 leaves).

B. A Note on the Computational Cost of the Algorithm

Regarding the computational cost of the algorithm, it depends much on the problem instance. For the smallest

TABLE V
RESULTS (IN SECONDS) OBTAINED BY THE PROPOSED EVOLUTIONARY ALGORITHM WITH DIFFERENT PERCENTAGE OF NODES TESTED AS ROOT OF THE TREE (SERVICES FOLLOWING A UNIFORM PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Dandelion EA (best value)						
1%	10.2841	12.1943	13.8686	14.8038	15.7507	16.8447
25%	10.2822	12.1784	13.4259	14.6299	15.1360	16.0932
50%	10.2782	12.1133	13.2822	14.2017	14.7140	15.4591
75%	10.2782	12.0892	13.2566	14.1370	14.6234	15.1893
100%	10.2782	12.0884	13.2566	14.1330	14.6099	15.1614
Dandelion EA (mean value)						
1%	10.4085	12.7351	14.5776	16.4836	17.3186	18.5386
25%	10.3618	12.5186	14.1260	15.3152	16.3300	17.3227
50%	10.3361	12.3411	13.7213	15.0236	16.0435	16.7326
75%	10.2997	12.1940	13.4325	14.6008	15.3155	15.8346
100%	10.2812	12.0930	13.2580	14.1453	14.6356	15.1804
Dandelion EA (standard deviation)						
1%	0.0719	0.4650	0.6502	1.0706	0.9903	1.2525
25%	0.1124	0.2167	0.3668	0.5367	0.7323	0.8141
50%	0.1037	0.1826	0.3521	0.3982	0.6746	0.8508
75%	0.0651	0.1128	0.2422	0.4759	0.4576	0.6719
100%	0.0028	0.0088	0.0026	0.0225	0.0253	0.0110
LB and Huffman value						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821

TABLE VI
RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA USING DIFFERENT EVOLUTIONARY OPERATORS (SERVICES FOLLOWING A ZIPF PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Dandelion EA (best value)						
Multi-point+Roulette wheel	8.8532	10.1498	10.9000	11.4312	11.8441	12.1791
Multi-point+tournament	8.8543	10.1496	10.8945	11.4247	11.8347	12.1705
Two-point+Roulette wheel	8.8525	10.1508	10.8995	11.4336	11.8448	12.1800
Two-point+tournament	8.8491	10.1483	10.8928	11.4230	11.8366	12.1672
Dandelion EA (mean value)						
Multi-point+Roulette wheel	8.8639	10.1558	10.9107	11.4439	11.8593	12.1953
Multi-point+tournament	8.8621	10.1627	10.9059	11.4334	11.8423	12.1745
Two-point+Roulette wheel	8.8623	10.1632	10.9153	11.4529	11.8626	12.1939
Two-point+tournament	8.8565	10.1512	10.8982	11.4308	11.8436	12.1719
Dandelion EA (standard deviation)						
Multi-point+Roulette wheel	0.0083	0.0066	0.0114	0.0075	0.0118	0.0118
Multi-point+tournament	0.0080	0.0234	0.0119	0.0085	0.0048	0.0037
Two-point+Roulette wheel	0.0082	0.0117	0.0120	0.0103	0.0111	0.0108
Two-point+tournament	0.0094	0.0039	0.0064	0.0069	0.0047	0.0055
LB and Huffman value						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781

instance tackled, the one with 50 services, the real computation time of the complete evolutionary algorithm (including a complete search space size reduction), in a computer with an INTEL Core II processor was about 5 s. For the largest instance with 300 services the computation time was about 300 s. Note that this is the case when all the nodes are tested as root nodes (as explained in Section IV-B). The question is if we can control the algorithm's computation time by reducing the number of nodes tested as root nodes, and, of course, how this affects the algorithm performance. To answer these questions, we have carried out a set of experiments on the synthetic problems defined above, with different numbers of services, Zipf and uniform service distributions and different numbers of nodes tested as a root node in the algorithm. Tables IV and V show the results of these experiments, and Fig. 11 shows the computation time for various cases, including the computation time of the Huffman algorithm. Note that, as expected, the best performance of the EA is obtained when all the nodes are tested to be the root. Fig. 12 shows the best and mean values obtained with the proposed EA, and the Huffman value, on average from all the experiments carried out (from 50 to 300 nodes, Zipf and uniform distributions), for different numbers of nodes tested as root of the tree. From this figure it is possible to see that we need to test about 40% of the roots to beat the Huffman algorithm, and 85% to beat Huffman on average. Regarding the computation

