# On the Design of Adaptive Control Strategies for Evolutionary Algorithms

Jorge Maturana and Frédéric Saubion

LERIA, Université d'Angers
2, Bd Lavoisier 49045 Angers (France)
{maturana,saubion}@info.univ-angers.fr

**Abstract.** This paper focuses on the design of control strategies for Evolutionary Algorithms. We propose a method to encapsulate multiple parameters, reducing control to only one criterion. This method allows to define generic control strategies independently from both the algorithm's operators and the problem to be solved. Three strategies are proposed and compared on a classical optimization problem, considering their generality and performance.

## 1 Introduction

Evolutionary Algorithms (EAs) [1] are metaheuristics inspired by natural evolution that are used to find sufficiently acceptable solutions to complex optimization problems. A set of candidate solutions, known as *population*, evolves by means of genetic operators. The two main operators are *mutation*, that modifies randomly an individual from the population, and *crossover*, that combines two of them. A *selection* process chooses the individuals that will survive in the next generation population. The whole process is repeated until a termination condition is satisfied. One of the strongest advantages of EAs over traditional optimization methods is their ability to escape from local optima. They have been successfully applied to various application domains.

Most of the time, the performance of these algorithms are strongly related to a suitable setting of several parameters such as population size and operator's application rate. The tuning of these parameters is difficult to achieve and often depends on empirical experiments or intuition. From the problem's resolution point of view, these parameters can be used to control the exploration of the search space and the exploitation of its interesting areas. If exploitation (also known as intensification of the search) is excessive, premature convergence may occur, while if exploration is too excessive (i.e., diversification), the algorithm becomes inefficient. The management of these two search strategies is indeed the central preoccupation of search (meta)heuristics.

Another important difficulty when using EAs to solve specific problems is the limited efficiency of generic evolution operators. Generally, the performance of an EA is also strongly related to the definition of efficient dedicated operators that take into account the structural properties of the considered problem (without

N. Monmarché et al. (Eds.): EA 2007, LNCS 4926, pp. 303–315, 2008.

neglecting the importance of encoding). These operators are often controlled by parameters, even in its most elemental way of application rate. The influence of those parameters over the Balance between Exploration and Exploitation (EEB) is a priori unknown, and knowledge about it is usually acquired across computationally expensive sets of experiments.

Control strategies often rely on specific rules to control a particular parameter. This makes it impossible to apply the acquired knowledge to algorithms with different operators: *knowledge is not exportable because it is not expressed in general terms*. It would be then interesting to generate control strategies w.r.t. EEB, which would allow us to encapsulate the complexity of handling specific parameters and define simpler and more general control schemes.

In this paper, we present a study about general control strategies based on a more global view of EA behavior. We first use a method to encapsulate the complexity of handling multiple parameters, even if they are associated to unknown-behavior operators [2]. This scheme focuses on a particular criterion: the population diversity. The population diversity and quality (i.e. mean fitness), produced by different combination of parameter's values, are measured during a training phase. Then, the combinations that provide maximal quality for different levels of diversity are identified and used later during the control phase. Genotypic diversity[1] is highly correlated with EEB: if exploitation is intensive, individuals will tend to concentrate in the higher fitness zones, so diversity will be low. On the other hand, if exploration is sparse, individuals will be dispersed in the search space, and diversity will increase.

Then, we propose several control strategies for managing diversity along the search process. By using diversity as the main controlling parameter, strategies can be expressed in more general terms of exploration and exploitation, common to all EAs. These control strategies are experimented and compared on a well-known combinatorial optimization problem: the quadratic assignment problem.

This paper is organized as follows. Sect. 2 summarizes relevant work, Sect. 3 provides an overview of our approach, while Sect. 4 analyzes several criteria to define general control strategies. Sect. 5 discusses experimental results, to finally draw conclusions in Sect. 6.

## 2 Relevant Related Work

### 2.1 Parameter Control

In [3], a broad number of approaches to control EA parameters have been reviewed and classified according to the taxonomy of Fig. 1.

