ELSEVIER

January 1995

Pattern Recognition Letters

Pattern Recognition Letters 16 (1995) 89–96

Topological clustering of maps using a genetic algorithm

Dario Maio [a,*], Davide Maltoni [b], Stefano Rizzi [a]

[a] *DEIS - Facoltà di Ingegneria, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy*
[b] *Corso di Laurea in Scienze dell'Informazione, Università di Bologna, Sede di Cesena, Italy*

Received 8 July 1994; revised 9 August 1994

Abstract

This paper presents a genetic approach to the problem of map topological clustering. Maps are symbolically represented as graphs whose vertices are landmarks in the environment. Clustering is performed according to a fitness function which takes functional requirements into account.

Keywords: Clustering; Fitness function; Genetic algorithms; Learning; Maps

1. Introduction

The problem of clustering assumes a significant role in a variety of research areas ranging from pattern recognition to computer vision. In the field of autonomous agents, an interesting application of clustering arises from the wish to emphasize the topological characteristics of the environment maps, together with the need for decomposing path-planning tasks in order to reduce their complexity (Nitzan, 1985).

In our work we consider the case in which agents are given no a priori topological or metric description of the environment, so that they must learn it on-line by interpreting sensor data. Meta-knowledge of typical sensor patterns in the environment enables recognition of *landmarks* through a sensor-based classification algorithm.

Pursuing a hybrid approach to knowledge representation, in (Maio and Rizzi, 1994) we have proposed a layered architecture to represent environmental knowledge. On the symbolic layer, the environment map is represented by a graph whose vertices are landmarks and whose edges are *routes*, that is, feasible inter-landmark paths. Clustering allows for the symbolic representation of the environment to be distributed over different abstraction levels. At each level, clusters are represented by connected graphs; the connectivity constraint is necessary in order to make decomposition of path-planning tasks feasible. If no meta-knowledge for clustering is available, aggregation must be based on topological and metric criteria. In (Maio and Rizzi, 1993) we have presented a heuristic algorithm called *clustering by discovery* for topological clustering in a map being learned by exploration as the agent moves within the environment. Since at each exploration step new data may be acquired, clustering by discovery is an example of clustering for time incremental data (Chaudhuri, 1994).

In this paper we define a fitness function which allows for evaluating a clustering with respect to different topological and metric criteria, and propose a genetic algorithm which determines a near-optimal clustering on a given map by maximizing its fitness.

* Corresponding author. Email: dario@deis64.cineca.it

SSDI 0167-8655(94)00069-7

The algorithm uses an encoding technique and genetic operators defined *ad hoc*. In general, the time complexity of genetic algorithms discourages their use for real-time applications; nevertheless, the genetic approach is essential in producing near-optimal solutions to be used as comparison terms for evaluating other heuristic approaches to clustering.

2. Graph formalism for map representation

Let V and E be, respectively, the sets of landmarks and routes experienced at a given time. We define to be *symbolic layer* or *map* the non-directed connected [1] graph $\mathcal{M}=(V, E)$ whose vertices and edges correspond, respectively, to landmarks and routes.

We will denote with $\mathbf{pos}(v)$ the vector representing the position of landmark v (for details on the metric of maps, see (Maio and Rizzi, 1992)), and with $[v \leftrightarrow v']$ the route connecting landmarks v and v'.

Given the map $\mathcal{M}=(V, E)$, we define a *clustering* on $\mathcal{M}$ as a partitioning $\xi=\{V_1, ..., V_p\}$ of V. We call *clusters* the p sub-graphs $\mathcal{C}_1=(V_1, E_1), ..., \mathcal{C}_p=(V_p, E_p)$, where

$$E_i=\{[v \leftrightarrow v'] \in E: v \in V_i \wedge v' \in V_i\}, \quad i=1, ..., p.$$

We call *cardinality* of a cluster $\mathcal{C}_i$ the number of vertices it contains. We define the *position* of $\mathcal{C}_i$ as

$$\mathbf{pos}(\mathcal{C}_i)=\frac{1}{n_i}\sum_{vj \in V_i}\mathbf{pos}(v_j)$$

where n_i is the cardinality of $\mathcal{C}_i$.

Though $\mathcal{M}$ is connected, one or more clusters induced on $\mathcal{M}$ by a given clustering ξ may be non-connected. We will regard as legal only the clusterings whose clusters are connected.

