# Evolutionary Algorithms with Clustering for Dynamic Fitness Landscapes

Susana Esquivel, Victoria Aragón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional*
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis, Argentina
{esquivel,vsaragon}@unsl.edu.ar
Workshop de Agentes y Sistemas Inteligentes

**Abstract**

Interest on dynamic multimodal functions risen over the last years since many real problems have this feature. On these problems, the goal is no longer to find the global optimal, but to track their progression through the space as closely as possible. This paper presents three evolutionary algorithms for dynamic fitness landscapes. In order to mantain diversity in the population they use two clustering techniques and a macromutation operator. Besides, this paper compares two crossover operators: arithmetic and multiparents two points, respectively. Effectiveness and limitations of each algorithm are discuss and analyzed.

**Keywords:** Dynamic Multimodal Functions, Evolutionary Algorithms, Clustering Algorithms, Macromutation.

*LIDIC is financed by Universidad Nacional de San Luis and ANPCyT (Agencia Nacional para promover la Ciencia y Tecnología).

# 1 Introduction

The dynamic multimodal fitness landscape is a search space which topological features of the cones could change over time. Dynamic multimodal fitness landscapes present a challenge to any search technique due to the fact that the total or partial number of topological features or problem constraints or both could change. So, new cones could appear, and these should be located quickly in order to avoid the lose of potential good solutions. Consequently, an evolutionary algorithm working on these landscapes must be able quickly adapts localizing and keeping actual and new optimal solutions.

For the changes there may be a variety of dynamic properties. For example, the magnitude of a change be small, large or chaotic and the speed of a change can be rapid or slow.

When changes are large, abrupt or chaotic the similarity between solutions found so far and the new ones can be worthless. Even under these hard environments Evolutionary Computation (EC) offers advantages, which are absent in non-population-based heuristics, when we search for solutions to non-stationary problems. The main advantage lies in the fact that Evolutionary Algorithms (EAs) keep a population of solutions. Consequently, facing the change, they allow moving from one solution to another one to determine if any of them are of merit in order to continue the search from them instead of from scratch [2].

The main problem with EAs is that eventually they converge to an optimum and thereby lose the diversity necessary for efficiently exploring the search space and also their ability to react to a change in the environment when such change occurs. Then, it is necessary to complement EAs with some strategy in order to mantain diversity in the population. Niching has been developed for this purpose. Niching tries to spread out the population over multiple cones, they should mantain diversity in the population. Clustering algorithms group similar elements from a set forming subsets, each subset represents a niche.

Goldberg and Smith [5], Cobb [4] and Grefenstette [11] initiated the research related to the behaviour of EAs on dynamic fitness landscapes between 1987 and 1992. Recently the interest in this area was increased dramatically [16], [12], [21], [24], [14], [23], [22], [17], [15]. Some research on evolutionary algorithms with clustering are described in [19, 20].

The paper is organized as follows. Section 2 presents the techniques used to mantain diversity. Section 3 describes the EAs characteristics. Section 4 describes the dynamic test functions used. Section 5 defines the metrics used. In section 6 the experiments performed are explained. In section 7 results are discussed and, finally section 8 shows our conclusions and future work.

# 2 Techniques to Keep Diversity in the Population

## 2.1 Clustering Algorithms

The goal of clustering algorithms (CA) is to group data units in clusters in a way that data units in a cluster are as similar as posible but clusters are as different as posible. In the context of evolutionary algorithms, individuals are data units and clusters are niches or species.

A basic CA [7] is a partitioning algorithm (PA). PAs construct a partition of a set $D$ of $n$ objects into a set of $k$ clusters. $k$ is an input parameter for these algorithms. Specifically, center-based PAs begin with a guess about the solution, and then refine the positions of centers until a local optimal is reached. K-Means and Hybrid belong to the family of center-based clustering algorithms. Each of them has its own objective function, which defines how good a clustering solution is. The goal of each algorithm is to minimize its objective function. Since these objective functions cannot be minimized directly, they use iterative update algorithms which converge on local minima.