Parameter setting strategies are divided in two main sets: those that fix parameters for the whole search *before the run*, and those that change their values *during the run*. In the first group the central task consists in finding fixed recommended values. In the second group parameter's values change during the run,

[1] Measured as the difference between individuals in the population. Since this approach can be applied to different problems, diversity must be defined accordingly.

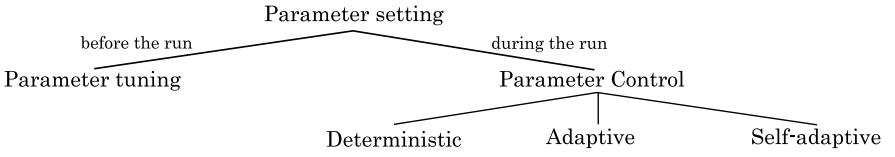


**Fig. 1.** Taxonomy of parameter control proposed by Eiben et al. [3]

those are divided according to how the adjustment is achieved: *Deterministic* control changes parameter's values by using deterministic rules, usually in relation with the number of elapsed generations. *Adaptive* control modifies parameters according to the current state of the search. Finally, parameter modification in *self-adaptive* control is performed by coding parameters inside individuals and make them evolve together.

Most studies focus on specific parameters control, with just a few exceptions. An adaptive genetic algorithm is presented in [4], where the relationship between state measures and parameters are encoded in control rules. In [5] a statistical method is used to measure relevance and to tune the parameters of an EA thanks to a second one. Two dynamic control strategies are compared in [6], where parameters are awarded according to their past performance.

Moreover, the integration of different fields of Artificial Intelligence has led to new kinds of control approaches. One of these approaches involves Fuzzy Logic (FL), where fuzzy rules are used to set parameter's values based on performance measures [7]. Our approach to handle parameters is based on this mixture, but applies FL not to control but to modeling behavior, while control is based on adaptive heuristics.

### 2.2 Fuzzy Logic Controllers

FL is an extension of classic boolean logic where levels of truth are expressed by a membership function with values ranging from 0 (false) to 1 (true). One of the most useful applications of FL are Fuzzy Logic Controllers (FLC) [8,9]. FLCs allows to infer answers from rules such as "IF car_speed is high AND road is dry, THEN risk is medium". Since FLCs are universal approximators of continuous functions [10], they act as modeling tools that express the output w.r.t. inputs. An early application was proposed by Wang and Mendel [11], in which many of the subsequent methods have been based.

## 3 Handling Multiple Parameters in EAs

### 3.1 Overview of the Approach

When faced to an EA, the user needs to understand its behavior in order to correctly tune its parameters and benefit from acceptable performances. Most of the time, a learning process (usually a long set of experiments using the algorithm with different parameters values) is not included in the algorithm but relies on

the user's expertise and intuition. Then, she/he may apply her/his personal, and often empirical, control rules.

In a similar way, our approach is divided in two phases: *Learning* and *Control*. During *Learning*, the algorithm produces examples (EA's generations) using different parameter's values, to capture the mapping of these combinations with genotypic diversity. Since populations with similar levels of diversity may vary in terms of quality, another model is built, in order to link parameters and quality, measured in terms of mean fitness.

Both models are used to find the combinations of parameter's values that produce the higher quality populations corresponding to different values of diversity, which are obtained from a fine partition of reachable diversity. With this approach, the only parameter to modify thereafter is diversity.

During the *Learning* phase, three main problems arise: *dimensionality*, *inertia* and *noise*. Dimensionality is related to the fact that the amount of examples to be generated depends exponentially on the number of controlled parameters. Inertia is related to the resistance to the change of diversity and mean fitness values between consecutive generations. Here, we understand noise as the short-term variation product of random operators that induce inaccuracy in modeling.

During the *Control* phase, the controller changes diversity (and therefore parameters) in order to correctly exploit the search space and try to escape from local optima. It allows the user to express a generic strategy that can be applied to algorithms with different operators and solving different problems.

