Exploratory Data Analysis with Interactive Evolution

Sergey Malinchik and Eric Bonabeau

Icosystem Corporation, 10 Fawcett St., Cambridge, MA 01238, USA
{sergey, eric}@icosystem.com
http://www.icosystem.com

Abstract. We illustrate with two simple examples how Interactive Evolutionary Computation (IEC) can be applied to Exploratory Data Analysis (EDA). IEC is particularly valuable in an EDA context because the objective function is by definition either unknown *a priori* or difficult to formalize. The first example involves what is probably the simplest possible transformation of data: linear projections. While the concept of linear projections is simple to grasp, in practice finding the appropriate two-dimensional projection that reveals important features of high-dimensional data is no easy task. We show how IEC can be used to quickly find the most informative linear projection(s). In another, more complex example, IEC is used to evolve the "true" metric of attribute space. Indeed, the assumed distance function in attribute space strongly conditions the information content of a two-dimensional display of the data, regardless of the dimension reduction approach. The goal here is to evolve the attribute space distance function until "interesting" features of the data are revealed when a clustering algorithm is applied.

Keywords: Interactive evolutionary computation, data mining, exploratory data analysis.

1 Introduction

At a time when the amount of data that is potentially available for analysis and exploitation is increasing exponentially in all fields and industries, the ability to quickly explore gigantic databases for valuable patterns is becoming a crucial competitive advantage. The problem is: what should one look for? Exploratory Data Analysis (EDA) [1,2] is the art of exploring data without any clear *a priori* ideas of what to look for. Most techniques used in EDA are *interactive* and *visual*. There are a number of display techniques that take care of the visualization aspect. As the dimensionality of the data (that is, the number of attributes) increases, informative low-dimensional projections of the data are necessary (most often in two dimensions). Families of techniques such as multi-dimensional scaling (MDS) [3] are used to generate such projections. While there exist powerful visualization techniques, the interactivity of EDA is often *ad hoc* and best characterized as tinkering. That is because in EDA the notion of "interestingness" of a particular display or pattern is, by definition, difficult to formalize. Thus designing data mining patterns, filters, projection algorithms, clustering algorithms and other search models that will produce "interesting" results when applied to a dataset is a difficult problem [2].

If one assumes that interesting patterns will be recognized when they are discovered even though they could not be formulated ahead of time, a technique originally developed to generate "interesting" images and pieces of art [4,5,6,7], Interactive Evolutionary Computation (IEC), can be used. Although IEC has been applied to some data mining problems in the past (see [8] for a review), it has never been described as a canonical tool to perform EDA. This paper is an attempt to show how pervasive the technique might become. IEC is a directed search evolutionary algorithm which requires human input to evaluate the fitness of a pattern (here, the fitness might be how interesting a detected pattern is) and uses common evolutionary operators such as mutation and crossover [9] to breed the individual-level rules that produced the fittest collective-level patterns. IEC combines computational search with human evaluation [8,10].

2 The Problems

We illustrate the use of IEC in EDA with two simple examples using the same five-dimensional real-valued dataset described in section 2.1. In the first example, IEC is used to evolve two-dimensional linear projections of the dataset, a simple but surprisingly powerful aid in exploring data; the linear projections are evaluated according to how much insight they provide the user about the data. In the second example, IEC is used to evolve the distance function in attribute space so as to produce the most compelling clusters using a simple parametric clustering algorithm.

2.1 The Dataset

All experimental results described in this paper were obtained using the same synthetic, five-dimensional real-valued dataset. The five dimensions are represented as (X_1, X_2, X_3, X_4, X_5). The dataset contains 24,000 points in 5 separate clusters. The clusters are generated in the following way: each of the five coordinates (x_1, x_2, x_3, x_4, x_5) of each data point is generated independently from a Gaussian distribution whose mean is equal to the corresponding coordinate of the center of the cluster that point belongs to, and whose standard deviations, listed below, reflect the cluster's elongation or compression along certain dimensions. Some of the planar projections of some of the clusters are then rotated by a certain angle.