We briefly describe the work of a general iterative clustering technique: let $X$ a $d$-dimensional set of $n$ data points $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$ as the data to be clustered, let $C$ a $d$-dimensional set of $k$ centers $C = \{\vec{c}_1, \ldots, \vec{c}_k\}$ as the clustering solution that an iterative algorithm refines. A membership function $m(\vec{c}_j|\vec{x}_i)$ defines the proportion of data point $\vec{x}_i$ that belongs to center $\vec{c}_j$ with constrains $m(\vec{c}_j|\vec{x}_i) \geq 0$ and $\sum_{j=1}^{k} m(\vec{c}_j|\vec{x}_i) = 1$. A weight function $w(\vec{x}_i)$ defines how

much influence data point $\vec{x}_i$ has in recomputing the center parameters in the next iteration, with constrains $w(\vec{x}_i) > 0$, this weight function was introduced in [1]. The steps for a general iterative clustering technique are:

1. Initialize the algorithm with $k$ guess centers $C$.
2. For each data point $\vec{x}_i$ compute its membership $m(\vec{c}_j|\vec{x}_i)$ in each center $\vec{c}_j$ and its weight $w(\vec{x}_i)$.
3. For each center $\vec{c}_j$ recompute its location from all data points $\vec{x}_i \in X$ according to their memberships and weights $\vec{c}_j = \frac{\sum_{i=1}^{n} m(\vec{c}_j|\vec{x}_i)w(\vec{x}_i)\vec{x}_i}{\sum_{i=1}^{n} m(\vec{c}_j|\vec{x}_i)w(\vec{x}_i)}$.
4. Repeat steps 2. and 3. until convergence.

In this work we consider the following two clustering algorithms that show the features described above.

- K-Means Algorithm (KM) [8]: Partitions data into $k$ sets. The solution is then a set of $k$ centers, each center is located at the centroid of the data for which it is the closer center. For the membership function, each data point belongs to its nearest center. The objective function that the KM algorithm optimizes is: $KM(X, C) = \sum_{i=1}^{n} min_{j \in \{1,\ldots,k\}} ||\vec{x}_i - \vec{c}_j||^2$. This objective function gives an algorithm which minimizes the squared distance between each center and its assigned data points. The membership and weight functions for KM are:

$$m_{KM}(c_l|\vec{x}_i) = \begin{cases} 1 & if\ l = \min_j ||\vec{x}_i - \vec{c}_j||^2 \\ 0 & \text{en otro caso} \end{cases}$$

$$w_{KM}(\vec{x}_i) = 1$$

  respectively. KM has a hard membership function $m(\vec{c}_j|\vec{x}_i) \in \{0, 1\}$ (every point belongs only to its closest center) and a constant weight function that gives all data points equal importance. KM is easy to understand and implement, making it a popular algorithm for clustering.

- Hybrid Algorithm(H) [8]: Uses the hard membership function of KM. But it uses a varying weight function, which gives more weight to points that are far from every center. The membership and weight functions for H are:

$$m_H(c_l|\vec{x}_i) = \begin{cases} 1 & si\ l = \min_j ||\vec{x}_i - \vec{c}_j||^2 \\ 0 & \text{en otro caso} \end{cases}$$

$$w_H(\vec{x}_i) = \frac{\sum_{j=1}^{k} ||\vec{x}_i - \vec{c}_j||^{-p-2}}{(\sum_{j=1}^{k} ||\vec{x}_i - \vec{c}_j||^{-p})^2}$$

  respectively.

## 2.2 Macromutation Operator

Random Immigrants: This idea was proposed in [10, 9]. The approach consists in replacing a percentage of the population by randomly generated individuals. The technique is applied only when a change in the environment was produced.

# 3 Characteristics of Evolutionary Algorithms

We choose three evolutionary algorithms, each algorithm combines one clustering algorithm (K-Means or Hybrid ) with random immigrants and a different crossover operator. All algorithms share the representation, selection method and mutation operator.

### 3.1 Representation

The population $P$ is made of a constant number N of chromosomes that depends on the dimensionality of each studied function. Each individual consists of a single chromosome, where each gene is a real value in the interval $[-1.0, 1.0]$ representing a coordinate in the search space. That is the $ith$ individual in the population $P$, is represented by the chromosome: $P^i =< x_{i1}, x_{i2}, \ldots, x_{ir} >$, where $x_{ij}$ denotes the $jth$ coordinate of the $ith$ individual with $j$=1,...,$r$ and $r$ is the chromosome length.

### 3.2 Operators

- Selection: The parents for the mating pool were selected by means of tournament selection [2].
- Mutation: Uniform mutation is used and it is applied with a $P_{mut}$ probability. When an individual undergoes mutation, each gene has exactly the same chance of undergoing mutation. As a result the mutated gene has a new allele value randomly chosen from the domain of the corresponding parameter.
- Arithmetic Crossover [2]: let two parents $\vec{x}_1$ and $\vec{x}_2$, with a $P_{cross}$ probability, the operator creates one descendent as average from the parents: $x'_i = 0.5x_{1i} + 0.5x_{2i}$ for $i = 1, \ldots, r$.
- Multiparents [6]Two Points Crossover [2]: let three parents $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, with a $P_{cross}$ probability, it takes pairs of them ($\vec{x}_1$ and $\vec{x}_2$, $\vec{x}_1$ and $\vec{x}_3$ and $\vec{x}_2$ and $\vec{x}_3$) and it applies two points crossover to them creating six descendents. Only the child with the highest fitness is selected to join the new population.