To provide an easy integration with any EA, the controller algorithm has been designed to be as independent as possible. Communication occurs as follows: the EA informs about current diversity and quality, and the controller assigns new values to controlled parameters, decides the reinitialization of population and the end of the search.

### 3.2 Learning Phase

There are 4 subphases in *Learning*: 1.*Example production*, in which examples are generated for every fuzzy partition combination; 2.*Modeling*, where diversity and quality FLCs are built, based on earlier collected examples; 3.*Refinement*, in which new examples, focused on the most promising areas, are generated to achieve a fine tuning of the model; and 4.*Releasing*, where all examples are used to build the definitive model, which will be used during *Control* phase.

The effects of noise and inertia are specially disturbing during *Example production*. Several techniques have been used to mitigate those effects. The first one aims to eliminate the influence of initial population by ignoring a number of generations at the beginning of the search.

Inertia has the undesirable effect of flattening measures of diversity and quality, as shown in Fig. 2.a (the surface should be continuous), where the examples of a each training grid cell (shown in the base of the plot), has been generated before passing to the next cell. Note that flattening occurs even in a close area, so it is advisable to use a training grid fine enough. We have defined a grid in function of fuzzy partitions of FLC's input variables. The intersection of

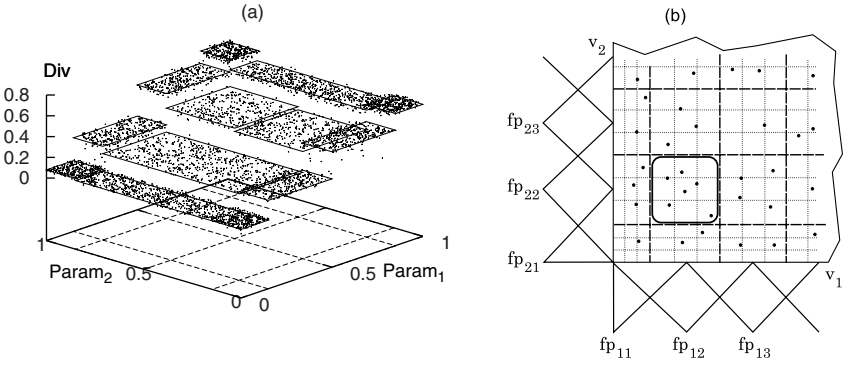


**Fig. 2.** (a) Formation of platforms (emphasized by squares) in a 4x4 coarse training grid, (b) influence area ($fp_{12}$,$fp_{22}$) for two dimensions, in a partition with fineness of 3

all parameter's partitions define what we have called *influence areas*, that are subdivided by a factor of fineness (Fig. 2.b).

In order to avoid abrupt changes in parameters that would increase the undesirable effects of inertia, we have defined a visiting order called *smooth*, in which just one parameter value is modified in a minimal amount each time. Fig. 3 shows examples for 2 and 3 parameters in contrast with classical "nested loop" visiting order.

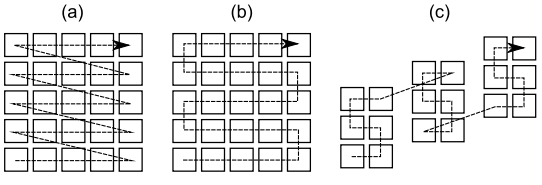


**Fig. 3.** Visiting orders: (a) classical nested loop, (b) smooth in 2D, (c) smooth in 3D

Once examples are collected, the *Modeling* subphase takes place. A Takagi-Sugeno FLC with polynomials of order 1 is used. To obtain the coefficients of the polynomials of the rule corresponding to an influence area, the algorithm performs a multiple linear regression of the examples collected to build FLCs in the sense *parameters* → *diversity/fitness*.

Fitness FLC is built similarly to diversity FLC, with the difference that an exponentially descending weighted average correction is done to consider the effects of long-term operators, like mutation. This method has also the advantage of reducing the noise. To cancel the bias produced by this correction, an even number of visiting runs are performed, shifting the direction every time.