The five clusters are defined as follows:

- Cluster #1:
  3000 points, center located at: (650, 255, 540, 500, 300), standard deviations for each coordinate: (50, 40, 20, 200, 60); (X_3, X_4) coordinates rotated by 20 degrees in the (X_3, X_4) plane around point (x_3=0, x_4=0).

- Cluster #2:
  6000 points, center located at: (450, 200, 400, 500, 300), standard deviations for each coordinate: (50, 70, 150, 40, 60); (X_3, X_4) coordinates rotated by 30 degrees in the (X_3, X_4) plane around point (x_3=0, x_4=0).
- Cluster #3:
  7000 points, center located at: (400, 400, 500, 200, 300), standard deviations for each coordinate: (90, 50, 160, 40, 20); (X_3, X_5) coordinates rotated by 90 degrees in the (X_3, X_5) plane around point (x_3=0, x_5=0).
- Cluster #4:
  4000 points, center located at: (600, 550, 350, 300, 540), standard deviations for each coordinate: (80, 70, 20, 40, 80).
- Cluster #5:
  4000 points, center located at: (350, 580, 600, 340, 600), standard deviations for each coordinate: (60, 70, 40, 40, 60)

Although the dataset is simple, discovering its cluster properties (# of clusters, locations, shapes) without any *a priori* knowledge is a non-trivial task.

2.2 Evolving Linear Projections

The general problem here is to find a linear transformation from a N-dimensional space (also called attribute space) onto a two-dimensional representation. For example, with real-valued attributes in N dimensions, the transformation would be from R^N to R^2. Each point x is characterized by N coordinates or attributes: $x = (x_1, x_2 .. x_N)$. x is projected onto a point with coordinates (x', y') in a two-dimensional space, via the following transformation:

$$x' = \sum_{i=1}^{N} \alpha_i x_i \ , \qquad (1)$$

$$y' = \sum_{i=1}^{N} \beta_i x_i \ , \qquad (2)$$

where $-1 \le \alpha_i, \beta_i \le 1$ for all i=1,… N and

$$\sum_{i=1}^{N} \alpha_i^2 = 1 \, , \qquad (3)$$

$$\sum_{i=1}^{N} \beta_i^2 = 1 \, . \qquad (4)$$

The transformation's parameters $\alpha = (\alpha_1, \ldots, \alpha_N)$ and $\beta = (\beta_1, \ldots, \beta_N)$ belong to the unit sphere in R^N. The problem here is very simple: what are the most informative linear projections from the user's perspective? The problem in defining a fitness function for "most informative" is that we don't know ahead of time what it means. Although it is probably the simplest possible EDA problem one can think of, it is of practical importance when one wants to display very high-dimensional data using a small number of two-dimensional projections. Such projections obviously destroy a potentially significant amount of information but two-dimensional displays constitute the most commonly used way of visualizing data. Finding a good set of two-dimensional projections will help the user tremendously.

2.3 Evolving Distance Functions in Attribute Space

Clustering is a useful and commonly used technique in EDA. The goal of clustering is to compress the amount of data by categorizing or grouping similar items together [11]. There are many different clustering algorithms. However, the clusters resulting from the application of one specific clustering algorithm to a data set are heavily dependent on the distance function assumed in attribute space. That distance function is rarely questioned and even more rarely an object of study. For example, when dealing with real-valued data, it is often implicitly assumed without further examination that the relevant distance function is the Euclidian distance or L_p (L_1 = city-block distance; L_2 = Euclidian distance; L_∞ = max norm). But that might not appropriately reflect the potentially complex structure of attribute space. For example, the use of the Euclidian distance in attribute space for the dataset described in Section 2.1 does not lead to the extraction of the data's clusters because of the elongation and compression of the clusters along certain dimensions. The problem here is to discover a distance function in attribute space that contains some of the fundamental properties of that space. To do so we apply a simple parametric clustering algorithm (K-means, well-suited to the globular clusters of our dataset) to the data using a variety of distance functions and we then examine the resulting clusters; the distance function is evolved until the clusters look "right" or provide valuable information.