### 3.3 Description of the EAs

The EAs proposed are:

- *AEKM*: uses K-Means with arithmetic crossover.
- *AEH*: uses Hybrid with arithmetic crossover.
- *AEKMM*: uses K-Means with multiparents two points crossover.

All of them use random inmigrantes. All the algorithms, except for the differences indicated above, work in a similar manner: Clustering algorithm splits population in so many subpopulations as number of clusters exist. Clustering is applied after population is initialized and after a change occurs. The elitism strategy takes the best individual of each cluster and from those picks the best as global optimal. The population is generated by the selection, crossover and mutation operators that correspond. These operators work on subpopulations so interaction between subpopulations and clusters does not exist. The algorithms have a function called function_changes, it is responsible for the detection of a change in dynamic fitness function. In our case, changes occur at constant intervals, then this function only verifies if the generation number corresponds to one where the change must occur. If a change must occur then the apply_changes function is called obtaining a new dynamic fitness function. Another function is the occured_function, it tests if a change effectively has occured, in which case subpopulations are re-evaluated with the new function, then subpopulations are unified and over it clustering is applied in order to reagrup similar individuals and finally elitism is applied. Random immigrants are inserted when a change occurs after subpopulations are unified but before split them again. Algorithms finish when a fixed amount of changes are reached.

## 4 Dynamic Test Problems

This Section describes the test problem generator for the dynamic fitness landscapes used, named DF1. It was proposed by R. Morrison and K. De Jong [18]. The static function used in DF1 can be specified for any number of dimensions and it is defined as:

$$f(x_1,\ldots,x_n) = \max_{i=1,M}[H_i - R_i * \sqrt{(x_1 - x_{1i})^2 + (x_2 - x_{2i})^2 + \ldots + (x_n - x_{ni})^2}]$$

where ($x_1$,...,$x_n$) are the genes of an individual, M specifies the number of cones in the environment, and each cone is independently specified by its location ($x_{1i}$,...,$x_{ni}$), $x_{ki}$ $\in$ [-1.0,1.0] with $k$ = 1,...,$n$, its height $H_i$ and its slope $R_i$. Every of these independently specified cones is blended together using the max function. Each time the generator is called it produces a randomly generated morphology in which random values for each cone are assigned based on user-specified ranges: $H_i \in [\ Hbase, Hbase + Hrango]$ and $R_i \in [\ Rbase, Rbase + Rrango]$.

DF1 permits to control the dynamic of each change (i.e, how topological features on the landscape change) through an one-dimensional, non-linear function that has simple bifurcation transitions to progressively more complex behavior: the logistics function $Y_i = A * Y(i-1) * (1 - Y(i-1))$, where A is a constant, and $Y_i$ is the value at iteration $i$. As A is increased a more complicated dynamic behavior emerges. DF1 uses Y values produced on each iteration to select step a size for the dynamic portions of the landscape. It requires the values chosen for A to be greater than one (small change) and less than four (chaotic change). DF1 allows to change any topological feature on the landscape. However, in our work we just change the location of the highest cone (global optimal).

## 4.1 Test Problems Used

We considered 3 static functions generated by DF1. All of them were selected in order to validate our algorithms, with respect to a small set of landscapes. Each landscape has a different complexity level. Table 1 provides the general parameters for each test function: height $H$, slope $R$, maximum fitness of highest and sloppiest MaxH and MaxR, respectively. For all functions, independently of the number of cones, one of them was initialized as global maximum (global optimal) using MaxH and MaxR, the rest of the cones were generated as specified in Section 4. Besides, FD3 has a cone with height = slope = 7.0. Following is the description of the function features.