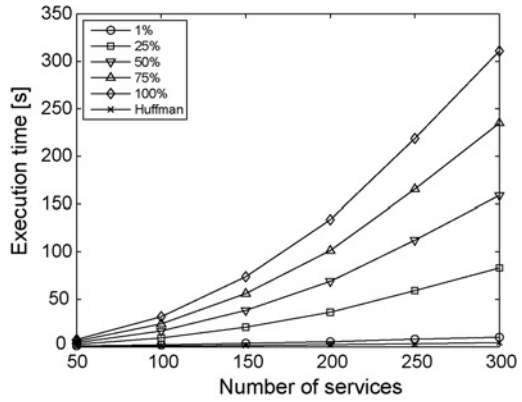

Fig. 11. Total computation time of the EA with different number of nodes tested as root of the tree and computation time of the Huffman algorithm.

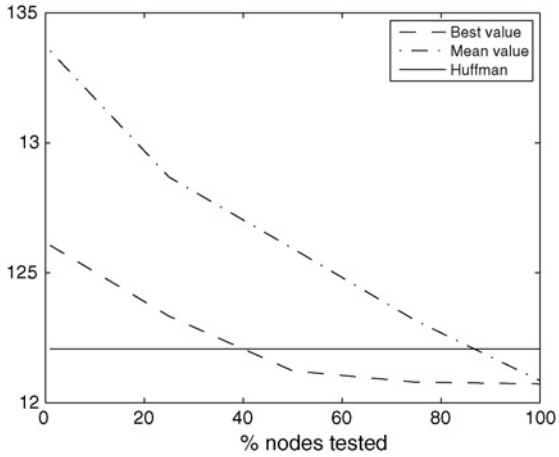

Fig. 12. Best and mean values obtained with the proposed EA and Huffman value, on average from all the experiments carried out (from 50 to 300 nodes, Zipf and uniform distributions), for different number of nodes tested as root of the tree.

time, it is straightforward that the time complexity reduction is proportional to the reduction in the number of tested roots, i.e., testing 50% of the roots reduces the computation time by a factor of 2. This reduction of the computation time is only worth if the Huffman algorithm's performance is improved (in best value), which is obtained when over 40% of nodes are tested as root nodes. In this case, the improvement over Huffman is worth the computational cost.

C. Experiments With Various Types of Selection and Crossover Operators

Another important aspect of this research is to test the performance of the proposed algorithm using different operators in the evolutionary search. To do this, we use the same set of instances as in the previous sections, and we run the proposed evolutionary algorithm with search space reduction, using two-point crossover, multi-point crossover, roulette wheel selection and tournament selection, and compare the performance of the four possible combinations of these operators (all nodes tested as possible roots). Tables VI and VII show the results obtained with the different combinations of operators in the instances with a Zipf and a uniform probability distribution of services, respectively. The results show that the proposed EA performs slightly better using a combination of two-point crossover and tournament selection for the Zipf probability distribution of services, than with other combinations of operators. In the case of uniform distribution of services, multi-point crossover with tournament selection also provides very good performance, specially in problems with a large number of services. A comparison of these results with the Huffman and lower bound given in Table I shows that in all cases the EA improves the results of the Huffman approach, and it is quite close to the entropic bound. Anyway, note that the results obtained by the proposed EA with different operators are quite similar, in many cases the differences between different configurations of the algorithm are not significant. This means that the proposed approach performs well independently of the chosen set of crossover and mutation operators.

D. Experiments Involving the Huffman Algorithm

To finish the experimental part of the paper based on synthetic instances, we carry out a study of the possibility of using the Huffman algorithm to improve the performance of the EA. Specifically, we analyze the possibility of initializing the population of the EA with mutated versions of the Huffman solution. Tables VIII and IX show the results of a EA with the best combination possible of selection and crossover operators, for different percentages of mutation of the initial Huffman solution, for the case of Zipf and uniform distributions of services, respectively. First, note that this initialization slightly improves the performance of the EA in the case of considering a Zipf distribution of services. This can be seen by comparing Table VIII with the corresponding Table VI, where the performance of the EA with different operators is shown. This improvement is more significant in the largest instances, with 200, 250, and 300 services. The performance of the EA with Huffman initialization in the case of a uniform distribution of services is quite similar to the EA with random initialization.