A commonly used clustering method is K-means clustering [11], which is a least-squares partitioning method allowing users to divide a collection of objects directly into K disjoint clusters. The energy function E_K that is minimized in K-means is the sum of the squared distances between each data item x_m and the nearest cluster centroid:

$$E_K = \sum_m \| x_m - c(x_m) \|^2 \qquad (5)$$

where $c(x_m)$ is the centroid that is closest to x_k. The algorithm starts by initializing a set of K cluster centroids denoted by c_i, $i=1,\ldots K$. Data points are examined sequentially in a random order and each point is assigned to the nearest centroid. The positions of the centroids are then adjusted iteratively by re-computing the centroids to move them closer to the set of points that belong to the corresponding cluster. Several passes through data are performed until every data point is consistently assigned to one cluster, that is, until the assignment of points to clusters is stable.

K-means is characterized by simplicity and computational efficiency but it is sensitive to cluster shapes. In particular, it fails when clusters are too close to one another. When clusters are strongly anisotropic (for example, elongated or compressed along certain dimensions) it is helpful to define the distance function in such a way that it counterbalances cluster asymmetry, thereby revealing the structure of attribute space. The family of weighted Euclidean distance functions in R^N is explored using IEC. A distance function in that family is given by:

$$d_w\left(x^p, x^q\right) = \sqrt{\sum_{i=1}^{N} w_i (x_i^p - x_i^q)^2} \tag{6}$$

where x^p and x^q are two points in R^N, and $\boldsymbol{w}=(w_1, w_2, \ldots, w_N)$ is the weight vector that characterizes that distance function, with $0 \le w_i \le 1$, $i=1,\ldots,$ N, and

$$\sum_{i=1}^{N} w_i = 1 \,. \tag{7}$$

The object of the IEC-based search is to evolve $\boldsymbol{w}$ so as to extract valuable information from the clusters.

3 Interactive Evolution

3.1 Overview of the Search Mechanism

The IEC search method works as follows. A small initial population of solutions is generated, where a solution is a linear projection (Example 1) or a distance function (Example 2).

- The resulting two-dimensional representations are computed, generated and displayed to a human observer.
- The observer selects the displays that are the most interesting –the fittest individuals in the population according to whatever set of objective and subjective criteria the observer may be using, or assigns a fitness value to each display.
- A new population (new generation) of solutions is generated by applying mutation and crossover operators to the solutions that correspond to the previous generation's fittest displays.
- In addition to the offspring and mutated versions of those solutions, randomly generated solutions are injected into the population to ensure diversity.
- The new population's two-dimensional representations are then calculated and the results displayed to the observer, and so forth.
- This procedure is iterated until interesting displays emerge from the search, pointing toward corresponding linear projections or distance functions.

The user interface is a critical component of the method. The observer evaluates solutions based on visual inspection of their two-dimensional representations. The visual interface is shown in Figure 1. Obviously this method can only work if the population size is kept small and if interesting solutions emerge after a reasonably small number of generations.

3.2 Genetic Algorithm

The evolutionary mechanism is the same in Example 1 and Example 2.

- A simple real-valued representation is used to encode the genotype. In Example 1, the genotype is a juxtaposition of 2*N* real-valued genes: $(\alpha_1,\ldots,\alpha_N,\beta_1,\ldots,\beta_N)$, with *N*=5.