3. Fitness function

In (Maio and Rizzi, 1993) we identified the set of clustering requirements summarized below.

- *Visibility.* In order to ensure a good intra-cluster mobility to the agent, a bound ρ on the maximum radius of clusters should be placed.
- *Parallel efficiency.* Clustering can be used to decompose planning algorithms following a *divide-et-impera* policy. To this end, all clusters should have the same cardinality.
- *Predictability.* High cluster cardinality leads to high management complexity; on the other hand, low cardinality strongly reduces the effectiveness of decomposing methods. Hence, the average cardinality of clusters should be equal to a given value η.
- *Homogeneity.* Clustering should reveal the topological features of the map. Hence, density of vertices should be homogeneous within each cluster.
- *Regularity.* Irregularly-shaped clusters cause decomposition techniques to generate non-optimal solutions. Therefore, clusters should be convex and regular.

We formalize these requirements by defining, for a clustering ξ on map $\mathcal{M}$, a fitness function f as a weighted sum of five components:

$$f(\xi, \rho, \eta)=\alpha_1 f_{\mathrm{vis}}(\xi, \rho)+\alpha_2 f_{\mathrm{par}}(\xi)+\alpha_3 f_{\mathrm{pre}}(\xi, \eta)+\alpha_4 f_{\mathrm{hom}}(\xi)+\alpha_5 f_{\mathrm{reg}}(\xi)$$

where $0 \leqslant \alpha_i \leqslant 1$ for $i=1, ..., 5$ and $\sum_{i=1}^{5}\alpha_i=1$. Each component $f_i(\,)$ expresses the degree to which ξ meets one of the criteria listed above, and ranges from 0 (no adherence) to 1 (maximum adherence).

4. Optimization techniques for map clustering

The definition of a fitness function to evaluate how "good" a clustering is, turns the clustering problem into an optimization problem; nevertheless, the connectivity constraint imposed on clusters makes the application of operational research techniques to obtain optimal solutions very complex. Further difficulty arises from the impossibility of knowing a priori the optimal number of clusters and their approximate location. Thus, we must focus on heuristic techniques.

Hill-climbing is a local search procedure which, in its deterministic formulation, chooses at each step, among all possible moves in the state space, the one leading to the state with highest fitness. In the case of topological clustering, enumerating all possible moves in the state space means considering (and evaluating) all clusterings which can be obtained from a given

[1] The graph is necessarily connected if its vertices and arcs are learned by exploration.

clustering by applying one of three operators:

(a) move vertex $v \in \mathscr{C}_i$ to cluster $\mathscr{C}_j$ (can be applied only if there exists a route $[v \leftrightarrow v']$ such that $v' \in \mathscr{C}_j$ and if deleting v from $\mathscr{C}_i$ does not violate the connectivity constraint on $\mathscr{C}_i$);

(b) split cluster $\mathscr{C}_i$ in two without violating the connectivity constraint on both resulting clusters;

(c) merge clusters $\mathscr{C}_i$ and $\mathscr{C}_j$ (can be applied only if there exists a route $[v \leftrightarrow v']$ such that $v \in \mathscr{C}_i$ and $v' \in \mathscr{C}_j$).

If the number of clusters were fixed a priori, only operator (a) should be employed and enumeration would be feasible. Since also the number of clusters must be optimized, so that all three operators must be tried, the cost of enumeration becomes prohibitive. In particular, as to operator (b), note that a given cluster can be legally split in several ways; on the other hand, most splitting (those producing two clusters whose numbers of vertices are strongly unbalanced) would never be executed since they would cause fitness to decrease.

Some variants of hill-climbing have been proposed in order to avoid getting trapped into local maxima. The most interesting are iterative hill-climbing, which still suffers from the expense of enumerating all the adjacent points, and stochastic hill-climbing, which avoids enumeration and can be considered as a simulated annealing with fixed temperature (Ackley, 1987).