A final comparison for showing the good performance of the Dandelion EA proposed in this paper can be carried out against a Monte Carlo simulation starting from the Huffman solution. Specifically, the Monte Carlo simulation is run from the initial Huffman solution which is randomly mutated (5% of the elements in the corresponding Dandelion code are changes), and if a better solution is found, it is considered as the current solution to apply the next mutation. This process is carried out for a number of objective function evaluations similar to the one in the proposed EA, and 30 different runs of the Monte Carlo simulation have been launched, to analyze the mean and standard deviation values, such as in the EA case. The results obtained with the Monte Carlo simulation are shown in Table X, where results for the Zipf and uniform services distributions are jointly displayed. Note that the Monte Carlo simulation works fine, obtaining solutions of good quality for the problem. However, the proposed EA performs better, as can be seen in the comparison of Table X, both for the Zipf and uniform services distributions.

E. Design of a Real Call Center With the Proposed Approach

To illustrate the performance of our approach in a real case, we tackle the design of a call center for the Italian mobile

TABLE VII
RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA USING DIFFERENT EVOLUTIONARY OPERATORS (SERVICES FOLLOWING A UNIFORM PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Dandelion EA (best value)						
Multi-point+Roulette wheel	10.2782	12.0884	13.2566	14.1330	14.6099	15.1614
Multi-point+tournament	10.2782	12.0882	13.2559	14.1313	14.6056	15.1412
Two-point+Roulette wheel	10.2782	12.0882	13.2573	14.1348	14.6089	15.1468
Two-point+tournament	10.2782	12.0881	13.2559	14.1318	14.6058	15.1413
Dandelion EA (mean value)						
Multi-point+Roulette wheel	10.2812	12.0930	13.2580	14.1454	14.6356	15.1805
Multi-point+tournament	10.2824	12.0901	13.2573	14.1336	14.6073	15.1455
Two-point+Roulette wheel	10.2832	12.0960	13.2740	14.1418	14.6259	15.1660
Two-point+tournament	10.2798	12.0894	13.2569	14.1339	14.6102	15.1526
Dandelion EA (standard deviation)						
Multi-point+Roulette wheel	0.0028	0.0089	0.0026	0.0226	0.0254	0.0110
Multi-point+tournament	0.0035	0.0015	0.0013	0.0025	0.0039	0.0050
Two-point+Roulette wheel	0.0052	0.0104	0.0401	0.0063	0.0089	0.0106
Two-point+tournament	0.0019	0.0031	0.0014	0.0037	0.0041	0.0081
LB and Huffman value						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821

TABLE VIII
RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA WITH MUTATIONS OF THE HUFFMAN TREE USED TO FORM THE INITIAL POPULATION (SERVICES FOLLOWING A ZIPF PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Best value						
12.5%	8.8491	10.1483	10.8924	11.4230	11.8339	12.1643
25%	8.8491	10.1483	10.8921	11.4230	11.8343	12.1657
37.5%	8.8491	10.1483	10.8921	11.4230	11.8343	12.1654
50%	8.8491	10.1483	10.8919	11.4233	11.8348	12.1644
Mean						
12.5%	8.8532	10.1514	10.8961	11.4270	11.8398	12.1685
25%	8.8519	10.1558	10.9023	11.4311	11.8425	12.2015
37.5%	8.8519	10.1558	10.9023	11.4311	11.8425	12.1963
50%	8.8556	10.1529	10.9026	11.4351	11.8418	12.2106
Standard deviation						
12.5%	0.0058	0.0062	0.0063	0.0041	0.0055	0.0043
25%	0.0037	0.0114	0.0097	0.0074	0.0048	0.1620
37.5%	0.0037	0.0114	0.0097	0.0074	0.0048	0.1379
50%	0.0084	0.0087	0.0095	0.0084	0.0045	0.2146
LB and Huffman value						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781

operator TIM. TIM is now the mobile brand of the Italian operator *Telecom Italia*, while formerly it was a separate company (though owned by Telecom Italia itself). At present it can boast a share of 39.4% of the Italian mobile market, with 36 millions mobile lines. In this paper, we examine the redesign of the customer care service call center, offered by TIM through its "green" number 119. The IVR (service tree) used by TIM is formed by 50 different services, structured as shown in Fig. 13. Table XI shows the probabilities associated to each service, obtained from the company. This initial service tree is representative of a call center managing a very large number of customers each day, with very different range of services. The entropic bound for this real system is 179.3241 s. Fig. 14 shows the best tree obtained for the TIM call center with the proposed approach, with an average time of users in the system (objective function value) of 181.0245. On the other hand, Fig. 15 shows the tree obtained with the Huffman algorithm, with an average time of 185.9082, almost 6 s worse than the best tree obtained with our approach.