  In Example 2, the genotype is a juxtaposition of *N* real-valued genes $(w_1,\ldots,w_N)$ with *N*=5.
- Population size is equal to nine.
- Initial gene values are generated randomly from a uniform distribution over the allowed range.
- The user assigns fitness values to the various displays. The default value for all displays before user intervention is 0. In Example 2, the user can play with different views (two-dimensional projections onto coordinate planes) of the data before assigning a fitness value. After visual inspection of the dataset (using, for example, the linear projections evolved in Example 1), the user sets the number of clusters (K) and the tool randomly generates the starting positions of the cluster centroids in the original 5-dimensional space. Each of the nine solutions displayed represents the results of the application of the K-means clustering algorithm with a different distance function in attribute space, characterized by the weight vector ***w***.
- The results are displayed simultaneously to the user. The user can move back and forth and reassign fitness values.
- The user can select one or more displays to be part of an elite pool. Unless otherwise specified by the user, only one elite individual is selected by the algorithm, which is the one with the highest fitness.
- Clicking the mouse on the corresponding image user indicates his choice and the tool automatically sets the fitness function to the value of 5. Double clicking means the selection of the elite individual, which fitness function will be equal to 10. The user's choice is indicated on the screen by a green or yellow frame around ordinary selected individuals or elite ones, respectively.
- The mating pool is comprised of solutions with fitness larger than zero.
- The next generation is created in the following way: one elite solution remains unchanged. Four solutions are created from the ranked mating pool. Two offspring are created from two randomly selected parents in the mating pool by applying a random single point crossover. The last four solutions are created from single parents from the mating pool after application of a single point mutation. The new value of the mutated point is drawn from a normal distribution function whose mean is equal to the current gene value and whose standard deviation equal to one third of the gene value range.

4 Results

4.1 Example 1

The search is performed until the user finds a rich enough representation of the data in the form of a linear two-dimensional projection. It typically takes from 5 to 15 generations to reveal the internal structure of the dataset described in Section 2.1.

Figure 1 demonstrates the evolution of the projections of the five-dimensional dataset. Initial projections are very poor, providing little insight into the data.

A typical selection strategy by the user consists of selecting projections based on the visual improvement of some data substructure.

4.2 Example 2

The user performs the interactive search for distance functions in the same way as for Example 1, looking for valuable and consistent information.

Although the user may not know ahead of time what "valuable" information is contained in the data, there are obvious shapes he will be looking for in the search, for example ensembles of points that seem to belong to the same cluster when viewed

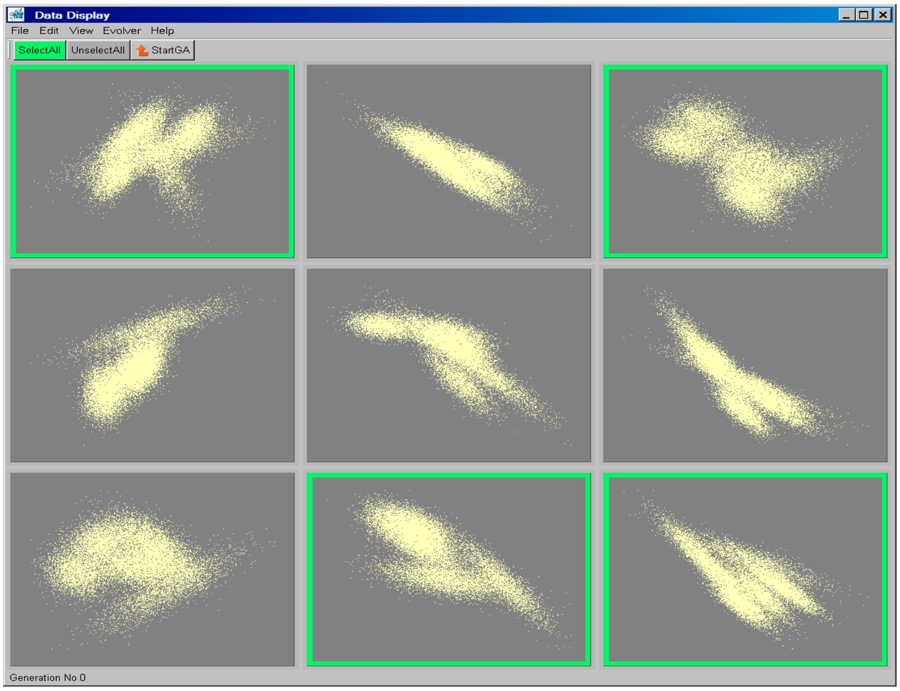

Fig. 1 (a)