Simulated annealing is an iterative probabilistic algorithm in which a deterioration of the objective function in an iteration step may be accepted with a given probability (Kirkpatrick et al., 1983). This technique, derived from statistical mechanics, has been applied successfully to a variety of optimization problems; it is, however, not always trivial determining an effective method for generating configurations and selecting an efficient annealing schedule (Davis and Steenstrup, 1987). The application of simulated annealing to optimize clustering fitness raises some interesting issues. First of all, our fitness function may be very uneven, with the global maximum lying on top of a very narrow peak. This is especially true for strongly dishomogeneous maps, where reassigning even only one vertex in the optimal solution may cause the homogeneity criterion to decrease dramatically. In order to deal with uneven objective functions, simulated annealing requires a very slow cooling rate, thus increasing the time necessary to obtain a solution. Secondly, merging vertex reassignment and cluster creation/deletion in the definition of adjacency between clustering states would require an accurate tuning. In (Rose et al., 1990) clustering optimization is performed through a deterministic annealing technique in which the cost function is deterministically optimized at each temperature. This approach cannot be adopted in topological clustering since, being the number of clusters not known a priori, deterministic optimization at a given temperature is not feasible.

A genetic algorithm (GA) is a stochastic algorithm used in optimization problems, based on an evolutionary biological model (Goldberg, 1989). The main strength of the genetic approach with respect to traditional search techniques lies in its implicit parallelism. In fact, instead of evaluating and improving a single solution, a GA considers a *population* of solutions at the same time; thus, the search is directed by gathering information from different points in the state space. In particular, by adopting a genetic approach, the problem of generating adjacent configurations in the state space is demanded to a proper encoding scheme and *ad hoc* genetic operators.

5. Applying genetic algorithms to map topological clustering

Different alternatives for encoding the problem of object partitioning have been proposed in the GA literature (Bhuyan et al., 1991; Jones and Beltramo, 1990). 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 choice of the *ad hoc* encoding scheme and genetic operators described in the following subsections.

Encoding scheme

The encoding scheme adopted resembles that proposed in (Davis, 1985). Given a map $\mathscr{M} = (V, E)$ in-

cluding n vertices, each chromosome in the population is represented by a string of length n consisting of a permutation of the first n integers, where each integer references a vertex in the map, and a separator splitting the string in two. We call the two substrings to the left and right of the separator *seed* and *growth*, respectively.

Every chromosome c maps into exactly one consistent solution to the clustering problem by means of a decoding procedure in which first each vertex in seed(c) is used to initialize a different cluster, and then the vertices in growth(c) are progressively added to the clusters created. If we denote with $\Delta(\mathbf{pos}(e), \mathbf{pos}(e'))$ the Euclidean distance between the positions of entities e and e' (vertices or clusters), the decoding procedure of c may be sketched as follows.

```
decode(c)
{S=∅;
 for each v_i∈seed(c)
 {create new cluster C_i=({v_i}, ∅);
  S=S∪{C_i};
  }
 orderly copy vertices from growth(c) to circular list
 L;
 while length(L)>0
 {v_j=next(L);
  H={C_i∈S: ∃[v_j↔v_k]∈M, v_k∈C_i};
  if H≠∅
  {find C_best∈H:
     ∀C_i∈H (Δ(pos(v_j), pos(C_best))
              ≤Δ(pos(v_j), pos(C_i)));
   add v_j to C_best;
   for each [v_j↔v_k]∈M: v_k∈C_best
     add [v_j↔v_k] to C_best;
   delete v_j from L;
   }
 }
 return ξ={V_i: C_i=(V_i, E_i), C_i∈S};
}
```

Due to the connectivity constraint, it may happen that the vertices in the growth cannot be orderly assigned to clusters. When a vertex cannot be assigned to any cluster, the algorithm momentarily leaves it apart and tries again to assign it at the next iteration. Since $\mathcal{M}$ is a connected graph, we are guaranteed that all vertices are assigned in a finite number of steps.

Fig. 1 shows the clustering associated to a sample chromosome on a map including 15 vertices.

It is remarkable that, by adopting this encoding/decoding technique, all possible strings consisting of a permutation of the first n integers and a separator position between 1 and n represent a consistent solution to the clustering problem. The number of clusters, p, is equal to the length of the seed, hence it is determined by the separator position.

Genetic operators

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. Two copies of the chromosome having maximum fitness in P are always included in P_{next}; the remaining $z-2$ chromosomes of P_{next} are chosen randomly from P with probability proportional to their fitness.