Table 1: Test Functions Parameters

| Function | Hbase | Hrange | Rbase | Rrange | MaxH | MaxR |
|---|---|---|---|---|---|---|
| FD1 | 60.0 | 10.0 | 50.0 | 10.0 | 70.0 | 59.01 |
| FD2 | 60.0 | 10.0 | 50.0 | 10.0 | 80.0 | 80.0 |
| FD3 | 5.0 | 0.0 | 4.0 | 0.0 | 10.0 | 17.0 |

On FD1 (see Figure 1 a and b), the highest cone was fixed in 70.0, the next in 68.9 and the rest height cones were fixed between 60.3 and 68.5. Therefore, FD1 is considered the most complex test function. On FD2 (see Figure 1 c and d), the highest cone was fixed in 80.0, the next in 68.9 and the rest height cones were fixed between 60.3 and 68.5. Thus, FD2 is considered the least complex test function. Since local optimals may not disorient the search process strongly. Finally, on FD3 (see Figure 1 e and f) the highest cone was fixed in 10.0, the next in 7.0 and the rest height cones were fixed in 5.0. The 7.0-heigh cone should act like an attractor in order to strongly disorient search process.

We worked on different functions, every static function was scaled in dimensionality and multimodality, FD#f-#d#c indicates number of static function, dimensions and the number of cones. The number total of test functions is 12. All functions have a characteristic: not one cone on the landscape covers another cone on the same landscape, at least until the function starts to change.

# 5 Performance Measures

The goal of an evolutionary algorithm in dynamic environments is not only to find one global optimal but rather a perpetual adjustment to changing environment conditions [13]. Besides the accuracy of a solution at time $t$, the stability of a solution is of interest too. Several measures are proposed in [13]. This paper considers different measures. These are:

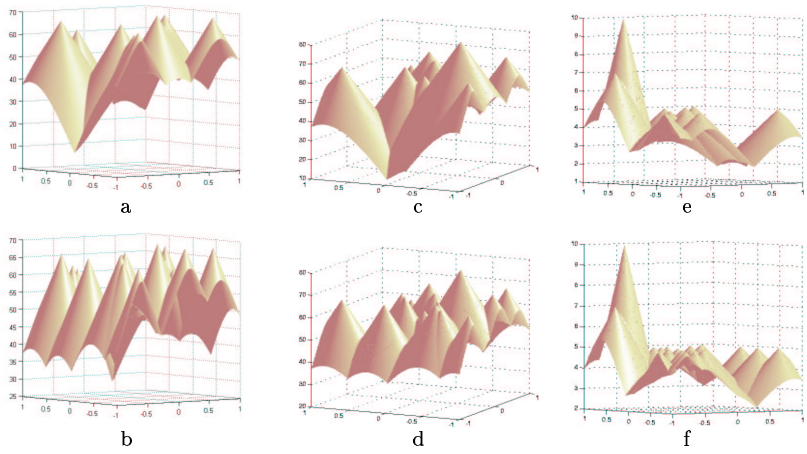


Figure 1: a- FD1-2d10c. b- FD1-2d20c. c- FD2-2d10c. d- FD2-2d20c. e- FD3-2d10c. f- FD3-2d20c.

- *Accuracy* [13]: accuracy for a sequence of $K$ changes is defined as:

$$Acc = \frac{1}{K} \sum_{i=0}^{K-1} \frac{F(best^{(i)}) - Min^{(i)}}{Max^{(i)} - Min^{(i)}}$$

  where $K$ is the number of changes suffered by the fitness function, $best^{(i)}$ is the best candidate solution in the population, $Max^{(i)} \in R$ and $Min^{(i)} \in R$ are the best and worst values in the search space in the generation just before $ith$ change, with $i \in [0, K-1]$. The accuracy values range between 0 and 1. When accuracy is 1, it means that the best individual in the population is found as the global optimal in generation just before the change.

- *Stability* [13]: in the context of dynamic optimization, an adaptive algorithm is called stable if changes in the environment do not affect the optimization accuracy severely. Even in the case of chaotic changes an algorithm should be able to limit the fitness drop. It is defined as:

$$Stab = \max\{0, Acc^{(t-1)} - Acc^{(t)}\}$$

  and ranges between 0 and 1. A value close to 0 implies a high stability. This measure should not serve as the only criterion since it makes no statement on the accuracy level.

Measures based on fitness of one or more individuals could have some problems when they are used in an isolated way. In some cases, individual fitness could be misleading. We use several measures, some of them are based on fitness (accuracy and stability) and the others are based in the proximity to the global optimal. These can only be used if, all the time, location, height and slope of the cones contained in the landscape are known. This is our case because DF1 allows us to know these values.

So all the performance metrics that we used are:

- Number of correct reactions to changes ($RC$),
- Average accuracy ($\overline{Acc}$),
- Average stability ($\overline{Stab}$),
- Average occupied cones ($\overline{CC}$), we specified that a cone is occupied if at least one individual belongs to the cone and
- Average distance of the best individual to the current optimal ($\overline{DO}$).