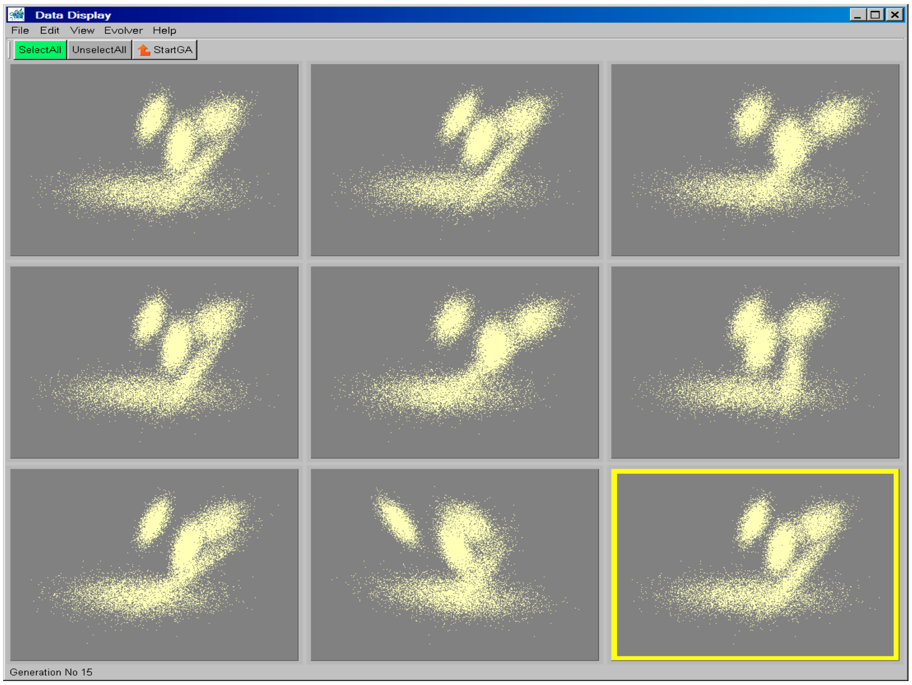

Fig. 1 (b)

Fig. 1. Example 1: User interface with nine solutions displayed. (a) Initial population. (b) Generation 15. Projection parameters: $\alpha_1 = 0.02$, $\alpha_2 = -0.04$, $\alpha_3 = 0.6$, $\alpha_4 = 0.08$, $\alpha_5 = -0.05$, $\beta_1 = -0.11$, $\beta_2 = -0.15$, $\beta_3 = -0.35$, $\beta_4 = 0.74$, $\beta_5 = 0.26$.

from different projections but are not assigned to the same cluster by the clustering algorithm, thereby suggesting a flaw in the distance function. Difficult clusters for a clustering algorithm such as K-means to map are for example elongated clusters. The user will therefore tend to favor distance functions that can map those clusters.

In Figure 2 we present nine displays resulting from the application of K-means clustering for randomly generated values of the weight vector $\boldsymbol{w}$ (Figure 2a), and nine displays obtained after 15 generations with interactively evolved $\boldsymbol{w}$ (Figure 2b). In both cases, the display uses the same specific projection of the dataset, obtained from the IEC run described in Example 1. A criterion that emerged as the main basis for the user's selection of the best distance functions is the presence of homogeneously colored clusters. If a cluster is not homogeneously colored or if it appears to be cut in the middle, the user easily detects this spatial inconsistency and assigns a low fitness value to such a solution. Although there is a range of values of w that yield a good clustering, all of the best solutions typically share the same properties: w_3, w_4 and w_5 have significantly lower values than w_1 and w_2, thereby indicating that the attribute space has anisotropic properties. A good solution arrived at with IEC is: $w_1 = 0.33$, $w_2 = 0.35$, $w_3 = 0.12$, $w_4 = 0.06$ and $w_5 = 0.12$.