As a matter of fact, selection is not carried out using the fitness function $f()$, but a scaled function $f'()$. *Scaling* is commonly used in GAs in order to normalize the fitness range in the different phases of evolution. In fact, the first generations usually contain few "good" chromosomes and a lot of "average" and "bad" ones; since good solutions quickly overwhelm the others, the advanced generations consist of chromosomes having similar (high) values of fitness, so that mediocre solutions have approximately the same probability as the best ones of being selected for reproduction. Linear scaling (Goldberg, 1989), which we adopted in our GA, uses a linear transformation

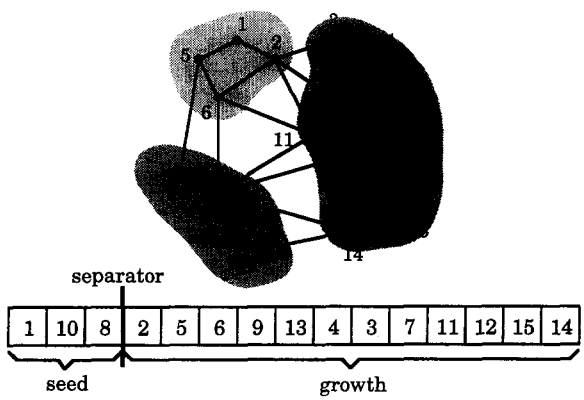


Fig. 1. Clustering associated to a sample chromosome on a map including 15 vertices.

$$f'() = k_1 f() + k_2$$

to adjust f at each generation.

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 entails the two phases described below.

The first is a partial crossover of the seeds and generates offspring chromosomes in which the number of clusters is the average of that of their parents. Let c_1 and c_2 be the parent chromosomes, and c'_1 and c'_2 the offspring chromosomes. The position of the separator in the offspring is

$$\text{sep}(c'_1) = \text{sep}(c'_2) = (\text{sep}(c_1) + \text{sep}(c_2))/2$$

if $\text{sep}(c_1) + \text{sep}(c_2)$ is even;

$$\text{sep}(c'_1) = \lfloor (\text{sep}(c_1) + \text{sep}(c_2))/2 \rfloor ,$$

$$\text{sep}(c'_2) = \text{sep}(c'_1) + 1$$

otherwise.

The compositions of the offspring seeds are determined by the following constraints:

$$\text{seed}(c'_1) \cup \text{seed}(c'_2) \subseteq \text{seed}(c_1) \cup \text{seed}(c_2) ,$$

$$\text{seed}(c_1) \cap \text{seed}(c_2) \subseteq \text{seed}(c'_1) ,$$

$$\text{seed}(c_1) \cap \text{seed}(c_2) \subseteq \text{seed}(c'_2) .$$

The seed of each offspring consists of all the vertices belonging to the intersection of the parents' seeds plus other vertices taken randomly from the union of the parents' seeds. Since one or more vertices not belonging to the intersection of the parents' seeds may be inserted in both offspring, one or more vertices of the parents' seeds may not be present in any of the offspring's seeds. Each chromosome must contain a consistent permutation, hence, including a vertex in the seed really means inverting the positions of two vertices in the chromosome. An example of seed crossover is shown in Fig. 2.

The second crossover phase is substantially equivalent to the *Partially Matched Crossover* (PMX) (Whitley et al., 1989) applied to the growths. First, two positions a and b of the growths of c'_1 and c'_2 are chosen randomly from the interval $[\text{sep}(c'_2), n]$. The substrings of the growths of c'_1 and c'_2 enclosed between a and b are called *matching sections*, and include $(b-a)$ vertices. Secondly, the matching sections of c'_1 and c'_2 are exchanged by inverting vertices as necessary. Occasionally, one of the vertices to be replaced may belong to the seed; in this case the replacement is not executed, in order to prevent vertices from crossing the separator and isolate the seed from the growth. Fig. 3 shows an example of growth crossover.

Each chromosome c generated by reproduction has a given probability (*mutation* rate) of mutation. In order to keep vertex reassignment and cluster creation/deletion conceptually separate and mutually independent, we adopt two different mutation operators which can be applied separately or jointly on the same chromosome.

The first (*permeability*) modifies the number of clusters by moving the separator one position backwards or forwards. If the separator is moved from $\text{sep}(c)$ to $\text{sep}(c)-1$, the vertex in position $\text{sep}(c)$ becomes part of the growth; otherwise the vertex in position $\text{sep}(c)+1$ becomes part of the seed. In order to avoid the vertices in positions $\text{sep}(c)$ and $\text{sep}(c)+1$ being the only candidates to this mutation, the vertex involved is inverted with the one in a position chosen randomly within the same substring before the separator is moved. By doing so, all the vertices in c have the same probability of being moved from one substring to the other. An example is shown in Fig. 4(a).

