ELSEVIER

Pattern Recognition Letters 19 (1998) 1293–1300

Pattern Recognition Letters

Genetic operators for hierarchical graph clustering

Stefano Rizzi [1]

DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

Received 19 May 1998; received in revised form 1 September 1998

Abstract

In this paper we propose an encoding scheme and ad hoc operators for a genetic approach to hierarchical graph clustering. Given a connected graph whose vertices correspond to points within a Euclidean space and a fitness function, a hierarchy of graphs in which each vertex corresponds to a connected subgraph of the graph below is generated. Both the number of clustering levels and the number of clusters on each level are not fixed a priori and are subject to optimization.

Keywords: Hierarchical clustering; Genetic algorithms; Autonomous robots; Euclidean graphs

1. Introduction

Clustering has a key role, in different application fields, in revealing hidden structures and extracting typical prototypes from a dataset. On the other hand, 'flat' clustering gives no information about the structure existing between clusters; hierarchical clustering deals with this issue by grouping data into a tree structure, thus building a multi-level representation capable of revealing inter-cluster relationships. A survey of classical approaches to hierarchical clustering is given by Jain and Dubes (1988).

In the field of autonomous mobile robots, hierarchical clustering may be profitably used to emphasize the structural and topological characteristics of the environments and to derive their high-level representations to be used for reasoning and for planning navigation paths. In particular, we consider an autonomous robot moving in an environment where landmarks corresponding to distinctive places and objects can be detected by sensors such as cameras or sonars. The environment is initially unknown; the robot's mission is to acquire its description incrementally during an exploration process. The map built by the robot is structured as a graph of landmarks and inter-landmark routes; each landmark is represented by a point within the plane. In the navigation-oriented multi-layered architecture for environment representation we proposed in (Maio and Rizzi, 1996), the environment map is hierarchically clustered in order to introduce multiple abstraction levels in the representation and to allow the complexity of path-planning problems to be decreased by applying divide-and-conquer techniques.

Map clustering mainly differs from classical pattern clustering in the following aspects:

- The pattern of connectivity between landmarks must be taken into account when clustering. In

---

[1] Tel.: +39-51-644-3542; fax: +39-51-644-3540; e-mail: srizzi@deis.unibo.it

0167-8655/98/$ – see front matter © 1998 Elsevier Science B.V. All rights reserved.
PII: S0167-8655(98)00110-X

fact, the agent should be enabled to plan partial paths within clusters.
- The objective function to be optimized by clustering takes into account multiple factors, expressing structural and functional requirements; clusters are evaluated both as singles and as a whole.
- The number of levels in hierarchical clustering and the number of clusters on each level are not defined a priori, hence, they must be subject to optimization.

These peculiar aspects make most hierarchical approaches in the literature unsuitable for map clustering (Fisher, 1996; Hofmann and Buhmann, 1995; Hofmann and Buhmann, 1996; Miller and Rose, 1994).

In (Maio et al., 1995) we have shown how map flat clustering can be carried out by optimizing directly, through a genetic algorithm, a fitness function. In this paper we extend the flat approach to the case of hierarchical clustering by proposing an encoding scheme for chromosomes and ad hoc genetic operators. The approach is general since, given a graph in a Euclidean space and a fitness measure defined on a hierarchy of graphs, both the encoding scheme and the operators can be applied for clustering.