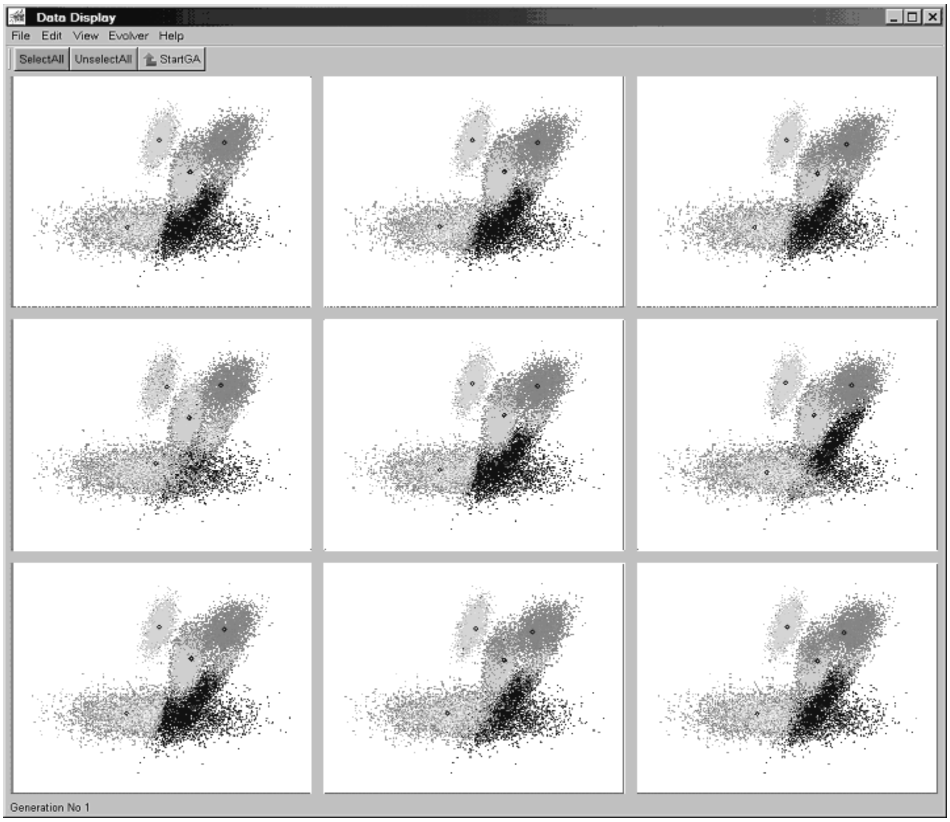

Fig. 2. (a)

5 Discussion

We have illustrated with two simple toy examples how IEC can assist in EDA and could in fact become a canonical EDA tool, helping to structure the user's intuition and insights in situations where the value of the results from a data mining algorithm is not known ahead of time and/or is difficult to formalize. There are obviously limitations to what can be achieved with IEC since it relies on small populations and a small number of generations [8]. One can however increase the potential of an IEC search by combining it with an automated pre-processing or filtering of the solutions when some objective, formalized criteria and constraints are known, so that only pre-processed solutions are presented to the user. Another useful extension would be to give the user the ability to explicitly select sub-structures or sub-modules of the display that seem promising. The specific genetic algorithm used in this paper can also be improved, for example by introducing a self-adaptable mutation step size.