The second mutation operator (*scrambling*) leaves the seed unchanged, but inverts the vertices in two random positions of the growth (Fig. 4(b)). since our decoding technique considers the order in which vertices appear in the growth, this operator alters the structure of clusters.

6. Experimental tests and conclusion

We carried out several simulations on maps with different topologies using populations of size 20, since we experimented that the advantages deriving from a larger population do not compensate for the increased computational complexity. The starting population consists of a family of z strings generated randomly. The separator position is chosen randomly in the interval $[1, n]$; the distribution function used has

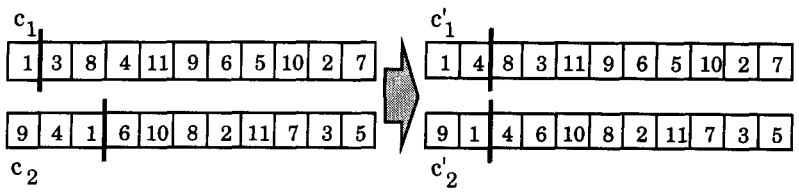


Fig. 2. An example of seed crossover. Vertex 1 is common to both parent seeds, hence it must be included in the seed of both offspring chromosomes. For each offspring chromosome, the vertex for filling the seed is chosen randomly from the set {9,4}. Inserting 4 in the seed of c'_1 entails inverting it with 3; inserting 1 in the seed of c'_2 entails inverting it with 4.

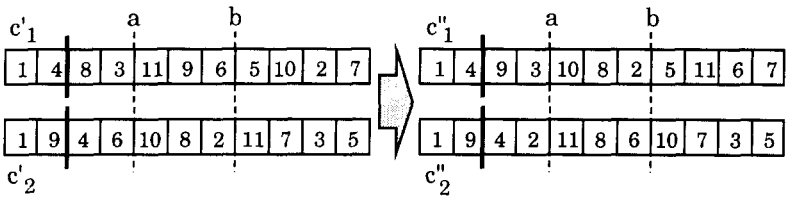


Fig. 3. Example of growth crossover. Exchanging the matching sections of c'_1 and c'_2 requires inverting vertices 11 and 10, 9 and 8, 6 and 2 in c'_1, and vertices 10 and 11, 8 and 9, 2 and 6 in c'_2. Vertices 8 and 9 are not inverted since 9 belongs to the seed of c'_2.

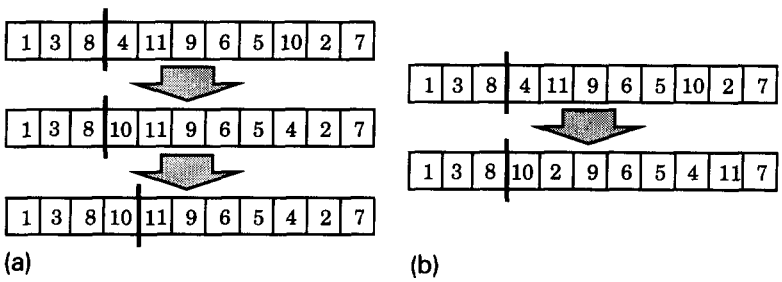


Fig. 4. Example of application of the permeability (a) and scrambling (b) operators.