In this real example, we also test the possibility of including constraints on grouping the nodes. A company owning the call center may wish to group certain services to better structure the call center. In this case, our algorithm is still applicable: let us suppose that we want to group services s_1, s_2, and s_3, with probabilities p_1, p_2, and p_3, respectively. The idea is to form an *auxiliary node* which represents s_1, s_2, and s_3, with a probability $p_a = \sum_{i=1}^{3} p_i$. Then the EA is run using this auxiliary service, and a tree T^* is obtained. The auxiliary node a is then expanded to represent again nodes s_1, s_2, and s_3, so the tree is now the complete one (T) with the objective function given in this case by $f(T) = f(T^*) + 3 \cdot \sum_{i=1}^{3} p_i$. Note that the 3 in the fitness function refers to the fact that

TABLE IX

RESULTS (IN SECONDS) OBTAINED BY THE DANDELION EA WITH MUTATIONS OF THE HUFFMAN TREE USED TO FORM THE INITIAL POPULATION (SERVICES FOLLOWING A UNIFORM PROBABILITY DISTRIBUTION)

Number of services	50	100	150	200	250	300
Best value						
12.5%	10.2782	12.0881	13.2556	14.1314	14.6063	15.1414
25%	10.2782	12.0881	13.2557	14.1313	14.6065	15.1413
37.5%	10.2782	12.0881	13.2556	14.1314	14.6063	15.1412
50%	10.2782	12.0881	13.2557	14.1313	14.6056	15.1417
Mean						
12.5%	10.2833	12.0913	13.2565	14.1338	14.6131	15.1596
25%	10.2805	12.0904	13.2567	14.1327	14.6117	15.1485
37.5%	10.2833	12.0913	13.2565	14.1338	14.6131	15.1596
50%	10.2807	12.0901	13.3022	14.1325	14.8212	15.1527
Standard deviation						
12.5%	0.0047	0.0082	0.0009	0.0038	0.0127	0.0264
25%	0.0029	0.0054	0.0012	0.0017	0.0049	0.0065
37.5%	0.0047	0.0082	0.0009	0.0038	0.0127	0.0420
50%	0.0032	0.0047	0.2518	0.0016	0.6536	0.0080
LB and Huffman value						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821

TABLE X

RESULTS (IN SECONDS) OBTAINED USING A MONTE CARLO METHOD STARTING FROM THE HUFFMAN TREE (SERVICES FOLLOWING ZIPF AND UNIFORM DISTRIBUTIONS)

Number of services	50	100	150	200	250	300
Monte Carlo from Huffman (Zipf)						
Best value	8.8517	10.1511	10.8961	11.4286	11.8374	12.1723
Mean	8.8736	10.2238	10.9261	11.5012	11.9015	12.2369
Standard deviation	0.0171	0.0556	0.0439	0.0550	0.0454	0.0390
LB, Huffman, and EA results						
Lower bound	8.7304	10.0512	10.8036	11.3297	11.7336	12.0611
Huffman	8.9673	10.2964	11.0318	11.5581	11.9569	12.2781
EA best	8.8491	10.1483	10.8924	11.4230	11.8339	12.1643
Monte Carlo from Huffman (uniform)						
Best value	10.2878	12.0938	13.2676	14.1374	14.6354	15.1606
Mean	10.3657	12.1684	13.3485	14.1970	14.6767	15.2258
Standard deviation	0.0583	0.0407	0.0640	0.0382	0.0141	0.0440
LB, Huffman, and EA results						
Lower bound	10.2018	11.9975	13.1789	14.0203	14.4959	15.0581
Huffman	10.5058	12.2078	13.4689	14.2535	14.6869	15.2821
EA best	10.2782	12.0881	13.2556	14.1313	14.6056	15.1412

The entropic lower bound, the initial Huffman value, and the best value obtained with the proposed EA are also shown for comparison.