2. The graph clustering problem

Definition 1. Let $\mathscr{G} = (V, E)$ be a non-directed graph; with $(v', v'') \in E$ we denote the edge connecting the two vertices $v', v'' \in V$. Let a partitioning $\xi = \{V_1, \ldots, V_m\}$ of V be given; we call *clusters* the m subgraphs $\mathscr{C}_1 = (V_1, E_1), \ldots, \mathscr{C}_m = (V_m, E_m)$, where

$$E_i = \{(v', v'') \in E \mid v' \in V_i \wedge v'' \in V_i\} \quad \text{for } i = 1, \ldots, m.$$

We call the *bridge* between $\mathscr{C}_i$ and $\mathscr{C}_j$ the set of the edges connecting one vertex in $\mathscr{C}_i$ with one vertex in $\mathscr{C}_j$:

$$(\mathscr{C}_i, \mathscr{C}_j) = \{(v', v'') \in E \mid v' \in V_i \wedge v'' \in V_j\} \quad \text{for } i, j = 1, \ldots, m,\ i \neq j.$$

A *clustering* on $\mathscr{G}$ is defined as a partitioning ξ in which every cluster produced is a connected graph.

Definition 2. Given a non-directed graph $\mathscr{G}$ and a clustering ξ, we call the *image of $\mathscr{G}$ through ξ* the graph $\mathscr{G}^* = (V^*, E^*)$ whose vertices and edges are, respectively, the clusters and the non-empty bridges induced by ξ:

$$V^* = \{\mathscr{C}_1, \ldots, \mathscr{C}_m\},$$
$$E^* = \{(\mathscr{C}_i, \mathscr{C}_j) \mid (\mathscr{C}_i, \mathscr{C}_j) \neq \emptyset,\ i, j = 1, \ldots, m, i \neq j\}.$$

It can be easily proved that the image of a connected graph is itself connected (Maio and Rizzi, 1996).

Definition 3. A *hierarchical clustering of height n* is a sequence of n clusterings, each applied to the image graph generated by the preceding clustering, which produces a hierarchy of $n + 1$ graphs (including the original graph).

Let a non-directed connected graph $\mathscr{G}^{(0)}$ and a hierarchical clustering $\xi^{(1)}, \ldots, \xi^{(n)}$ be given. We call $\mathscr{G}^{(0)}$ the *0-graph*, and the connected graph $\mathscr{G}^{(k)}$ $(k = 1, \ldots, n)$, image of $\mathscr{G}^{(k-1)}$ through $\xi^{(k)}$, the *k-graph*. We call *k-vertices* the vertices of the k-graph $(k = 0, \ldots, n)$, and *k-edges* its edges.

A k-vertex is denoted by $\mathscr{V}_i^{(k)}$ and corresponds to a subgraph of the $(k - 1)$-graph; its *cardinality* is defined as the number of $(k - 1)$-vertices it contains. We will assume that the n-graph always contains exactly one n-vertex, which corresponds to the whole $(n - 1)$-graph.

Definition 4. Let each 0-vertex $\mathscr{V}^{(0)}$ be associated to a point in a d-dimensional Euclidean space, $\mathbf{pos}(\mathscr{V}^{(0)})$, which we call its *position*. We define the *position* of a k-vertex $\mathscr{V}^{(k)}$ as:

$$\mathbf{pos}(\mathscr{V}^{(k)}) = \operatorname{avg}\{\mathscr{V}_i^{(k-1)} \in \mathscr{V}^{(k)}\}, \left(\mathbf{pos}(\mathscr{V}_i^{(k=1)})\right) \quad \text{for } k = 1, \ldots, n,$$

where $\operatorname{avg}_S(f)$ denotes the average of function f extended to the set S.

2.1. A fitness function for map clustering

Let $\xi^{(k)}$ be the flat clustering which, applied to the $(k - 1)$-graph $\mathscr{G}^{(k-1)}$, produces the k-vertices which constitute the k-graph $\mathscr{G}^{(k)}$. Our approach to determine the optimal clustering $\xi^{(k)}$ is guided by the six requirements summarized below, each justified with reference to the application field of autonomous agents.