Since controller requires the inverse, i.e. which parameters produce a given level of diversity with maximum quality, a dense grid of parameter combinations must be created to first find –using Diversity FLC– which ones have the required

diversity, and then –using Fitness FLC– the one with the higher quality. Values of diversity and corresponding optimal parameters are stored in the so-called *cachedDiv* table.

In order to refine the model, a kind of "beta testing" is performed during the *Refinement* subphase, generating examples with the optimal parameter's values found, including a normal-distributed error. After that, during *Releasing* subphase, all generated examples are used to build the definitive model.

## 4 Control Strategies for EAs

By modeling diversity and quality w.r.t. parameter values, control strategies can be expressed closer to EEB than existing control methods. Therefore, the strategies could be applied to EA's that solve different problems with other operators. The issue is then to manage diversity in order to escape from local optima and to better exploit promising areas. This section discusses several criteria to design such strategy.

If an excessive diversity is allowed, it is likely that an excessive exploration will occur, without concentrating in the most interesting zones. On the other hand, if diversity falls to a small value, all individuals will tend to concentrate and will be trapped in a local optima. Additionally, if the latter happens, it would be difficult to reconstruct a population both diverse and of good quality, since all secondary optima must be found again. Therefore, an intermediate "correct" value of diversity should be maintained to have a good balance between exploration and exploitation, and mainly avoid the loss of diversity. Of course, all problems have different levels of "correct" diversity, so the algorithm must be able to identify it. A possible approach is to observe the fitness value of the better individual during the last generations. If the same value is often repeated, it is likely that the population is converging to the point corresponding to that fitness.

Another consideration is to alternate between stages of exploration and exploitation. Actually, during preliminary experiments, we have noted that there are some problems that were solved very easily with a simple zigzag between minimal and maximal diversities. It seems that it is sometimes necessary to "forget" what has been found to have the chance to explore a totally different zone of the search space and –perhaps– find a better solution. The question of how long, in terms of generations, should last this forgetting period is also another issue to consider: if it is too short the population will return to its initial position. On the other hand, if it is too long, the algorithm will loose computation time, although, since the parameters corresponding to maximal diversity are set to be quality-optimal, the risk of loosing the entire wealth of the population is much smaller than those when population is regenerated.

We have also experimented strategies with a small oscillation around the nominal level of diversity, that both performs a local exploration/exploitation and helps to stabilize the value of observed diversity in relation to commanded diversity. Another well-known consideration is to first explore and then exploit,

in such way that the algorithm concentrates progressively in the most promising areas of the search space.

**Tested Control Strategies.** In order to compare control strategies, we have tested three different approaches that emphasize some of the aspects discussed earlier. Those strategies varies from maintaining diversity in a rather stable level to moving it abruptly during the search.

- **MX (Mixed):** it integrates first-explore-then-exploit, forgetting and small oscillation. A series of intermediate descending diversity levels: 0.7, 0.6, 0.5, 0.4, 0.3 and 0.2, in the range of achievable diversity, belonging to $[0, 1]$, are commanded to the EA, with an oscillation of 10% of this range, both above and below the nominal level. A period of 300 generations are executed at each level, which are extended in case of finding an historical improvement. After the algorithm has performed the descending steps of diversity, this one is raised to its maximum value, to escape from local optima, during 200 generations. After this, decreasing starts again.
- **CD (Correct Diversity):** it tries to reach an adequate diversity level. Every 10 generations, the fitness of the best individuals of the last 100 generations are considered. If more than 46 of them have the same value, commanded diversity is risen by 0.003, while if there are less than 17, diversity is lowered by 0.001. Those values where obtained from preliminary experiments. If the repeated value is not the higher one, or if the higher one has been obtained recently, the rising rule is relaxed a bit. Note that it is faster to rise diversity than to decrease it, what reflects the importance of avoiding premature convergence.
- **ZZ (ZigZag):** it implements a wide oscillation around a central value of diversity. This value is given by the mean of commanded diversities corresponding to the last five historic improvements. The oscillation, centered at this point, grows until the limits of possible diversity. If an historic improvement is reached, the amplitude of the oscillation is reset to zero, to start growing again.