TABLE XI

SERVICE PROBABILITIES FOR THE TIM CALL CENTER

Node	Prob. (%)	Node	Prob. (%)	Node	Prob. (%)	Node	Prob. (%)	Node	Prob. (%)
1	11.8998	11	0.0813	21	2.7703	31	3.0268	41	2.9683
2	1.9627	12	0.2413	22	1.6476	32	0.8785	42	1.9085
3	1.0670	13	1.8517	23	0.8600	33	2.4605	43	1.6544
4	1.3318	14	0.9814	24	0.7593	34	1.9374	44	0.9889
5	2.4261	15	5.4547	25	4.1258	35	1.0288	45	5.0186
6	0.9723	16	0.9598	26	0.2508	36	4.1902	46	2.6290
7	2.7217	17	0.1676	27	1.8098	37	0.2462	47	1.2070
8	1.9713	18	2.8786	28	1.7442	38	0.0958	48	3.0028
9	0.7559	19	0.1113	29	0.0004	39	0.2514	49	0.0453
10	0.3198	20	6.4028	30	3.0777	40	0.8317	50	4.0256

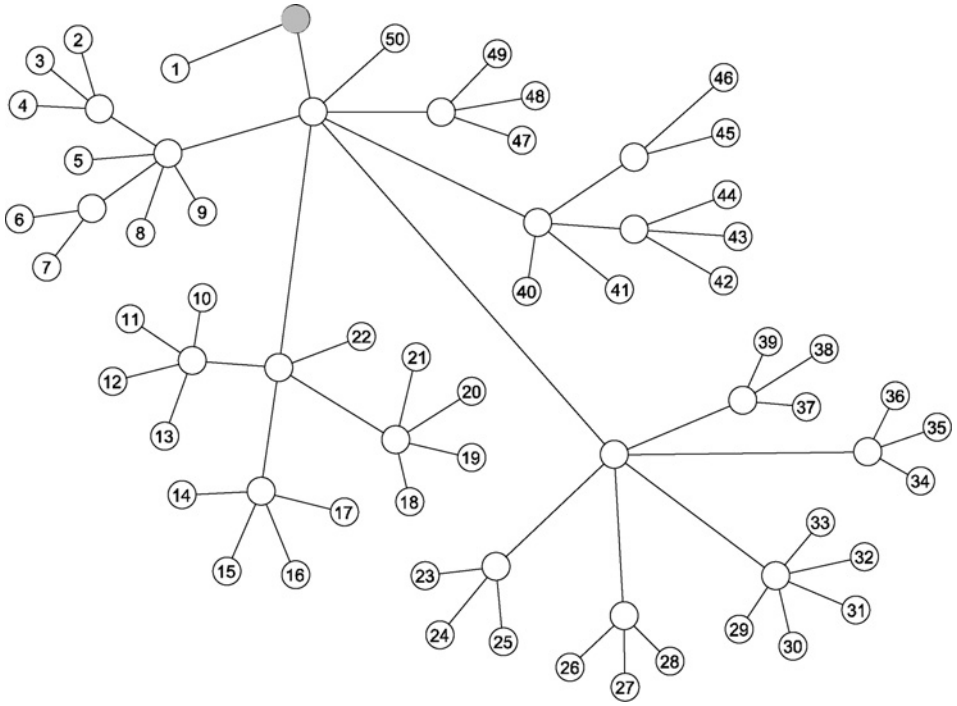

Fig. 13. Initial service tree used by TIM (not optimized).

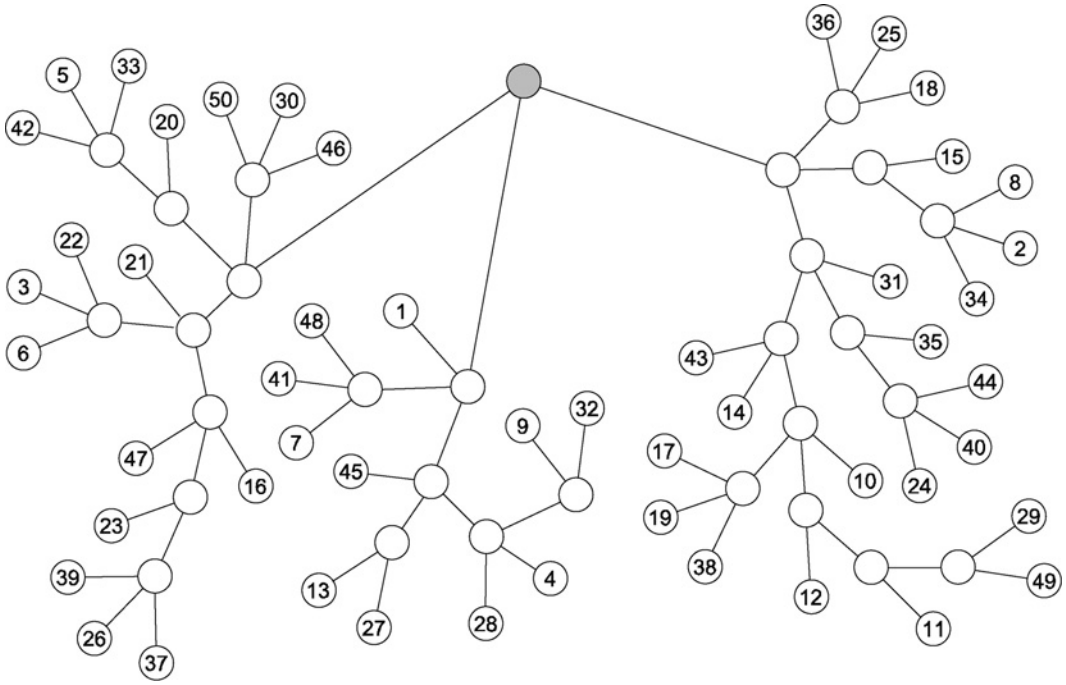

Fig. 14. Optimized tree with the Dandelion EA for the TIM call center.

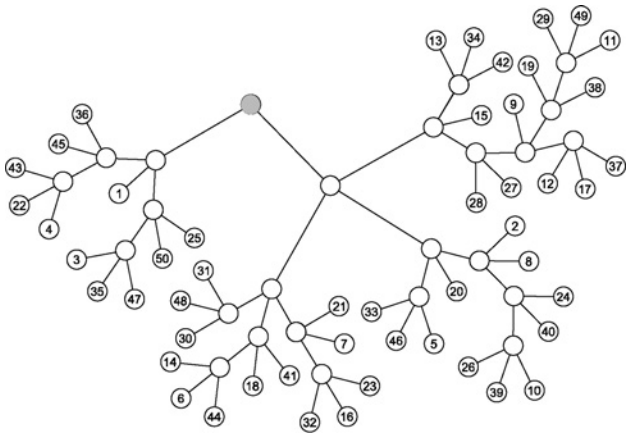

Fig. 15. Tree obtained using the Huffman algorithm, for the TIM call center.

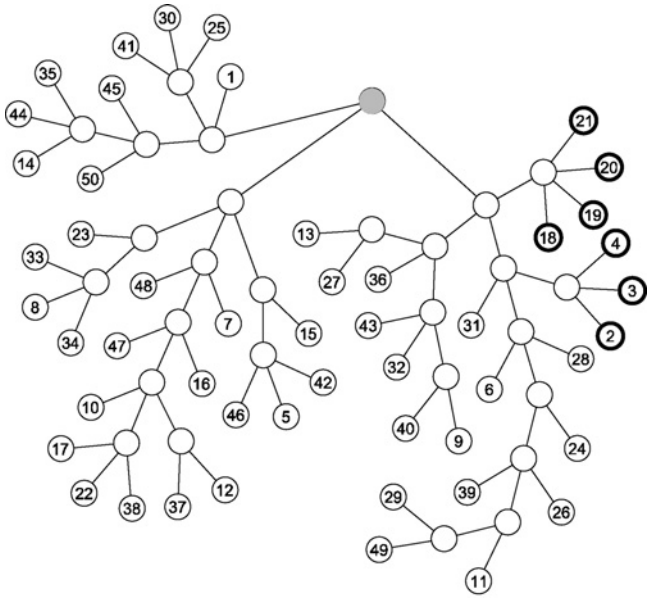

Fig. 16. Tree obtained using the EA in the case of including two groups of services constraint.

we are dealing with grouping three nodes. In order to show an example of this procedure, let us consider the design of the TIM call center with two groups of services, on one hand s_2, s_3, and s_4 and on the other hand s_{18}, s_{19}, s_{20}, and s_{21}. Fig. 16 shows the final service tree obtained with the EA and services groups constraint (groups of services in boldface). In this case, the average time of this tree is 192.509, whereas the Huffman algorithm obtains a tree with an average time of 195.8301.

VI. Conclusion

This paper has proposed an evolutionary algorithm for designing interactive voice response systems within call center systems in an optimal way. The problem is equivalent to the design of an oriented tree with the minimum average service time. The proposed evolutionary algorithm uses an encoding based on the Dandelion code, recently proposed as a good technique for tree encoding in heuristic algorithms. Several modifications of the Dandelion encoding have been applied in order to adapt it to the specific problem tackled and to improve the performance of the algorithm. Specifically, two search space size reduction procedures have been presented which improves the quality of the solutions found by the evolutionary algorithm. Systematic experiments have been carried out in order to test the performance of our approach, including a real case of the design of a call center for an Italian mobile telecommunications company. The results obtained showed that our approach is able to obtain very good services trees improving the results of a Huffman approach and very close to a lower bound (entropic bound) for the problem. As future work, we can consider the recursive use of the algorithm for the case of grouping services in such a way that tree substructures can be considered for the nodes which must be grouped. Also, the joint optimization of the IVR part and human part of the call centers can be accomplished with certain modifications in the algorithm.

References

[1] H. G. Durbin, "International customer relations management from Istanbul by electronic medium," Ph.D. dissertation, Eur. School Bus., Reutlingen Univ., Reutlingen, Germany, 2005.

[2] A. Gilmore and L. Moreland, "Call centers: How can service quality be managed?," *Irish Marketing Rev.*, vol. 13, no. 1, pp. 3–11, 2000.

[3] A. Feinberg, I. S. Kim, L. Hokama, K. de Ruyter, and C. Keen, "Operational determinants of caller satisfaction in the call center," *Int. J. Service Ind. Manage.*, vol. 11, no. 2, pp. 131–141, 2000.

[4] G. Koole and A. Mandelbaum, "Queueing models of call centers: An introduction," *Ann. Operations Res.*, vol. 113, nos. 1–4, pp. 41–59, 2002.

[5] G. Tom, M. Burns, and Y. Zeng, "Your life on hold: The effect of telephone waiting time on customer perception," *J. Direct Marketing*, vol. 11, no. 3, pp. 25–31, 1997.

[6] J. Anton, "The past, present, and future of customer access centers," *Int. J. Service Ind. Manage.*, vol. 11, no. 2, pp. 120–130, 2000.

[7] A. Mandelbaum, A. Sakov, and S. Zeltyn, "Empirical analysis of a call center," William Davidson Faculty Ind. Eng. Manage., Technion-Israel Inst. Technol., Haifa, Israel, Tech. Rep., 2001.

[8] A. Fronzetti and M. Naldi, "Service times of interactive voice response systems in call centers," Dipartimento Informatica Sistemi Produzione, Universitá degli Studi Roma Tor Vergata, Rome, Italy, Internal Rep. RR-06.58, Dec. 2006.

[9] A. Deslauriers, P. L'Ecuyer, J. Pichitlamken, A. Ingolfsson, and A. Avramidis, "Markov chain models of a telephone call center with call blending," *Comput. Operations Res.*, vol. 34, no. 6, pp. 1616–1645, 2007.

[10] J. R. Artalejo, A. Economou, and A. Gómez-Corral, "Applications of maximum queue lengths to call center management," *Comput. Operations Res.*, vol. 34, no. 4, pp. 983–996, 2007.

[11] R. B. Ash, "Noiseless coding," in *Information Theory*. New York: Dover, 1965, pp. 27–45.

[12] T. Paulden and D. K. Smith, "From the Dandelion code to the Rainbow code: A class of bijective spanning tree representations with linear complexity and bounded locality," *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 108–123, Apr. 2006.

[13] E. Thompson, T. Paulden, and D. K. Smith, "The Dandelion code: A new coding of spanning trees for genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 91–100, Feb. 2007.

[14] R. Stolletz, "Characterization of inbound call-centers," in *Performance Analysis and Optimization of Inbound Call Centers*. Berlin, Germany: Springer, 2003, pp. 3–20.

[15] G. Barber, B. Cleveland, and H. Dortmans, "Forecasting the work load," in *Call Center Forecasting and Scheduling*. Annapolis, MD: Call Center Press, 2000, pp. 14–37.

[16] D. A. Huffman, "A method for the construction of minimum-redundancy codes," in *Proc. Instit. Radio Engineers*, 1952, pp. 1098–1101.

[17] R. G. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Inform. Theory*, vol. 24, no. 6, pp. 668–674, Nov. 1978.

[18] S. Piccioto, "How to encode a tree," Ph.D. dissertation, Dept. Math., Univ. California, San Diego, CA, 1999.

[19] N. Gans, G. Koole, and A. Mandelbaum, "Telephone call centers: Tutorial, review, and research prospects," *Manuf. Service Operations Manage.*, vol. 5, no. 2, pp. 79–141, 2003.
[20] M. S. Aguir, O. Z. Akşin, F. Karaesmen, and Y. Dallery, "On the interaction between retrials and sizing of call centers," *Eur. J. Operational Res.*, vol. 191, no. 2, pp. 398–408, Dec. 2008.
[21] G. K. Zipf, *Human Behavior and the Principle of Least Effort*. New York: Addison-Wesley, 1949.
[22] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE Infocom*, 1999, pp. 126–134.
[23] C. L. Hu and M. S. Chen, "On-line scheduling sequential objects for dynamic information dissemination," in *Proc. IEEE Globecom*, 2005, pp. 105–109.
[24] V. Darlagiannis and D. Hausheer, "Design, scalability and application of peer-to-peer overlay networks," *Deliverable 5.1 of the MMAPPS EU Project IST-2001-34201*, Aug. 2004.
[25] X. Liu and R. Deters, "An efficient dual caching strategy for web service-enabled PDAs," in *Proc. 22nd Annu. Assoc. Comput. Machinery Symp. Appl. Comput.*, 2007, pp. 126–134.
[26] A. Whitaker, M. Shaw, and S. Gribble, "Denali: Lightweight virtual machines for distributed and networked applications," in *Proc. USENIX Annu. Tech. Conf.*, 2002.

Sancho Salcedo-Sanz was born in Madrid, Spain, in 1974. He received the B.S. degree in physics from the Universidad Complutense de Madrid, Madrid, Spain, in 1998, and the Ph.D. degree in telecommunications engineering from the Universidad Carlos III de Madrid, Madrid, Spain, in 2002.

He spent one year at the School of Computer Science, University of Birmingham, Birmingham, U.K., as a Postdoctoral Research Fellow. Currently, he is an Associate Professor with the Departamento de Teoría de la Señal y Comunicaciones, Escuela Politécnica Superior, Universidad de Alcalá, Madrid, Spain. He has co-authored more than 80 international journal and conference papers in the field of operations research and artificial intelligence. His current research interests include optimization problems in science and engineering, genetic algorithms, hybrid algorithms, and neural networks.

Maurizio Naldi (M'90–SM'02) was born in Palermo, Italy, in 1963. He graduated cum laude in electronic engineering from the University of Palermo, Palermo, Italy, in 1988, and received the Ph.D. degree in telecommunications engineering from the University of Rome at Tor Vergata, Rome, Italy, 1998.

After graduation he pursued an industrial career, first at Selenia as a Radar Designer from 1989 to 1991, and then in the Network Planning Departments of Italcable from 1991 to 1994, Telecom Italia from 1995 to 1998, and WIND from 1998 to 2000, where he was appointed Head of the Traffic Forecasting and Network Cost Evaluation Group. In the 1992–2000 period, he was active in the standardization bodies (ETSI and ITU), in particular as an Associate Rapporteur for Broadband Traffic Measurements and Models at ITU Study Group 2. Since 2000, he has been with the Dipartimento di Informatica, Sistemi e Produzione, Universitá di Roma Tor Vergata, where he is now an Aggregate Professor. He is the author of more than 100 scientific publications in the fields of computer networks and traffic analysis. His current research interests include computational advertising and network economics.

Dr. Naldi is a Member of the Mathematical Association of America.

Ángel M. Pérez-Bellido was born in Madrid, Spain, in 1984. He received the B.S. and M.S. degrees in telecommunication engineering, in 2007 and 2009, respectively, from the Universidad de Alcalá, Madrid, Spain, where he is currently pursuing the Ph.D. degree.

He is currently a Research Fellow with the Departamento de Teoría de la Señal y Comunicaciones, Escuela Politécnica Superior, Universidad de Alcalá. His research interests include evolutionary computation, information theory topics, and optimization problems in telecommunications.

Jose A. Portilla-Figueras was born in Santander, Spain, in 1976. He received the B.S. and Ph.D. degrees in telecommunications engineering from Universidad de Cantabria, Cantabria, Spain, in 1999 and 2004, respectively.

He was a Research Fellow at the Department of Computer Engineering, Universidad de Cantabria. He is currently an Associate Professor with the Departamento de Teoría de la Señal y Comunicaciones, Escuela Politécnica Superior, Universidad de Alcalá, Madrid, Spain. His main research interests include networking problems and application of meta-heuristics algorithms in network optimization problems.

Emilio G. Ortíz-García was born in Madrid, Spain, in 1984. He received the B.S. and M.S. degrees in telecommunication engineering in 2007 and 2009, respectively, from Universidad de Alcalá, Madrid, Spain, where he is currently pursuing the Ph.D. degree.

He is currently a Research Fellow with the Departamento de Teoría de la Señal y Comunicaciones, Escuela Politécnica Superior, Universidad de Alcalá. His main research interests include evolutionary computation, support vector machines improvement and optimization, and regression problems in engineering.