Note: we indicated that EA reacts well or reacts acceptably when the best individual of the population belongs to the global optimal cone. All averages are taken from all runs.

# 6 Experiments

Except population size, the parameter settings for the EAs remained fixed throughout all experiments and were determined as the best after a series of initial trials. The population size was set to 100 and 150 for 2 and 5 dimensions functions, respectively. $P_{cross}$ and $P_{mut}$ were fixed at 0.6 and 0.5, respectively. Tournament size was 2. The percentage of random immigrants was set to 30% of the population size. The individuals to be replaced by immigrants were randomly selected with equal probability. A number of experiments were designed differing in the function selected and in the severity of the changes to perform on it. The number clusters was the number of cones in the function. Changes were produced each 10, 50 and 70 generations for all functions. The main goal here was to determine if the algorithms succeeded to faithfully tracking the changes in location of the cone containing the optimum value when the fitness landscape changes. The algorithm was allowed to run as many generations as changes were desired. For all experiments we fixed at 20 the number of changes. For each of these experiments 30 runs were performed with distinct initial populations. Thus, a maximum of 600 changes could be detected. The change in the location of the cone containing the optimum value is the only change analyzed in the present work. Parameter setting for severity of changes were for large changes A=1.5 and Cstepscale=0.99 and for chaotic changes A=3.8 and Cstepscale=0.5. The severity of changes was selected because we agree with Branke [3] in that small and frequent changes should be accounted by creating robust solutions for the algorithm, while large and infrequent changes should be handled by adaptation. The Cstepscale constant is used to move each coordinate on the range specified by the user.

# 7 Results

This paper considers that an EA has an acceptable performance if it acceptably reacts at least to 70% of the changes, that is, 420 changes. Functions have been arranged in number of dimensions and number of cones, 5-dimensional functions are considered more complex than 2-dimensional functions and 20-cone functions are considered more complex than 10-cone functions. Ordering functions from lower to higher complexity we obtained: 2d-10c, 2d-20c, 5d-10c and 5d-20c. Tables 2 and 3 show the function to which the algorithm has acceptably reacted , but also reacted to less complex functions. For instance, if 5d-20c belongs to the Table for FD2 under AEKM each 10 generations then AEKM has reacted to 2d-10c, 2d-20c, 5d-10c and 5d-20c functions.

Table 2: Algorithms and number of correct reactions to change for large changes - Number between () indicates percentage of number of correct reactions to change

| Interval | Function | *AEKM* | *RC* | *AEH* | *RC* | *AEKMM* | *RC* |
|---|---|---|---|---|---|---|---|
| 10 | FD1 | 2d10c | 493 (82.1%) | 2d10c | 492 (82%) | 2d10c | **502** (83.6%) |
| | FD2 | 5d20c | 491 (81.8%) | 5d20c | 500 (83.3%) | 5d20c | **510** (85%) |
| | FD3 | 2d20c | **600** (100%) | 2d20c | **600** (100%) | 2d20c | **600** (100%) |
| 50 | FD1 | 2d20c | 577 (96.1%) | 2d20c | 574 (95.6%) | 2d20c | **586** (97.6%) |
| | FD2 | 5d20c | **600** (100%) | 5d20c | 598 (99.6%) | 5d20c | **600** (100%) |
| | FD3 | 5d20c | 460 (76.6%) | 5d20c | 455 (75.8%) | 5d20c | **513** (85.5%) |
| 70 | FD1 | 2d20c | 594 (99%) | 2d20c | **598** (99.6%) | 2d20c | **598** (99.6%) |
| | FD2 | 5d20c | 599 (99.8%) | 5d20c | 450 (75%) | 5d20c | **600** (100%) |
| | FD3 | 5d20c | 499 (83.1%) | 5d20c | 499 (83.1%) | 5d20c | **562** (93.6%) |

From Table 2 we observed that AEKMM has a better performance that AEKM and AEH. This is more evident when algorithms have more time to evolve. For FD1, algorithms correctly react to the 2d-10c function when changes occur every 10 generations and to 2-dimensional functions when changes occur every 50 and 70 generations. For FD2, all algorithms acceptably achieve their purposes for all functions and all intervals between changes. On the other hand, for FD3, algorithms only can to react to the 2d-10c function when changes occur each 10 generations and for all functions when changes occur each 50 and 70 generations. Accuracy

varies from 0.81 to 0.99 with a stability from 0.05 to 0.09. Average occupied cones vary from 9.3 to 9.7 for 10-cone functions and from 17.9 to 19.5 for 20-cone functions. Average distance of the best individual to the current optimal varies from 0.05 to 0.5.