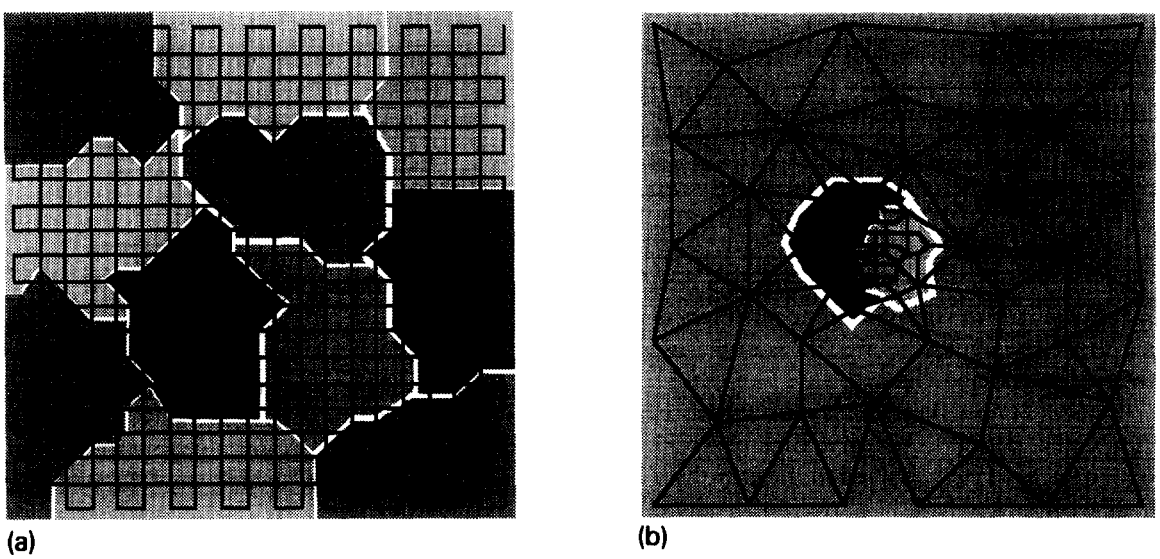
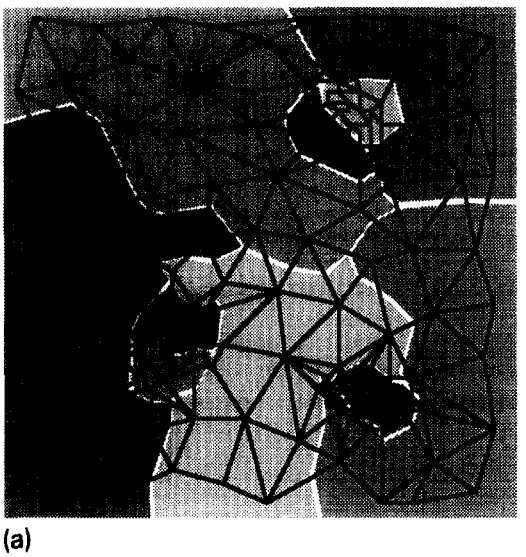


Fig. 5. Clusterings obtained on two maps: (a) a regular square-meshed map (values used for parameters: $\rho=0.4$, $\eta=35$; fitness obtained: $f=0.99$) and (b) a typical urban map ($\rho=0.5$, $\eta=30$; $f=0.86$). The maximum radius ρ is expressed as a fraction of the map global size.



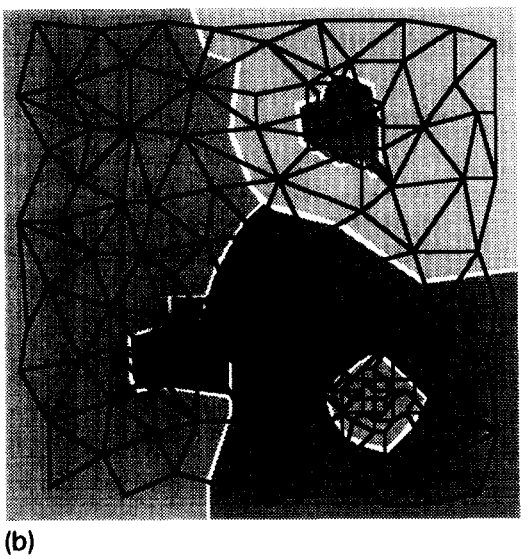


Fig. 6. Clusterings obtained on the same map with different values of ρ and η. (a) $\rho=0.5$, $\eta=20$ (fitness obtained: $f=0.88$); (b) $\rho=0.6$, $\eta=35$ ($f=0.88$).

a peak in position n/η, corresponding to the desired number of clusters.

Fig. 5 shows the clusterings obtained on two sample maps, and reports the corresponding fitness values. In particular, the clustering in Fig. 5(a) points out the good behaviour of the algorithm in terms of adherence to parallel efficiency and regularity; Fig. 5(b) shows how discontinuities in the density of vertices are revealed.