- *Visibility*. In order to ensure that an autonomous agent has a good intra-cluster mobility, an upper bound $\rho_{(k)}$ on the maximum radius of k-vertices is placed.
- *Equi-cardinality*. Clustering can be used to decompose management and planning algorithms following a divide-and-conquer policy, so as to decrease their complexity. To this end, all k-vertices should have the same cardinality.
- *Predictability*. High cluster cardinality leads to management complexity; on the other hand, low cardinality strongly reduces the effectiveness of decomposing methods. We introduce a parameter, $\eta_{(k)}$, which denotes the required average cardinality of k-vertices.
- *Homogeneity*. Clustering should reveal the relevant topological features of the environment; to this end, density of $(k-1)$-vertices should be homogeneous within each k-vertex.
- *Cohesion*. Clusters should be cohesive, meaning that the average length of the external edges should overcome that of the edges belonging to clusters. The aim of this criterion is, when $(k-1)$-vertices are arranged into one or more high-density areas surrounded by low-density belts, to have cluster borders lying on these belts.
- *Regularity*. Irregularly-shaped clusters may cause divide-and-conquer path-planning algorithms to generate non-optimal solutions; therefore, clusters should be regularly shaped.

These requirements can be formalized by defining, for clustering $\xi^{(k)}$, a fitness measure $f(\xi^{(k)}, \rho_{(k)}, \eta_{(k)})$ as a weighted sum of six components, each expressing the degree to which $\xi^{(k)}$ meets one of the criteria listed above and ranging from 0 (no adherence) to 1 (maximum adherence). The mathematical details concerning the definition of f can be found in (Maio et al., 1996).

The first step in generalizing fitness to the hierarchical case consists in discussing how $\rho_{(k)}$ and $\eta_{(k)}$ depend on k. As to parameter $\eta_{(k)}$, we may assume that the required cardinality of clusters is the same on all levels, and thus choose

$$\eta_{(k)} = \eta \quad \text{for } k = 1, \ldots, n.$$

The value of $\rho_{(k)}$, instead, should increase with k (clusters at high abstraction levels are larger than those at low levels). A k-vertex includes η $(k-1)$-vertices; roughly, the maximum radius of a k-vertex in the d-dimensional space will be

$$\rho_{(k)} = \rho_{(k-1)}\eta^{1/d} \quad \text{for } k = 2, \ldots, n.$$

If $\rho(1) = \rho$ is the radius of a 1-vertex, we may write

$$\rho_{(k)} = \rho\eta^{(k-1)/d} \quad \text{for } k = 1, \ldots, n.$$

The global fitness g for a hierarchical clustering ξ of height n is defined as the average of the fitnesses of the single levels

$$g(\xi, \rho, \eta) = \frac{1}{n}\sum_{k=1}^{n} f\left(\xi^{(k)}, \rho_{(k)}, \eta_{(k)}\right).$$

3. Applying genetic algorithms to map hierarchical clustering

Different alternatives for encoding the problem of object partitioning have been proposed in the GA literature (Babu and Murty, 1994; Bhuyan et al., 1991; Jones and Beltramo, 1990; Krovi, 1992). Map clustering is more difficult than classic partitioning problems, since the connectivity constraint makes most solutions unacceptable. The approach based on rejection of the inconsistent solutions cannot be pursued due to the huge number of solutions that include non-connected clusters. This justifies the choice of the ad hoc encoding scheme and genetic operators described in the following subsections.

3.1. Encoding scheme

In (1995), Maio et al. proposed an encoding scheme for map flat clustering which resembled that described in (Davis, 1985). Each chromosome consisted of a permutation of the vertices and a separator splitting the string in two, and was mapped into one consistent solution to the clustering problem by means of a decoding procedure in which first each vertex in the left part of the string was used to initialize a different cluster, and then the vertices in the right part were progressively added to the clusters created. Here we generalize this encoding scheme to hierarchical clustering.

Let $\mathscr{G}^{(0)}$ be a non-directed connected graph including λ vertices. Each chromosome c is represented by a string of length $2\lambda - 1$ in which the λ characters in odd positions consist of a permutation of the first λ integers while the $\lambda - 1$ characters in even positions are separators. Each integer references a vertex in the map; each separator can take value 0 or 1. Let n be the sum of the values of the separators. The n separators set to 1 divide c into $n + 1$ substrings; we denote with c_k the string of integers obtained by dropping all separators from the kth substring of c.