Table 3: Algorithms and number of correct reactions to change for chaotic changes - Number between () indicates percentage of number of correct reactions to change

| Interval | Function | *AEKM* | *RC* | *AEH* | *RC* | *AEKMM* | *RC* |
|---|---|---|---|---|---|---|---|
| 10 | FD1 | 2d10c | **474** (79%) | 2d10c | 456 (76%) | 2d10c | 407 (67.8%) |
| | FD2 | 5d20c | 454 (75.6%) | 5d20c | 447 (74.5%) | 5d20c | **462** (77%) |
| | FD3 | 2d20c | **600** (100%) | 2d20c | **600** (100%) | 2d20c | **600** (100%) |
| 50 | FD1 | 2d20c | 569 (94.8%) | 2d20c | **573** (95.5%) | 2d20c | 570 (95%) |
| | FD2 | 5d20c | 553 (92.1%) | 5d20c | 551 (91.8%) | 5d20c | **595** (99.1%) |
| | FD3 | 5d20c | 437 (72.8%) | 5d20c | 424 (70.6%) | 5d20c | **454** (75.6%) |
| 70 | FD1 | 2d20c | 589 (98.1%) | 2d20c | **590** (98.3%) | 2d20c | **590** (98.3%) |
| | FD2 | 5d20c | 564 (94%) | 5d20c | 566 (94.3%) | 5d20c | **594** (99%) |
| | FD3 | 5d20c | 441 (73.5%) | 5d20c | 464 (77.3%) | 5d20c | **494** (82.3%) |

From Table 3 we observed that, AEKMM's performance overcomes AEKM and AEH's performance under FD2 and FD3, which is more evident when algorithms have more time to evolve. However, for FD1, AEKM's performance was better than the other algorithms when changes occur every 10 generations, and AEH's performance was better than the other algorithms when changes occur every 50 and 70 generations. Again, for FD1, algorithms only achieve their purposes for 2d-10c function when changes occur each 10 generations and for 2-dimensional functions when changes occur each 50 and 70 generations. For FD2, all algorithms acceptably react to all functions and all intervals between changes. On the other side, for FD3, algorithms only correctly react for 2-dimensional functions when changes occur each 10 generations and for all functions when changes occur each 50 and 70 generations.

Accuracy varies from 0.8 to 0.99 with a stability from 0.05 to 0.08. Average occupied cones varies from 9.5 to 9.8 for 10-cone function and from 17.9 to 19.5 for 20-cone functions. Average distance of the best individual to the current optimal varies from 0.05 to 0.6.

Figure 2 shows average best individual per generation for every function when change occur every 10 generations under large and chaotic changes, for all the EAs considered.

# 8 Conclusion

The values of accuracy and stability metrics should not be analyzed in an isolated way because they may be misleading as to the real performance of the algorithms. In fact, if the difference between the heights of the landscape cones is not meaningful, even when the best individual of the population does not belong to the optimal cone, the accuracy and stability values could still be good and so they do not reflect the real ability of the algorithms to react to changes. For this reason we also used other significant metrics as number of changes in which the algorithms reacted correctly, distance of the best individual found by the EAs to the current optimal, among others.

Accuracy values found were very good, between 0.8 and 0.9, so the quality of solutions found was good, and the low stability values imply that this good behavior was stable over the evolution. In general, distance of the best individual to the current optimal increases when functions are more complex. However, these distances are not unacceptable.

The performance of the clustering algorithms was measured by the average occupied cones. In general, average occupied cones were from 9 to 10 for 10-cone functions and 17 to 19 for 20-cone functions when algorithms reacted well and when they did not do it. Although these values are acceptable, the fact that they remained acceptable over good and bad performance of the algorithm implies that the global optimal cone almost was always occupied in both cases. Sometimes global optimal cone was not occupied whereas some other times the best individual of the population did not belong to the global optimal cone. However, the population was always well spread . In average, more than 90% of the cones were occupied. Thus, mechanisms