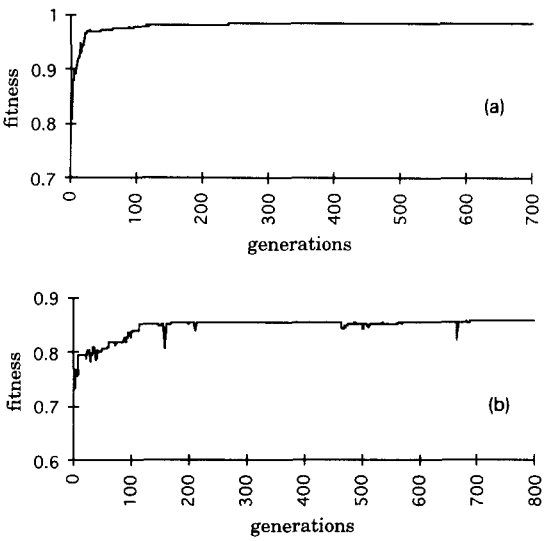


Fig. 7. Fitness during evolution for the clusterings in Figs. 5(a) and 5(b). The fitness displayed for each generation is that of the best chromosome in the current population.

In some cases the five requirements outlined in Section 3 may contrast with each other so that the optimal clustering expresses a trade-off between them; the weights α_i used in the fitness function as well as the values chosen for parameters ρ and η, are crucial in determining the trade-off point. Fig. 6 shows how clustering is influenced by ρ and η.

Though experimental evidence suggests that GAs converge to the optimum in most cases, no formal demonstration has been given yet in the literature. The diagrams in Fig. 7 show how the fitness varied during the evolutionary process corresponding to the clusterings in Fig. 5. Even after a relatively small number of generations (100–200), the solutions yielded are usually very good.

The main drawback of a GA is that, due to its time complexity, it can hardly be used in real-time applications. In autonomous agents clustering is carried out using exploration, so that new data to be immediately clustered are acquired continually. At present we are studying how, when a new vertex v is discovered during exploration and clustering must be recalculated, the starting population can be derived by properly extending with v the final population at the previous step. Since it seems reasonable to assume that in most cases the local perturbation caused by discovering a new vertex will not greatly alter the near-

optimal clustering, we expect that the derived starting population will be exceptionally "good", so that the number of generations necessary for convergence may greatly decrease.

References

Ackley, D.H. (1987). An empirical study of bit vector function optimization. In: L. Davis, Ed., *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann, Los Altos, CA, 170–204.

Bhuyan, J., V. Raghavan and V. Elayavalli (1991). Genetic algorithms for clustering with an order representation. *Proc. 4th Internat. Conf. on Genetic Algorithms and their Application*, 408–415.

Chaudhuri, B.B. (1994). Dynamic clustering for time incremental data. *Pattern Recognition Lett.* 15, 27–34.

Davis, L. (1985). Job shop scheduling with genetic algorithms. *Proc. First Internat. Conf. on Genetic Algorithms*, 136–140.

Davis, L. and M. Steenstrup (1987). Genetic algorithms and simulated annealing: an overview. In: L. Davis, Ed., *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann, Los Altos, CA, 1–11.

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

Jones, D.R. and M.A. Beltramo (1990). Solving partitioning problems with genetic algorithms. Research Report 48090-9055, General Moters Research Laboratories, Warren, MI.

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi (1983). Optimization by simulated annealing. *Science* 220, 671–680.

Maio, D. and S. Rizzi (1992). Clustering by discovery on maps. *Pattern Recognition Lett.* 13 (2), 89–94.

Maio, D. and S. Rizzi (1993). Map learning and clustering in autonomous systems. *IEEE Trans. Pattern Anal. Machine Intell.* 15 (12), 1286–1297.

Maio, D. and S. Rizzi (1994). A hybrid approach to path planning in autonomous agents. *Proc. 2nd Internat. Conf. on Expert Systems for Development*, Bangkok, 222–227.

Nitzan, D. (1985). Development of intelligent robots: achievements and issues. *IEEE J. Robotics and Automation* 1 (1), 3–13.

Rose, K., E. Gurewitz and G. Fox (1990). A deterministic annealing approach to clustering. *Pattern Recognition Lett.* 11 (9), 589–594.

Whitley, D., T. Starkweather and D. Fuguay (1989). Scheduling problems and traveling salesman: the genetic edge recombination operator. *Proc. Third Internat. Conf. on Genetic Algorithms and their Application*, 133–140.