Some control parameters were tuned to obtain reasonable results, even if our goal here is not to define the best heuristic for solving QAP but to compare several control approaches. However, it must be noted that different instances presented considerable differences among them, as we will see in the following sections. I order to provide more general and reliable control strategies over a wider set of benchmarks, a learning component could be studied in future research.

## 5 Experimentation

In this section, we describe the experiments we have carried out in order to compare the different control strategies described in section 4 [2]. We use an

[2] It is also possible to obtain fixed settings by taking parameters corresponding to a given diversity level in CachedDiv. Those settings normally produce worse performances than adaptive strategies, as shown in [2].

EA to solve the Quadratic Assignment Problem (QAP) with three operators, whose application rate parameters are controlled according to our method. Our purpose is not to be competitive w.r.t. state of the art solvers on this particular problem, but rather to compare performances of the previously presented control strategies.

### 5.1 Quadratic Assignment Problem

QAP is a well-known combinatorial optimization problem that can be stated as follows. Let us consider two matrices $A = (a_{ij})_{n\times n}$, $B = (b_{kl})_{n\times n}$, and a mapping function $\Pi$. The goal is to find a permutation $p_i = (\pi(1), \pi(2), \ldots, \pi(n))$ that minimizes:

$$f(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}$$

This problem was formulated by Koopmans and Beckmann [12] for a facility allocation problem, in which a set of $n$ facilities with physical flows between them (matrix $A$) must be placed in $n$ locations separated by known distances (matrix $B$). The goal is to minimize the cost ($flow \times distance$) of overall system.

A set of 38 medium-size instances, obtained from the QAPLIB repository [3], was selected to test the algorithm, covering instances from all families.

### 5.2 Evolutionary Algorithm

The individuals are encoded as permutations. Population size is set to 100 individuals and three operators are applied: standard exchange mutation, that simply interchange two allocations randomly, cycle crossover [13], that preserves the absolute position of allocations from parents to descendants, and a specialized operator called *remake* that randomly erases four allocations, try the 4! possible reconstructions and chooses the best one. In order to focus on the general abstraction and control methodology, the selection process is not considered here as a mechanism to control diversity. Therefore, we choose a very basic selection scheme (roulette wheel) that correspond to a rather naive genetic algorithm implemented by a non specialist user. A set of 15 runs of 10.000 generations (learning not considered) have been performed for each instance and strategy. This great amount of generations was defined to observe how definitive is premature convergence in every case.

Diversity is defined as the sum of the differences of encoded variables between all population's individuals, and scaled linearly in [0,1] between minimal an maximal possible values, given the number of variables and the population size.

### 5.3 Learning Phase Parameters

During *Learning* phase, 2.000 generations were ignored at the beginning of *Example production* and *Refining* phases. Parameter's value range were divided

[3] http://www.seas.upenn.edu/qaplib/

into 4 fuzzy partitions and subdivided with fineness of 3. Within each partition of fineness, 5 generations were executed. During *Refining*, diversity descends and mounts linearly for 800 generations each one. In order to eliminate the effects of the modeling in the strategy comparison, 15 preliminary runs were made for each instance and only one cachedDiv has been chosen for each instance. The chosen cachedDiv was the one that presented the smaller deviation of observed diversity from commanded diversity during test runs.

### 5.4 Results and Discussion

Table 5.4 presents the mean values of cost and the standard deviation (in parenthesis) for each strategy and instance. An additional column shows the best known solution published in QAPLIB (April 2007). At the bottom of the table we have included the average number of runs in which the best known value was reached, and the number of instances where each strategy significatively[4] outperformed the others.

MX and CD have obtained the best results across different instance's families, with a slight advantage of CD. On the other hand, ZZ seems to be the least efficient strategy, with a couple of exceptions. However, the mean number of times in which the best known value have been reached is not much different for ZZ, but all successful results are concentrated on a few instances, while other control strategies seem to have a more regular behavior. Therefore MX and CD appear as more generic and could work properly on other problems. Some instances are notably easier that others (considering the operators used), since they were optimally solved by all strategies in every run, while others by none.