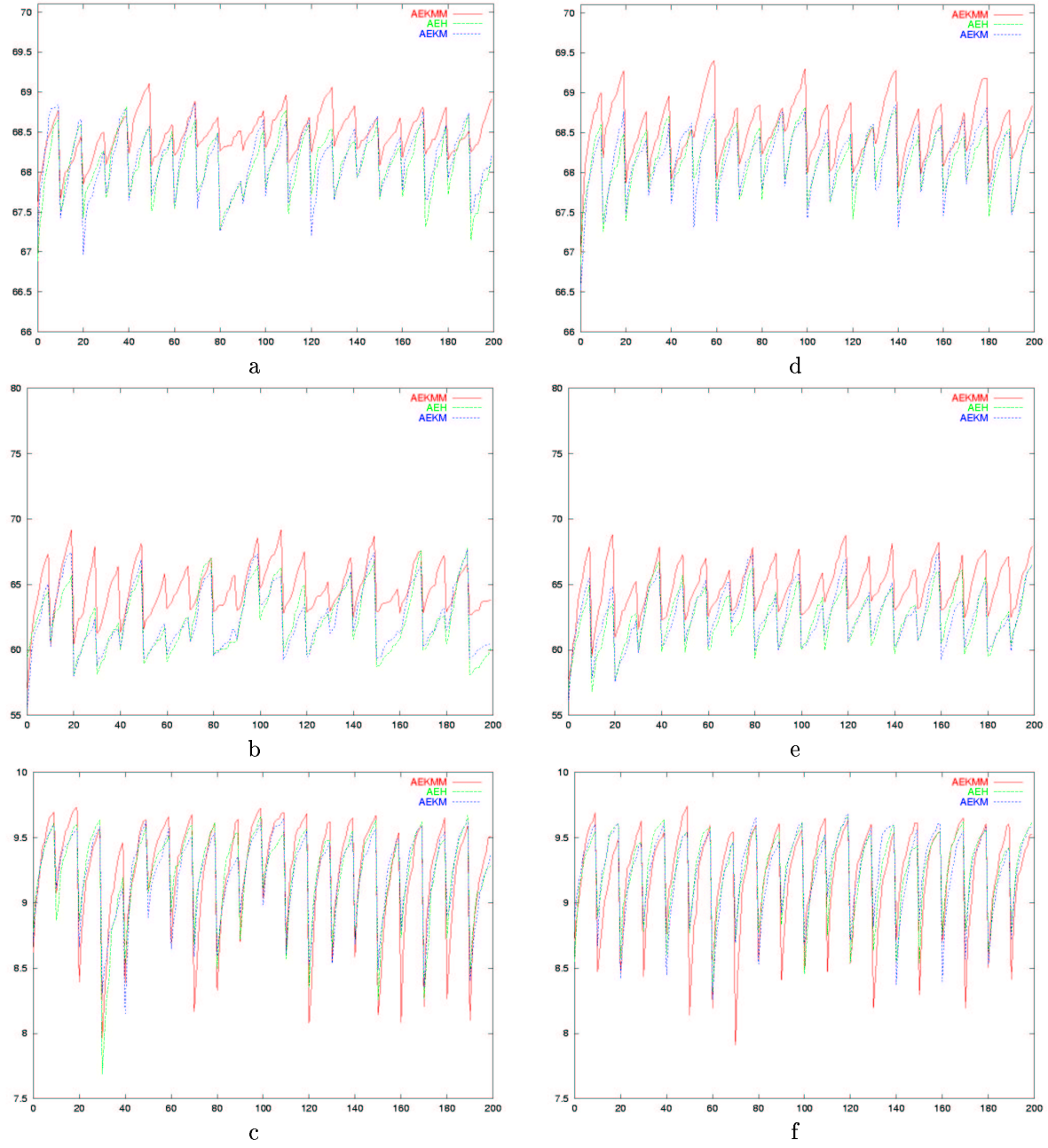


Figure 2: Average best individual per generation for a- FD1-2d10c b- FD2-5d20c c- FD3-2d20c when changes occur every 10 generations under chaotic changes. Average best individual per generation for d- FD1-2d10c e- FD2-5d20c f- FD3-2d20c when change occur each 10 generations under large changes.

that work on subpopulations achieved their goal: diversify the population, that is, spread population in order to represent different significant search subspaces.

For both clustering algorithms, we did not observe that one of them to be better than the other. In some experiments one of them shows a better performance than the other one and viceversa.

When the EAs did not appropriately react , it was not a problem of the clustering algorithms. The problem may have resulted eother from bad quality of initial solutions or from the fact the evolutionary operators did not perform a sufficient exploration and explotation in order to find great solutions. However, multiparents two points crossover seemed to be better than arithmetic crossover.