An example of chromosome ($\lambda = 10$) is shown below (separators are in boldface):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	**1**	4	**0**	8	**1**	5	**0**	7	**0**	1	**0**	2	**0**	6	**0**	9	**0**	3

In this case it is $n = 2$ and $c_1 = (0)$, $c_2 = (4,8)$, $c_3 = (5,7,1,2,6,9,3)$.

Chromosome c maps into exactly one hierarchical clustering, with height n, by means of a decoding procedure which is an orderly manner builds each image graph starting from the 1-graph up to the n-graph. At the kth step, the k-graph is built by considering two strings: a k-*seed*, obtained by orderly concatenating $c_1, \ldots, c_{n-k+1}$, and a k-*growth*, c_{n-k+2}; each character in the k-seed and the k-growth represents a ($k - 1$)-vertex built at the previous step. First each ($k - 1$)-vertex in the k-seed is used to initialize a different k-vertex, and then the ($k - 1$)-vertices in the k-growth are progressively added to the k-vertices created. If we denote with $\Delta(\mathbf{p}', \mathbf{p}'')$ the Euclidean distance between two positions $\mathbf{p}'$ and $\mathbf{p}''$, the decoding procedure of c may be sketched as follows:

```
decode(c, $\mathscr{G}^{(0)}$)
{  for k = 1 to n do
      $\xi^{(k)}$ = flatDecode(c, k);
   $\mathscr{G}^{(k)}$ = image($\mathscr{G}^{(k-1)}$, $\xi^{(k)}$);
}

flatDecode(c, k)
// seed(c, k) denotes the k-seed of c, growth(c, k) its k-growth
{  $\xi^{(k)} = \emptyset$;
   for each $\mathscr{V}_i^{(k-1)}$_seed(c, k)
      create new partition $V_i = \{\mathscr{V}_i^{(k-1)}\}$ and add it to $\xi^{(k)}$;
   orderly copy vertices from growth(c, k) to circular list $\mathscr{L}$;
   while length($\mathscr{L}$) > 0
   {  $\mathscr{V}_j^{(k-1)}$ = next($\mathscr{L}$);
      $H = \{V_i \in \xi(k) : \exists (V_j(k-1), V_l^{(k-1)}) \in \mathscr{G}^{(k-1)} : \mathscr{V}_l^{(k-1)} \in V_i\}$;
      if $H \neq \emptyset$
      {  find $V_{best} \in H : \Delta(\mathbf{pos}(\mathscr{V}_j^{(k-1)}), \mathbf{pos}(V_{best}))$ is minimum;
         add $\mathscr{V}_j^{(k-1)}$ to $V_{best}$ and delete it from $\mathscr{L}$;
      }
   }
   return $\xi^{(k)}$;
}
```

Due to the connectivity constraint, it may happen that the ($k - 1$)-vertices in the k-growth cannot be in an orderly manner assigned to k-vertices. When a ($k - 1$)-vertex cannot be assigned to any k-vertex, the algorithm momentarily leaves it apart and tries to assign it again at the next iteration. Since $\mathscr{G}^{(k-1)}$ is a connected graph for every k, it is guaranteed that all vertices are assigned in a finite number of steps.

It is remarkable that, by adopting this decoding technique, all the chromosomes represent a consistent solution to the clustering problem. The number of k-vertices is equal to the length of the k-seed, hence it is determined by the position of the ($n - k + 1$)th separator.

Conversely, there are clusterings that cannot be represented by strings. The existence of "forbidden" clusterings is due to the heuristic criterion used in the decoding algorithm to choose V_{best}, which aims at building compact clusters where near vertices are grouped together. A clustering which cannot be derived in accordance with that criterion is in contrast with the fitness requirements, hence it is not interesting.

Example 1. Consider the simple graph in Fig. 1(a) and the chromosome shown above, encoding a hierarchical clustering of height 2. The 1-seed and the 1-growth are (0,4,8) and (5,7,1,2,6,9,3), respectively; the resulting clustering is shown in Fig. 1(a)

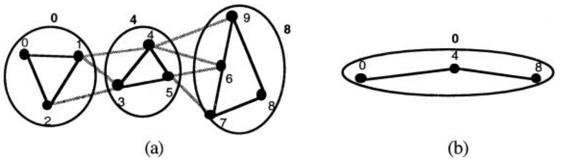

Fig. 1. Hierarchical clustering on a 2-dimensional graph: (a) The 0-graph; (b) the 1-graph.

by drawing in grey the 0-edges belonging to 1-edges, while the corresponding image graph is in Fig. 1(b). The 2-seed and the 2-growth are (0) and (4,8), respectively (each 1-vertex is denoted by the 0-vertex which initialized it); the resulting clustering features one 2-vertex.

3.2. Selection

The selection operator builds a new population, P_{next}, using the chromosomes belonging to the previous one, P. The size of the population, z, remains unchanged; the chromosomes of P_{next} are chosen randomly from P with probability proportional to their fitness. Cloning the best chromosome in P ensures that the best solution obtained at the previous step is not lost. Adopting a linear scaling for the fitness function g allows the fitness range to be normalized in the different phases of evolution (Goldberg, 1989).

3.3. Crossover

Reproduction is based on the *crossover* operator, which is applied to pairs of chromosomes chosen randomly (parents) and combines them to create new pairs with similar features (offspring). The ad hoc crossover operator we designed consists of three different elemental operators which can be applied singularly or consequently.

Height-oriented crossover. This operator acts on the separators, aimed at producing offspring chromosomes, a' and b', with heights equal to the average of the heights of the parent chromosomes, a and b.

Let n_a and n_b be the heights of a and b, respectively. If $n_a = n_b$, the offspring chromosomes are identical to the parents.

Conversely, let $n_a > n_b$. The height of the offspring, n', is $(n_a + n_b)/2$ if it is even, otherwise, it is chosen randomly between $\lfloor (n_a + n_b)/2 \rfloor$ and $\lceil (n_a + n_b)/2 \rceil$ Offspring a' is obtained by setting $na - n'$ separators in a, chosen randomly, to 0. Offspring b' is obtained by setting to $1\ n' - n_b$ separators in b, chosen randomly among those to the left of the rightmost separator set to 1.

Example 2. Consider the two parents below, encoding clustering of heights 3 and 2, respectively.

a	1	**1**	6	**0**	7	**1**	2	**0**	10	**0**	4	**1**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b	5	**0**	3	**1**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

The offspring height n' is chosen randomly between 2 and 3. If $n' = 2$, b' is identical to b and one separator is set to 0 in a; for instance:

a'	1	**1**	6	**0**	7	**0**	2	**0**	10	**0**	4	**1**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b'	5	**0**	3	**1**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

If $n' = 3$, a' is identical to a and one separator is set to 1 in b; for instance:

a'	1	**1**	6	**0**	7	**1**	2	**0**	10	**0**	4	**1**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b'	5	**1**	3	**1**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

Level-oriented crossover. This operator generates offspring chromosomes in which, at each common level, the number of clusters is the average of those of their parents; the heights of the chromosomes are left unchanged. This is obtained by changing the positions of the separators set to 1 without altering the sum of the separators.

After moving the 1 s, the vertices belonging to the 1-seeds of the parents are redistributed randomly among the 1-seeds of the offspring chromosomes; in order to maintain the structure of the highest levels as much as possible, each vertex is extracted with probability inversely proportional to the index of the substring it belongs to.

Example 3. Consider the two parents below, encoding clusterings of heights 3 and 2, respectively.

a	1	**1**	6	**0**	7	**1**	2	**0**	10	**0**	4	**1**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b	10	**0**	3	**1**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	5	**0**	2

Chromosome a encodes 6, 3 and 1 clusters on the first, second and third level, respectively; b encodes

4 clusters on the first level and 2 clusters on the second. The number of clusters in the offspring chromosomes should be 5 on the first level and 2.5 on the second (we choose 2 for *a* and 3 for *b*):

a′	1	**1**	6	**1**	7	**0**	2	**0**	10	**1**	4	**0**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b′	5	**0**	3	**0**	11	**1**	0	**0**	1	**1**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

The vertices appearing within the 1-seeds of *a* and/or *b* are then redistributed (where necessary, by swapping a couple of vertices placed in the 1-seed and in the 1-growth):

a′	7	**1**	1	**1**	4	**0**	6	**0**	10	**1**	2	**0**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b′	1	**0**	2	**0**	7	**1**	3	**0**	10	**1**	8	**0**	9	**0**	4	**0**	6	**0**	11	**0**	5	**0**	0

Cluster-oriented crossover. This consists of a *Partially Matched Crossover* (Whitley et al., 1989) applied to the 1-growths, and changes the positions of vertices without altering the values of the separators. Swaps between vertices in the matching sections of the parents are carried out only if they do not require swaps with vertices of the 1-seed.

This operator alters the sequence of vertices in the 1-seed, thus, it directly affects the composition of the 1-vertices; the clusters at the levels above are typically affected only in their positions.

Example 4. Consider the two parents below, encoding clusterings of height 2. Let the matching section be defined (randomly) as indicated by the arrows:

													←	–	–	–	–	–	→				
a	1	**1**	6	**0**	7	**1**	2	**0**	10	**0**	4	**0**	0	**0**	8	**0**	5	**0**	1	**0**	9	**0**	3
b	10	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	5	**0**	2

The partially matched crossover consists in swapping the two matching sections. Vertices 5 and 1 in *a* are not swapped, since vertices 6 and 7 do not belong to the 1-growth of *a*; for the same reason, vertex 9 in *b* is not swapped.

a′	1	**1**	6	**0**	7	**1**	2	**0**	10	**0**	8	**0**	9	**0**	4	**0**	5	**0**	1	**0**	0	**0**	3
b′	10	**0**	3	**1**	11	**0**	0	**1**	7	**0**	4	**0**	9	**0**	8	**0**	5	**0**	1	**0**	6	**0**	2

3.4. Mutation

Each chromosome generated by reproduction has a given probability of mutation. The ad hoc mutation operator we designed consists of three elemental operators which can be applied singularly or consequently.

Height-oriented mutation. This operator increases or decreases by 1 the height of the chromosome by changing the value of a separator chosen randomly.

Example 5. Consider the following chromosome:

a	5	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

By setting to 1 the 7th separator, the height is increased from 2 to 3:

a′	5	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	9	**1**	4	**0**	6	**0**	7	**0**	10	**0**	2

Level-oriented mutation. This operator modifies the number of clusters on a level by moving one of the separators set to 1, chosen randomly, one position backwards or forwards; as a result, one vertex moves from the seed to the growth or vice versa. In order to avoid the vertices adjacent to separators being the only candidates for this mutation, the vertex involved is inverted with the one in a position chosen randomly within the same substring. By doing so, all the vertices have the same probability of being moved from one substring to the other.

Example 6. Consider the following chromosome:

a	5	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

By moving the 4th separator to the right, after inverting vertex 1 with vertex 4, a mutates to:

a′	5	**1**	3	**0**	11	**0**	0	**0**	4	**1**	8	**0**	9	**0**	1	**0**	6	**0**	7	**0**	10	**0**	2

Cluster-oriented mutation. This operator works on the 1-growth by inverting the vertices in two random positions. Since our decoding technique considers the order in which vertices appear in the growth, this operator alters the structure of clusters.

Example 7. Consider the following chromosome:

a	5	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	9	**0**	4	**0**	6	**0**	7	**0**	10	**0**	2

By inverting vertices 9 and 7, *a* mutates to:

a′	5	**1**	3	**0**	11	**0**	0	**1**	1	**0**	8	**0**	7	**0**	4	**0**	6	**0**	9	**0**	10	**0**	2

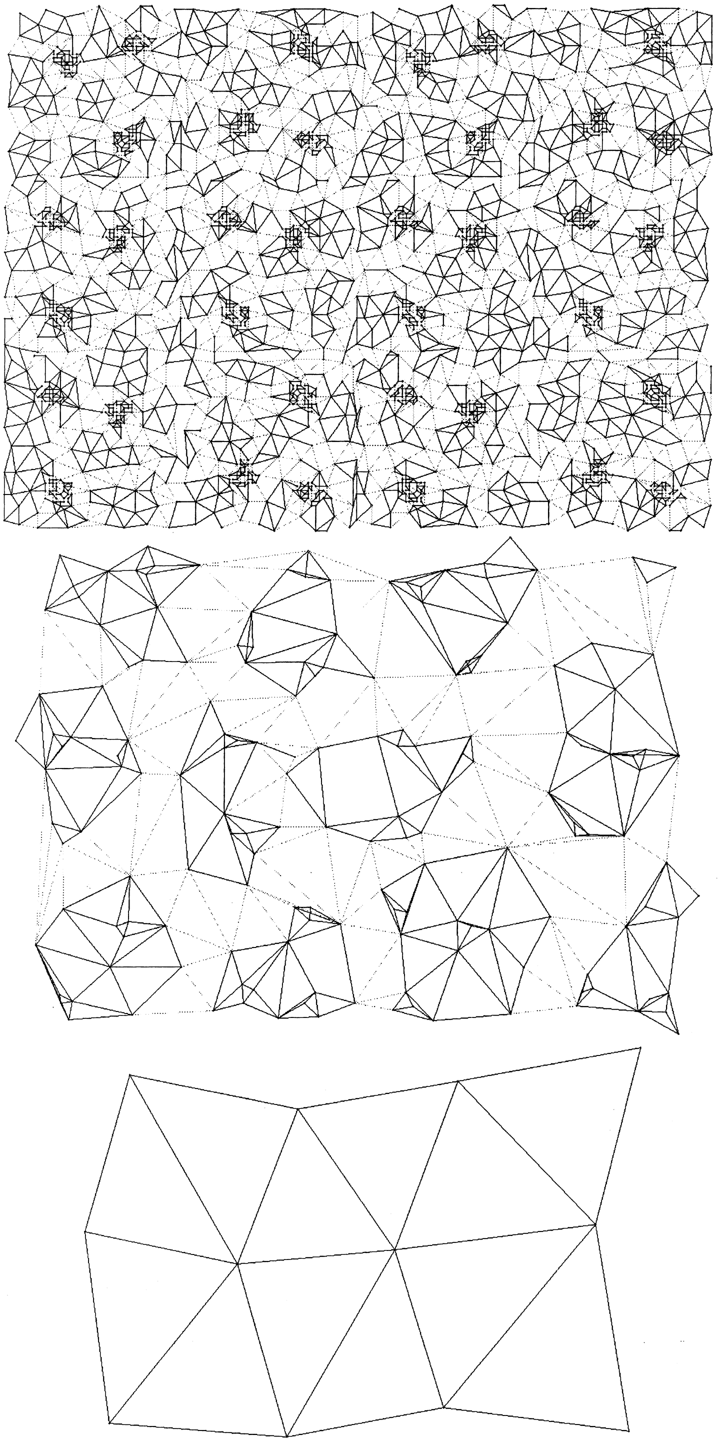

Fig. 2. Hierarchical clustering on a sample map ($\eta = 15$, $\rho = 250$): the 0-graph (top), the 1-graph (middle) and the 2-graph (bottom); the 3-graph includes only one vertex.

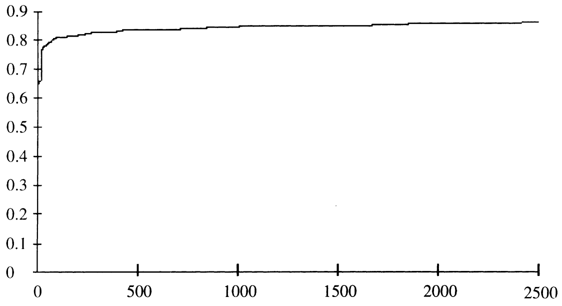

Fig. 3. Fitness during evolution for the clustering in Fig. 2. The fitness displayed for each generation is that of the best chromosome in the current population (population size $z = 20$).

4. Conclusion

In this work we have described a technique for hierarchical clustering of a connected Euclidean graph. The clusters obtained are connected graphs; the number of clustering levels and the number of clusters are not defined a priori, and are subject to optimization.

With reference to the robotics application domain, where vertices are positioned in the 2-dimensional space, Fig. 2 shows the hierarchical clustering obtained on a sample map. The diagram in Fig. 3 shows how the fitness varied during the evolutionary process corresponding to the clustering in Fig. 2. Even after a relatively small number of generations (100–200), the solutions yielded are usually very good.

References

Babu, G.P., Murty, M.N., 1994. Clustering with evolution strategies. Pattern Recognition 27 (2), 321–329.

Bhuyan, J., Raghavan, V., Elayavalli, V. 1991. Genetic algorithms for clustering with an order representation. In: Proc. of the 4th Int. Conf. on Genetic Algorithms and their Application, pp. 408–415.

Davis, L. 1985. Job shop scheduling with genetic algorithms. In: Proc. First Intl. Conf. on Genetic Algorithms, pp. 136–140.

Fisher, D., 1996. Iterative optimization and simplification of hierarchical clusterings. J. Artificial Intelligence and Research 4, 147–179.

Goldberg, D. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Hofmann, T., Buhmann, J.M. 1995. Hierarchical pairwise data clustering by mean-field annealing. In: Proc. Fourth Intl. Conf. on Artif. Neural Networks, Paris, France.

Hofmann, T., Buhmann, J.M. 1996. Inferring hierarchical clustering structures by deterministic annealing. In: Proc. Second Intl. Conf. Knowledge Discovery and Data Mining, Portland, Oregon.

Jain, A., Dubes, R. 1988. Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ.

Jones, D.R., Beltramo, M.A. 1990. Solving partitioning problems with genetic algoritms. Research Report 48090-9055, General Motors Research Laboratories, Warren, MI.

Krovi, R. 1992. Genetic algorithms for clustering: A preliminary investigation. In: Proc. 25th Intl. Conf. on System Sciences, vol. 4, Kauai, Hawaii, pp. 540–544.

Maio, D., Maltoni, D., Rizzi, S., 1995. Topological clustering of maps using a genetic algorithm. Pattern Recognition Lett. 16, 89–96.

Maio, D., Maltoni, D., Rizzi, S., 1996. Dynamic clustering of maps in autonomous agents. IEEE Trans. Pattern Analysis Mach. Intell. 18 (11), 1080–1091.

Maio, D., Rizzi, S. 1996. Layered knowledge architecture for navigation-oriented environment representation. Technical Report CIOC-C.N.R. 108, Bologna, Italy.

Miller, D., Rose, K., 1994. A non-greedy approach to tree-structured clustering. Pattern Recognition Lett. 15, 683–690.

Whitley, D., Starkweather, T., Fuguay, D. 1989. Scheduling problems and traveling salesman: The genetic edge recombination operator. In: Proc. of the Third Int. Conf. on Genetic Algorithms and their Application, pp. 133–140.