In order to analyze the characteristics of each strategy, we will concentrate on three representative instances that show the behavior of each one of the three strategies (Fig. 4). Relative ranking of the control scheme is indicated in each figure.

Considering CD on tai64c, we can see that, after an initial confusion due to the effect of initial population, the algorithm is able to rise diversity up to the level required by this particular instance. However, since the convergence-escaping heuristic of CD considers only one value of equally fitness-valued generations, it fails to detect excessive convergence when there are several values with fitness close to the local optima, as in ste36a and els19.

ZZ, nevertheless, solves els19 without any difficulty. This is partially accidental, since the starting diversity level agrees with the level required for this instance. Actually, placing the center of oscillation in the mean value of last successful improvements does not guarantee that this will be the right diversity to command. This appears clearly on tai64c. While CD lacks of means to detect sub-optimal multimodality, ZZ lacks of a criterion to well focusing on the right diversity level.

Note that CD and ZZ have opposite behavior w.r.t. diversity change : ZZ moves it continuously during the search, while CD stands quietly at a convenient value. Why CD solves better tai64c and ZZ does with els19 ? One explanation

[4] Using a t-Student test with significance level of 0.05.

**Table 1.** Mean and standard deviation of experimental results

| Instance | MX | CD | ZZ | Best known |
|---|---|---|---|---|
| bur26a | 5430603(2813) | 5429382(2801) | 5431717(1797) | 5426670 |
| bur26b | 3820099(2645) | 3820701(3237) | 3819815(2497) | 3817852 |
| bur26g | 10117735(637) | 10118015(1805) | 10118437(734) | 10117172 |
| bur26h | 7098797(227) | 7101812(11143) | 7099036(314) | 7098658 |
| chr12a | 9552(0) | 9552(0) | 9552(0) | 9552 |
| chr18b | 1534(0) | 1534(0) | 1534(0) | 1534 |
| chr20c | 14980(676) | 15564(967) | 14462(415) | 14142 |
| chr25a | 4282(145) | 4160(193) | 4738(189) | 3796 |
| els19 | 17255411(64685) | 17403382(428255) | 17212548(0) | 17212548 |
| esc32a | 133(1) | 137(3) | 146(4) | 130 |
| esc32b | 174(8) | 183(7) | 190(3) | 168 |
| esc64a | 116(0) | 116(0) | 116(0) | 116 |
| had12 | 1652(0) | 1652(0) | 1652(0) | 1652 |
| had20 | 6922(0) | 6922(0) | 6922(0) | 6922 |
| kra30a | 90618(536) | 90688(671) | 91805(556) | 88900 |
| lipa20a | 3696(21) | 3690(12) | 3695(21) | 3683 |
| lipa40b | 504178(40403) | 509671(41952) | 558748(18977) | 476581 |
| lipa60a | 108419(55) | 108263(46) | 108649(35) | 107218 |
| lipa60b | 3001663(133444) | 3005495(6122) | 3068971(5125) | 2520135 |
| nug15 | 1150(0) | 1150(0) | 1150(0) | 1150 |
| nug20 | 2573(3) | 2571(2) | 2574(6) | 2570 |
| nug30 | 6177(26) | 6156(21) | 6261(25) | 6124 |
| rou20 | 729645(1575) | 728832(1764) | 729987(3362) | 725522 |
| scr20 | 110375(428) | 110129(178) | 110178(255) | 110030 |
| sko42 | 15982(62) | 15962(48) | 16341(61) | 15812 |
| sko64 | 49193(120) | 49051(175) | 50934(209) | 48498 |
| ste36a | 9744(102) | 9704(98) | 10268(148) | 9526 |
| ste36b | 16209(293) | 16341(526) | 16973(254) | 15852 |
| ste36c | 8402215(76340) | 8360860(84502) | 8525723(69745) | 8239110 |
| tai20a | 710633(2832) | 709743(1618) | 712817(2402) | 703482 |
| tai20b | 122562707(222601) | 122824782(270418) | 122667094(266302) | 122455319 |
| tai40a | 3249903(12230) | 3231014(13179) | 3304776(11237) | 3139370 |
| tai40b | 647477626(8092672) | 650707492(12312552) | 654471314(8168843) | 637250948 |
| tai60a | 7615573(33603) | 7468530(19378) | 7722640(22912) | 7205962 |
| tai60b | 617635298(4993344) | 616454128(4540470) | 630041081(4689169) | 608215054 |
| tai64c | 1857916(2066) | 1856511(736) | 1857830(2128) | 1855928 |
| tho40 | 244248(1349) | 244027(1418) | 251943(1598) | 240516 |
| wil50 | 49029(56) | 48994(88) | 49728(80) | 48816 |
| avg. optima | 4.82 | 4.60 | 4.29 | |
| outperf. MX | — | 6 | 2 | |
| outperf. CD | 3 | — | 1 | |
| outperf. ZZ | 20 | 21 | — | |