Crossover operators are responsible for exploiting solutions. AEKMM uses a crossover that involves three parents and it generates several children. The best child is included in the next

generation. Results indicate that this crossover is more effective than the arithmetic crossover in order to exploit quality solutions. This may be due to the fact that multiparents two points crossover chooses the best child whereas arithmetic crossover do not.

Future work will involve the improvement of evolutionary operators in order to get a better exploration and explotation.

# References


[1] Zang B. Generalized k-harmonic means-boosting in unsupervised learning. Technical report, Hewlett-Packard Labs., 2000.

[2] Fogel D. Back T. and Michalewicz Z. *Handbook of Evolutionary Computation.* Oxford University Press, new york, oxford edition, 1997.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, 2002.

[4] H. G. Cobb. An investigation into the use of hypernutation as an adaptive operator in genetic algorithms having continuous, time dependent non-stationary environments. Technical Report 6760, Naval Research Laboratory, USA, 1990.

[5] Goldberg D. and Smith R. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 59–68. Lawrence Erlbaum Associates, 1987.

[6] Rau'e P-E Eiben A. and Ruttkay Z. Genetic algorithms with multi-parent recombination, 1994. Parallel Problem Solving From Nature (PPSN III), Lecture Notes in Computer Science 866, Ed. Yu Davidor, H-P. Schewefel and R. Manner, pp. 77-87.

[7] Sander J. Ester M., Kriegel H. and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceeding og 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, 1996. AAAI Press.

[8] Hamerly G. and Elkan C. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the ACM conference on information and knowledge management (CIKM)*, November 2002.

[9] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transaction on Systems, Man and Cybernetic*, 16:122–128, 1986.

[10] Cobb H. and Grefenstette J. Genetic algorithms for tracking changing environments. In *Proceeding of the 5th IEEE International Conference on Genetic Algorithms*, pages 523–530. Morgan Kauffman, 1993.

[11] Grefenstette J. Genetic algorithms for changing environments, 1992. In Proceedings of Second Parallel Problem Solving From Nature (PPSN-2), Brusse 28-30 September, pages 137-144, 1992.

[12] E. Hart. J. Lewis and G. Ritchie. A comparision of dominance mechanism and simple mutation in non-stationary problems. In Morgan Kaufmann, editor, *Proceeding 7th International Conference on Genetic Algorithms*, pages 138–148, 1997.

[13] Weicker K. Performance measures for dynamic environments, 2002. In J.J. Merelo, P. Adamidis, H.-G. Beyer, J.L. Fernndez-Villacaas, and H.-P. Schwefel, editors, Parallel Problem Solving from Nature, volume 2439 of LNCS, pages 64-73. Springer.

[14] Z. Michalewicz K. Trojanoswky. Searching for optima in non-stationary environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 1843–1850, Washington DC, USA, IEEE Service Centre, 1999.

[15] R. Morrison. *Designing Evolutionary algorithms for Dynamic Environments.* PhD thesis, George Mason University, USA, 2002.

[16] Mori N. Adaptation to changing environments by means of the memory based thermodynamic genetic algorithm. In Morgan Kaufmann, editor, *Proceeding 7th International Conference on Genetic Algorithms*, pages 299–306, 1997.

[17] T. Nanayakkara et al. Adaptive optimization in a class of dynamic environments using an evolutionary approach, 1999.

[18] Morrison R. and De Jong K. A test problem generator for non-stationary environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 2047–2053, Washington DC, USA, IEEE Service Centre, 1999.

[19] Tucker A. Sheng W. and Liu X. Clustering with niching genetic k-means algorithm. In *GECCO (2)*, pages 162–173, 2004.

[20] Stein G. Ulmer H. Streichert F. and Zell A. A clustering based niching ea for multimodal search spaces. In *Proceedings of the 6th International Conference on Artificial Evolution*, Marseille, France, 27-30 October 2003. Springer Verlag.

[21] Bäck T. On the behavior of evolutionary algorithms in dynamic fitness landscapes. In *Proceeding of the IEEE International Conference on Evolutionary Algorithms*, pages 446–451, IEEE Service Centre, 1998.

[22] Liles W. and De Jong K. The usefulness of tag bits in changing environments, 1999. IEEE, In Congress on Evolutionary Computation, volume 3, pages 2054-2060.

[23] K. Wicker and N. Weicker. On evolutionary strategy optimization in dynamic environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 2039–2046, Washington DC, USA, IEEE Service Centre, 1999.

[24] C. O. Wile. *Evolutionary Dynamics in Time-Dependent Environments.* PhD thesis, Institut für Neuroinformatik, Ruhr-Universität, Bochum, Germany, 1999.