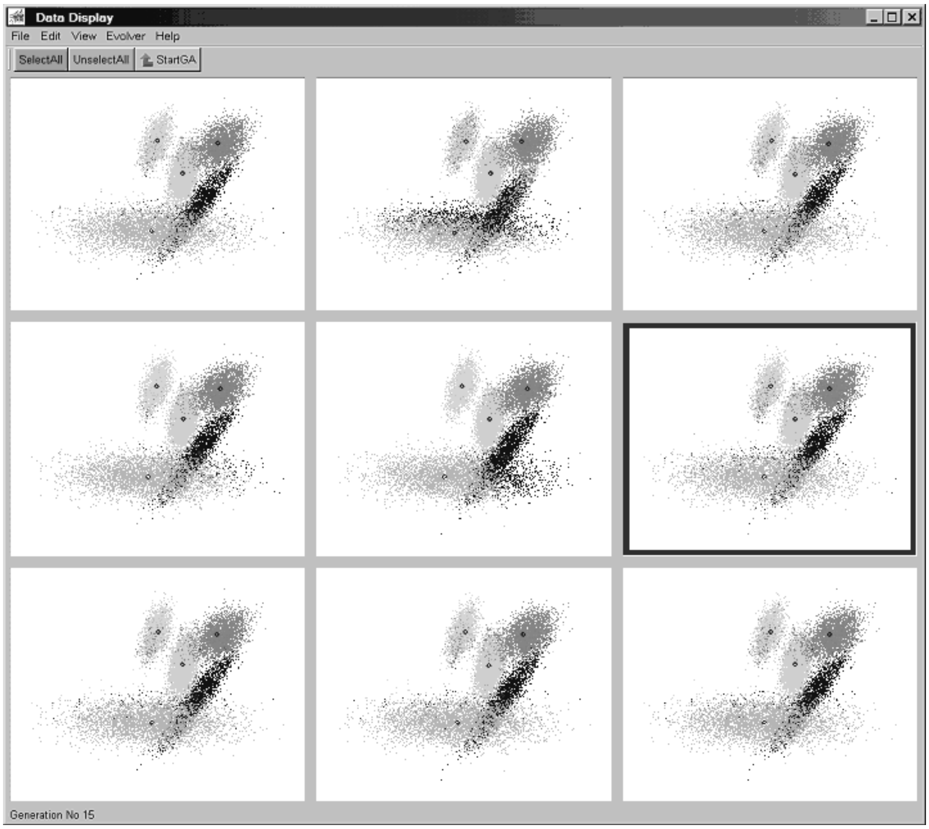

Fig. 2 (b)

Fig. 2. Example 2: User interface with nine solutions displayed. Black circles represent cluster centers. Different colors represent different clusters. (a) Initial population. (b) Generation 15.

References

1. Tukey, J. W. 1977. *Exploratory Data Analysis*. Addison Wesley, Reading, MA.
2. Hand, D., Mannila, H. & Smyth, P. 2001. *Principles of Data Mining*. MIT Press, Cambridge, MA.
3. Cox, T. F. & Cox, M. A. A. 1994. *Multidimensional Scaling*. Chapman and Hall, London.
4. Dawkins. R. 1987. *The Blind Watchmaker*. W. W. Norton, New York.
5. Sims, K. 1991. Artificial evolution for computer graphics. *Computer Graphics* **25**: 319-328.
6. Sims, K. 1992. Interactive evolution of dynamical systems. Pages 171-178 in: *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* (F. J. Varela & P. Bourgine, eds.), MIT Press, Cambridge, MA.
7. Sims, K. 1993. Interactive evolution of equations for procedural models. *Vis. Comput*. **9**: 446-476.

8. Takagi, H. 2001. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proc. IEEE* **89**: 1275-1296.
9. Michalewicz, Z. 1999. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer
10. Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G. & Cogne, F. 2000. Complex three-dimensional architectures grown by simple agents: an exploration with a genetic algorithm. *BioSystems* **56**: 13-32.
11. Jain, A. K. & Dubes, R. C. 1988. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ.