EvoGeneS, a New Evolutionary Approach to Graph Generation

L. P. Cordella[1], C. De Stefano[2], F. Fontanella[1] and A. Marcelli[3]

[1] Department of Information Engineering and Systems
University of Naples,
Via Claudio, 21 80125 Naples – Italy
{cordel, frfontan}@unina.it
[2] Department of Automation, Electromagnetism, Information Engineering and Industrial Mathematics
University of Cassino
Via G. Di Biasio, 43 02043 Cassino (FR) – Italy
destefano@unicas.it
[3] Department of Computer Science and Electrical Engineering
University of Salerno
84084 Fisciano (SA) – Italy
amarcelli@unisa.it

Abstract. Graphs are powerful and versatile data structures, useful to represent complex and structured information of interest in various fields of science and engineering. We present a system, called EvoGeneS, based on an evolutionary approach, for generating undirected graphs whose number of nodes is not a priori known. The method is based on a special data structure, called multilist, which encodes undirected attributed relational graphs. Two novel crossover and mutation operators are defined in order to evolve such structure. The developed system has been tested on a wireless network configuration and the results compared with those obtained by a genetic programming based approach recently proposed in the literature.

1 Introduction

Graphs are powerful and versatile data structures, useful to represent complex and structured information. In the last decades, there has been an increasing interest in studying and using graphs in many applications, also because the developments of computer technology made high computational cost problems to be dealt with.

Graphs have been used in various fields of science and engineering. They may effectively represent physical networks, such as transportation systems, power systems, and mobile communication infrastructures [1–3], but have been also used to model less tangible interactions, as might occur in ecosystems, databases or in the control flow of a computer program [1]. In fields like pattern recognition and machine vision, the high representational power of graphs make

them very attractive and well-suited to model complex patterns in terms of parts and their relations. Attributes of graph nodes and edges are often added to incorporate further information, leading to a graph representation form generally called Attributed Relational Graph (ARG) [4]. Examples of successful applications include shape analysis and 3-D object recognition [5, 6], character recognition [7], classification of ideograms and symbols in document analysis and technical drawing interpretation [8].

In many cases, a prominent problem is that of generating graphs exhibiting some particular properties. The generation of prototypes in pattern recognition problems, so as the generation of the optimal configuration of a physical network are examples of such problem. Thus, the use of graph representations often requires the definition of effective techniques for generating the graphs representing the desired solutions. To this purpose, two main different approaches can be identified, depending on the nature of the problem: in case of applications in which training samples are available, the graphs may be generated by exploiting the information included in such a training set. In all the other cases the solution is found by defining a function $\mathcal{F}$ able to measure the goodness of tentative solutions in a given space: the graphs representing the solutions are generated by finding all the absolute maxima of the function $\mathcal{F}$. Combinatorial, heuristic and inductive learning approaches have been used, among others, to generate graphs [9]. Several attempts to generate graphs using evolutionary approaches have also been done. Methods have been proposed in the fields of molecular design [10] and electrical circuit design [11], using a direct encoding of the evolving graph. It is worth noting that these methods define evolutionary operators tailored for the considered problem. Indirect encoding of graphs in terms of bit strings [12] or trees [13] has also been used. In the latter approach, for instance, a tree encodes the operations to be applied to a very simple starting graph, in order to transform it into another one arbitrarily complex.

We present a system, called *EvoGeneS* (Evolutionary Graph Generation System), based on an evolutionary approach, for generating graphs whose number of nodes is not a priori known. The proposed method aims at overcoming two major disadvantages of the methods discussed earlier by providing a direct encoding of graphs and two novel, general purpose and problem independent operators. A special structure, called multilist, encodes undirected ARG's, and demonstrated to be particularly convenient for generating new and different graphs under given constraints. For evolving multilists, two basic operators have been devised: the first one, called *crossover* by analogy to genetic algorithms, swaps parts of two multilists, thus swaps subgraphs of two graphs, thus generating graphs of variable length. The second operator, called *mutation*, operates on a multilist in such a way to change a graph into a new one whose node number is unchanged, whereas both node and link attributes can be modified.

In the following, after defining the multilist and the elementary operations defined for it, an application of the proposed evolutionary system will be illustrated. The results obtained by EvoGeneS will be compared with those of

EvoGraph, the approach described in [13], showing that EvoGeneS performance overcomes that of this Genetic Programming based approach.

2 *EvoGeneS*

EvoGeneS is essentially based on two elements: a new data structure encoding undirected relational graphs with attributes and two operators devised for such structure.

2.1 Graph Encoding

Let us consider a graph G of N nodes. Let also denote by A_n and A_a the sets of values for the attributes describing the nodes and the arcs of the graph, respectively. The data structure we have adopted for representing attributed relational graphs has been called *multilist* (ML in the following) since it is based on the list concept and consists of two basic lists. The first one, called *main list*, contains the information on graph node attributes, thus its number of elements is equal to the number N of nodes. Each element of the second list is on its turn a list, called *sublist*. One sublist is associated with each node and includes the attributes of the arcs connected to that node. In order to preserve information about the nodes interconnected by each arc, arc attributes are sorted in each sublist in a suitable order. Namely, the i-th sublist contains information on the arcs connecting the i-th node of the graph to the nodes following it in the main list, in the order they appear in such list. If two nodes are not connected, this information is anyway suitably stored in the proper place of a sublist. In practice, a "null" relation has been defined so that even the absent arcs are encoded in the ML representation of a graph (see Fig. 1). The *length* of a ML is defined as the number of elements of its main list. It is important to notice that, in this paper, we consider simple (i.e., without loops) undirected graphs, so that the relation linking the i-th node to the j-th node coincides with the relation linking the j-th node to the i-th node. For this reason, the length of the sublist associated with a node decreases as the position of the node in the main list increases: the first sublist is made of $N-1$ elements, the second sublist has $N-2$ elements and so on. In fact, the information on the link between each node and the previous ones in the main list is already expressed in the previous sublists. As a consequence, the sublist of the last node of the graph is void. Thus a ML has a triangular shape: the base of the triangle is the main list and is long N, while the height is represented by the first sublist and is long $N-1$. In the following, the operations defined on the ML's will be introduced.

2.2 The Operators

The just described data structure has been devised in such a way to make easier the application of operators able to generate new and different items from previously generated ones. Two basic operators have been defined for the ML:

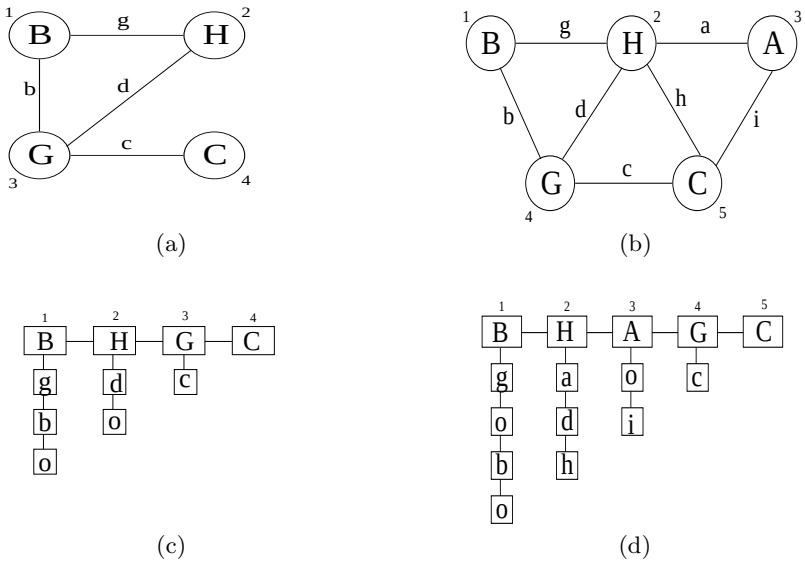


Fig. 1. Two graphs (top) and their encoding multilists (bottom). The horizontal list is the main one and the vertical lists are the sublists. The elements of the set A_n are denoted by capital letters, while those of A_a are denoted by small letters. The null element is denoted by the letter 'o'.

they have been called crossover and mutation by analogy to genetic algorithm operators. The former operator swaps parts of two ML's. In this way, it is possible to generate better solutions by combining solutions that contain only part of a good solution. The mutation operator, instead, generates a new graph whose number of nodes is unchanged, whereas the attributes of both nodes and arcs can be modified.

Crossover Operator The crossover is applied to two ML's, $L^{'}$ and $L^{''}$, called in the following *parents*, respectively encoding the graphs $G^{'}$ and $G^{''}$, and generates two new ML's, $M^{'}$ and $M^{''}$, called *offspring*, respectively encoding the graphs $H^{'}$ and $H^{''}$. The crossover operator allows generating ML's of variable length. In fact, if the parents are of length $N^{'}$ and $N^{''}$ respectively, the length of the offspring may varies in the interval $[2, (N^{'} + N^{''}) - 2]$. The operator is obtained by combining two more elementary operations that can be applied to a ML. The former, called *t-cut*, splits a generic ML L of length N in two ML's, the first one consisting of the first t nodes of L and the second one of the remaining $N - t$ nodes. The latter operation, instead, is called *merge* and, given two ML's $L^{'}$, of length $N^{'}$, and $L^{''}$, of length $N^{''}$, yields a new ML of length $(N^{'} + N^{''})$, encoding a graph including both the nodes of $G^{'}$ and $G^{''}$. To show how the crossover works, let $L^{'}$ be the ML of Fig. 1(a) ($N^{'} = 4$) and $L^{''}$ that of Fig. 1(b) ($N^{''} = 5$). Then, to apply the crossover to these ML's, two integers $t_1 \in [1, N^{'} - 1]$ and

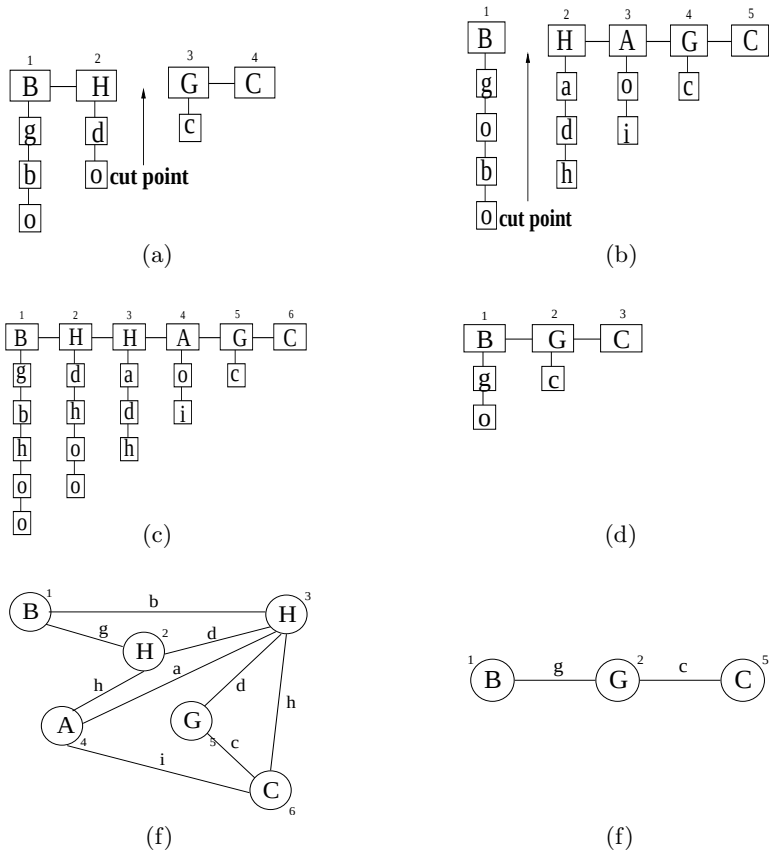


Fig. 2. The crossover operator. (a) and (b) the application of the t-cut to the multilists of Fig. 1. (c) and (d) The offspring obtained after the merge. (e) and (f) The resulting graphs.

$t_2 \in [1, N^{''} - 1]$ have to be randomly chosen. Let $t_1 = 2$ and $t_2 = 1$, then the 2-cut operation is applied to $L^{'}$ and two ML's are obtained: $L_1^{'}$ and $L_2^{'}$, both of length 2 (see Fig. 2(a)). Afterwards, the 1-cut operation is applied to $L^{''}$, which yields two ML's: $L_1^{''}$ of length 1 and $L_2^{''}$ of length 4 (see Fig. 2(b)). At this point, the merge operation is applied to $L_1^{'}$ and $L_2^{''}$: it yields a ML of length 6, which represents our first offspring (see Fig. 2(c)). The merge operation has to be applied also to the remaining ML's, $L_2^{'}$ and $L_1^{''}$. In this case a ML of length 3 is obtained, which represents the second offspring (Fig. 2(d)). The obtained graphs are shown in Figures 2(e) and 2(f). Note that the length of the offspring depends on the values chosen for t_1 and t_2.

Mutation Operator The mutation operator defined here, actually gives place to a sort of micro-mutation, because it does not modify the structure of the

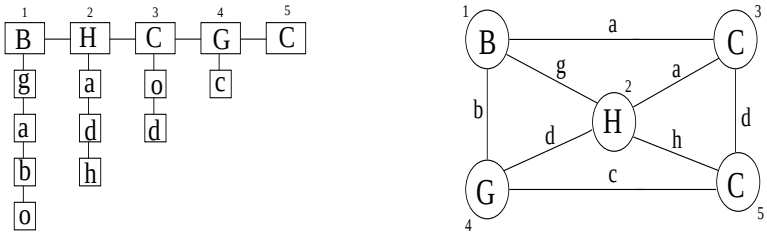


Fig. 3. The multilist (left) and the corresponding graph (right), derived from the application of the mutation operator, with a probability equal to 0.1, to the multilist of Fig. 1(b). The mutation modifies the attribute of node 3 and that associated with the arc which links nodes 3 and 5. Moreover, the mutation added a new arc, which links nodes 1 and 3, absent in the graph before of the application of the mutation operator.

ML to which it is applied, but only the values of the elements of the main list and of the sublists. Such an operation is based on a probability value, called *mutation probability* (p_m in the following). For each element in the main list, p_m represents the probability to replace its value with another one randomly chosen from the set A_n. The same occurs for the elements of the sublists, but in this case the value can be replaced either by one belonging to the set A_a or by the null value. Let L be a generic ML and $L^{'}$ be the ML produced by the application of the mutation operator to L. Let us examine the possible differences between the graphs encoded by L and $L^{'}$. Both graphs contain the same number of nodes, while the number of arcs of the two graphs may be different. In fact, when the mutation is applied to an element of the main list it changes only the attributes of that node leaving the number of nodes unchanged. Instead, when the mutation is applied to an element of a sublist, and either a null element is changed to a not null one or vice versa, the corresponding arc is added to or removed from the original graph, respectively. Finally, if a not null element is replaced by a different not null element, then both the graphs G and $G^{'}$ will contain the arc represented by that element, but the relation associated with the arc in G is different from that in $G^{'}$ (see Fig. 3). We should note that, generally, the differences between the graphs G and $G^{'}$ are directly proportional to the mutation probability p_m.

2.3 The Algorithm

The evolutionary algorithm implemented in EvoGeneS starts by generating at random a population of P individuals. Each individual is a ML encoding a graph representing a solution of the problem to be solved. The length of these initial individuals range from 2 to N_{max} nodes. Afterwards, the fitness of the individuals generated is evaluated. To generate a new population, first the best E individuals are selected and copied in the new population in order to implement an elitist strategy. Then $(P - E)/2$ couples of individuals are selected using the tournament method, to control loss of diversity and selection intensity. The

crossover operator is applied to each of the selected couples, according to a chosen probability factor p_c. Afterwards, the mutation is applied to the individuals according to a probability factor p_m. Finally these individuals are added to the new population. The process just described is repeated for N_g generations.

3 Testing the approach

In order to ascertain the effectiveness of the proposed approach, we have chosen a planning and optimization problem. To cope with this kind of problems, several approaches have been proposed in the literature, including genetic programming [13], simplex method [14], simulated annealing [15], Tabu search [16] and genetic algorithms [17]. In particular in [13], the wireless access point configuration problem, a hard non-linear optimization problem, has been considered. We have chosen the same problem, in order to compare our results with those presented in [13].

The scenario of the problem is the following: a community is planning to provide wireless Internet service to its citizens (clients) who are scattered around a given area. A certain number of access points need to be placed to cover all clients, because each access point has a limited service radius. All access points are wired and one of them is connected to an Internet gateway. The design problem consists in determining the optimal configuration of the AP's in the area to cover. To reduce the cost, a configuration with minimal number of AP's and minimum length of the wires connecting them is considered optimal. According to the constraints imposed, the wireless access point configuration problem can be formulated in different ways. E.g., [18] assumes that the AP's are located at a specified set of possible points. We assume that the AP's can be located at any place. More precisely, the problem to solve is defined as follows:
GIVEN a set of N_C clients located at (x_i^c, y_i^c) $i = 1 \dots N_C$ in an area of size $W \times H$ where $x_i^c \in [0, W]$ and $y_i^c \in [0, H]$, and the gateway G located at (x^g, y^g), let us assume that all AP's are equal and that the service radius of an AP is r_s;
FIND a configuration of wired access points located at (x_i^{AP}, y_i^{AP}) with $i = 1 \dots N_{AP}$, connected to the gateway port G in such a way that each client is covered by at least one AP and the total cost of the AP's and the wires is minimal. Thus, let C_{AP} be the cost of each AP and C_w the cost of a unit length wire, the aim is:
minimize $f = C_{AP} * N_{AP} + C_w * \sum |L_i|$
where the L_i are the lengths of the connections among AP's.

A more precise solution of the problem would require considering some parameters of an AP like transmission power, channel allocation and antenna directionality. In this paper, such parameters are not considered. Nevertheless, the proposed formulation keeps the essentials of the wireless configuration problem, avoiding a time-consuming simulation for evaluating the fitness of a configuration.

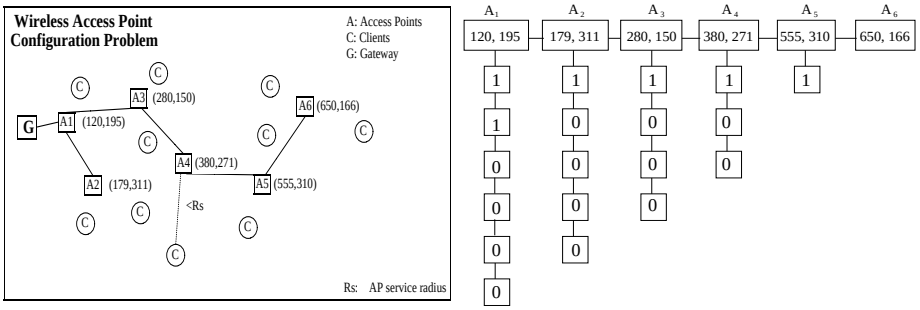


Fig. 4. An instance of the wireless access point configuration problem (left) and the multilist encoding it (right). The citizens (clients) are labeled as circled C, while the access point are represented by squares.

The Fitness Function To solve the problem, a configuration of AP's is represented with a graph whose nodes are the AP's and whose arcs are the wire segments connecting the AP's. The set of node attributes is made up of the AP coordinates in the area to cover (see Fig. 4). In the problem at hand, it is necessary to know only which nodes are linked to a given node. Hence, in the ML representation encoding the graph, the value 1 is used to indicate the presence of an arc, while the 0 indicates the absence of an arc.

As mentioned in the previous section the fitness function has to consider three aspects of the problem: the percentage of covered clients, the number of AP's employed and the total length of the wires connecting them. For this reason, the fitness function is the weighted sum of three terms. The first term F_{cover} should measure how well the clients are covered by the configuration of AP's: the more clients are covered, the better. The second term F_{wires} should measure how good the connection topology is: the shorter the wires used, the better. Finally, the term F_{AP} should estimate the goodness of a configuration as regards the number of wireless AP's employed: now, the fewer AP's are used, the better. It may be convenient that the fitness terms are normalized and suitably weighted, so as to reflect their different importance for evaluating the goodness of a configuration. Since the aim of this paper is that of presenting a general purpose method for graph generation, for the specific problem considered we have adopted the same fitness function as proposed in [13], in order to ascribe any difference in the performance of the methods to the way the solutions are generated, not to the way they are evaluated. Namely, the fitness terms are:

$$F_{\text{cover}} = \frac{4.0 * C_c}{C_c + C_T}; \qquad F_{\text{wires}} = \frac{10000}{10000 + L_w}; \qquad F_{\text{AP}} = \frac{C_T}{C_T + 1.5 * N_{AP}} \tag{1}$$

where C_c is the number of clients covered, C_T the total number of clients, N_{AP} the number of AP's and L_w is the total length of wire segments connecting the AP's. Then, the fitness function F_{tot} is the weighted sum of the above three terms:

$$F_{\text{tot}} = 0.7 * F_{\text{cover}} + 0.1 * F_{\text{wires}} + 0.2 * F_{\text{AP}} \tag{2}$$

Moreover, solutions (i.e., configurations) containing isolated AP's are penalized by multiplying their fitness by 0.5.

4 Experimental Results

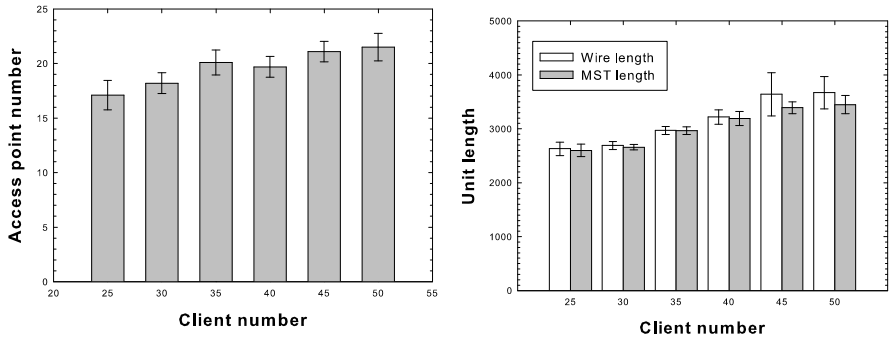
By analogy with the experimental framework presented in [13], we have considered a search space whose length and width are both equal to 1000, and we have assumed, for the sake of simplicity, that the clients are represented in this space by points having integer coordinates. The values of the evolutionary parameters have been experimentally determined and are summarized in Table 1.

In the experiments reported below, the number of clients has been varied starting from 25 up to 50 with increments equal to 5. For each considered value, a distribution of clients has been randomly generated, and 50 runs have been performed with different initialization of the population, so as to reduce the effects of randomness embedded in the evolutionary algorithms. At the end of each run, the best solution found by the algorithm is stored. The corresponding length of the wires connecting the AP's is computed and stored as well. For each distribution of clients, we have computed the mean $\overline{N}_{AP}$ and the standard deviation $\sigma_{N_{AP}}$ of the number of AP's found by our method while performing 50 runs (see Fig. 5(a)). The mean $\overline{L}$ and the standard deviation σ_L of the lengths of the wires have been computed as well (see Fig. 5(b)).

In order to highlight the effectiveness of the obtained results, for each solution provided by our method, we have separately computed the Minimum Spanning Tree (MST) [1] of the corresponding graph. In fact, the MST represents the connection topology with minimal wire cost, thus we have compared the length of the wires relative to the MST with that of our solution. In practice, for the sake of comparison, we have computed the mean $\overline{L'}$ of the wire lengths for each distribution of clients. Each length refers to one of the 50 runs and is obtained by using the MST over the set of AP's provided by our method for that run. The plot of $\overline{L'}$ as a function of the number of clients, is shown in Fig. 5(b). The results are very encouraging because in every run a complete coverage of the clients has been obtained and the wire costs are very close to those computed on the MST. Moreover, the number of AP's needed to solve the problem, as well

Table 1. Values of the basic evolutionary parameters used in the experiments.

Parameter	symbol	value
Population size	P	1000
Tournament size	$\mathcal{T}$	60
elitism size	E	40
Crossover probability	p_c	0.3
Mutation probability	p_m	0.04
Number of Generations	N_g	500
Maximum number of nodes	N_{max}	50



(a) (b)

Fig. 5. (a) The mean number of access points and its standard deviation as a function of the number of clients are respectively represented by bars and segments on top of the bars. (b) The mean of the wire lengths and its standard deviation as a function of the number of clients, computed by our method and by using the MST for finding the connection topology.

as the lengths of their connections, slightly increase with the number of clients. Finally, the standard deviations of both the number of AP's and the wire lengths assume small values, thus indicating that the solutions are widely independent of the initial conditions. Note that, for each distribution of clients, the evolutionary algorithm converges to solutions having almost the same number of AP's and the same wire cost.

Fig. 6 illustrates some results of one of the experiments performed by using a randomly generated distribution of 40 clients. In particular, the best solutions obtained at generation 10, 100, 300 and 500 are shown. During the initial generations, the evolutionary process tends to improve the client covering by adding more and more AP's, without optimizing the connection topology. Only after an almost complete coverage has been obtained, the system tries to reduce the number of AP's and focuses the search on optimizing the connection topology. This behavior can be explained considering that the term F_{cover} in the fitness function has the highest weight, while the term F_{wires} the lowest. For instance, at generation 10, all clients but one are covered using 23 AP's, but the connection topology is messy. At generation 100, all the clients are covered with 21 AP's and the connection topology is significantly improved. At generation 300 the total coverage is obtained with 20 clients and the connection topology is nearly optimal. At generation 500, finally, 19 AP's are used and the topology connection is optimal (i.e., it coincides with the MST of the related graph).

In comparing the results obtained by EvoGeneS with those reported in [13] it is worth noting that our system perform a global optimization in that both the number of AP's and their connections are simultaneously exploited for computing the fitness function. On the contrary, the genetic programming based method, is able to find a solution only when a sequential approach is adopted:

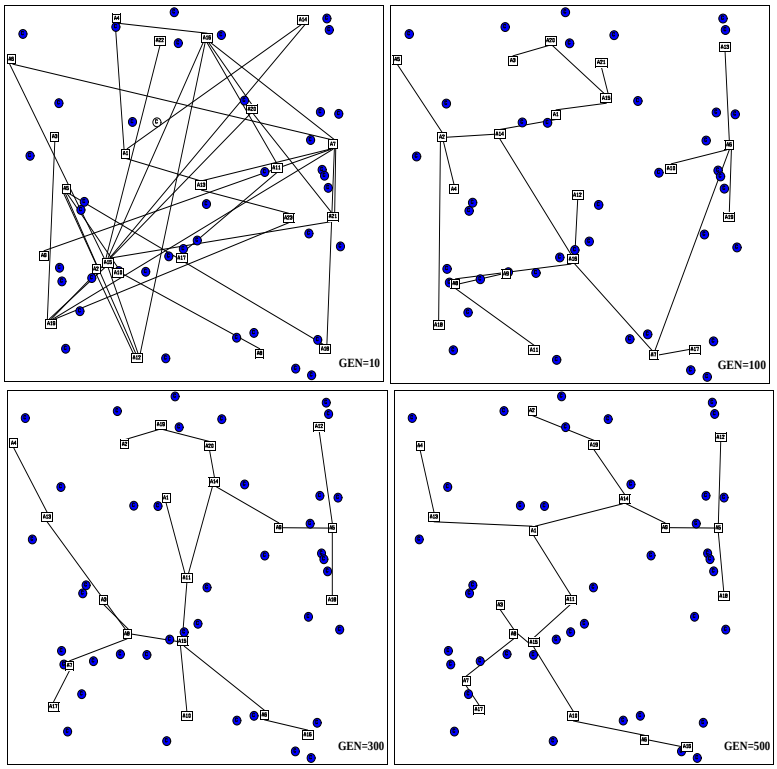


Fig. 6. The best solutions obtained at generation 10, 100, 300, 500, relative to an experiment with 40 clients. Black circles represent the clients covered by at least one access point, while white circles represent uncovered clients. The access points are represented by white squares.

first, solving the coverage problem by GP and then using a MST algorithm for determining the connection topology. Thus, it succeeds only when problem specific knowledge can be exploited to reformulate the original global optimization problem as a sequence of partial optimization problems.

5 Conclusion

We have presented a system, called EvoGeneS, based on an evolutionary approach, for generating undirected graphs whose number of nodes is not a priori known. The proposed approach solves two key problems encountered when using genetic algorithms for evolving graphs: it provides a suitable data structure for the direct representation of graphs and defines crossover and mutation operators for manipulating graphs with a variable number of nodes, i.e. representations of variable length.

The results obtained by EvoGeneS on a wireless access point network configuration problem show a significant improvement with respect to those reported in the literature. Moreover, the approach does not rely on any problem specific knowledge and therefore it is suitable to deal with any problem whose solutions can be represented by graphs.

References

1. Gross, J., Yellen, J.: Graph Theory and Its Application. McGrawHill (2001)
2. Cascetta, E.: Transportation systems engineering: theory and methods. Kluwer Academic (2001)
3. Crow, M.: Computational Methods for Electric Power Systems. CRC Press (2003)
4. Eshera, M.A., Fu, K.S.: A graph distance measure between attributed relational graphs for image analysis. In: Proceedings of 7th Int. Conf. on Pattern Recognition, IEEE Press (1984) 75–77
5. Pelillo, M., Siddiqi, K., Zucker, S.W.: Matching hierarchical structures using association graphs. Lecture Notes in Computer Science **1407** (1998) 3–13
6. Arcelli, C., Cordella, L., di Baja, G.S., eds.: Visual Form 2001, LNCS 2059. Springer-Verlag (2001)
7. Filatov, A., Gitis, A., Kil, I.: Graph-based handwritten digit string recognition. In: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2), IEEE Computer Society (1995) 845
8. Cordella, L.P., Vento, M.: Symbol recognition in documents: A collection of techniques. International Journal on Document Analysis and Recognition (IJDAR) **3** (2000) 73–78
9. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: Learning structural shape descriptions from examples. Pattern Recognition Letters **23** (2002) 1427–1437
10. Globus, A., Lawtonb, J., Wipkeb, T.: Automatic molecular design using evolutionary techniques. In Globus, A., Srivastava, D., eds.: The Sixth Foresight Conference on Molecular Nanotechnology, Westin Hotel in Santa Clara, CA, USA (1998)
11. Naofumi Homma, T.A., Higuchi, T.: Multiplier block synthesis using evolutionary graph generation. In: Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware. (2004) 79–82
12. Lohn, J.D., Colombano, S.P.: Automated analog circuit synthesis using a linear representation. Lecture Notes in Computer Science **1478** (1998) 125+
13. Hu, J., Goodman, E.: Wireless access point configuration by genetic programming. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, Oregon, IEEE Press (2004) 1178–1184
14. Wright, M.H.: Optimization method for base station placement in wireless applications. In: Proceedings of the 1998 IEEE Conference on Vehicular Technology, IEEE Press (1998) 287–291
15. Hurley, S.: Planning effective cellular mobile radio networks. IEEE Transactions on Vehicular Technology **51** (2002) 243–253
16. Lee, C., Kang, H.: Cell planning with capacity expansion in mobile communications a tabu search approach. In: IEEE VTC2000. (2000) 1678–1691
17. K.Lieska, Laitinen, E., Lahteenmaki, J.: Radio coverage optimization with genetic algorithms. In: Proceedings of PIMRC. Volume 1. (1998) 318–322
18. Koichi, E., Yoishinori, W.: Automatic cell design for wide area wireless lan systems. Special Issue on Devices and Systems for Mobile Communications **44(4)** (2003)