could be related to the shape of fitness landscape of both instances: if it is smooth, a quiet search, that walks across the "plains" and the "valleys" could be appropriate, while if it is rugged, a "messy search", that first jumps between "peaks" to then concentrate of them could be best suited. The interest in finding

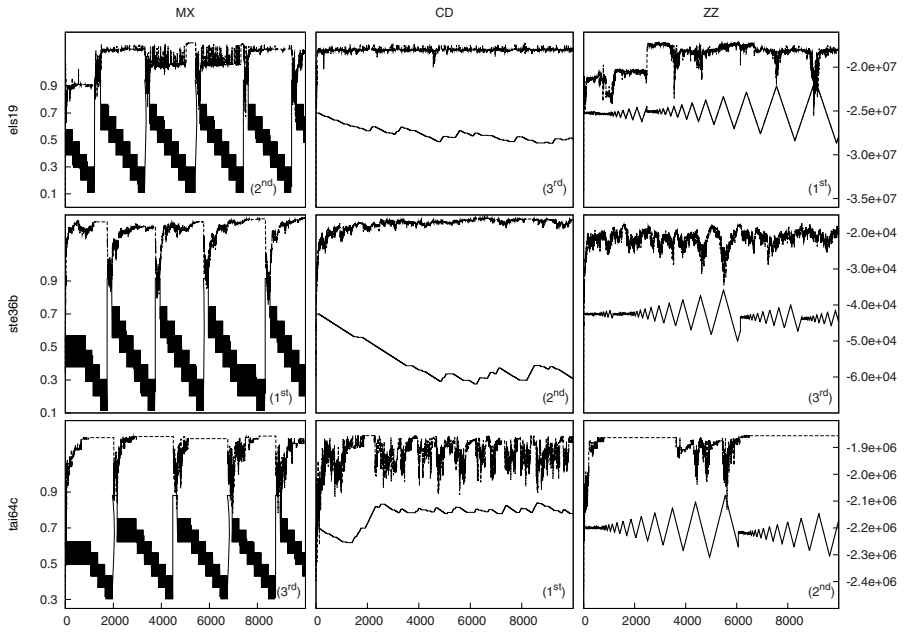


**Fig. 4.** Plot of commanded diversity (below) and fitness of best individual of the population (above) obtained with proposed strategies for representative instances els19, ste36b and tai64c. In parenthesis appears the relative order according to mean result.

out a relationship between fitness landscape and the best suited control strategy lies in the simplicity to know the former, thus it would be possible to automatically select the most performing strategy by measuring ruggedness at the beginning of the search.

In order to check this hypothesis, we have calculated the random walk correlation function, proposed by Weinberger [14]. This function takes a sequence of fitness values from a solution that is randomly modified by an operator, and calculates correlation between fitness values separated by $s$ iterations. We have calculated the correlation for all treated instances with values of $s$ ranging form 1 to 10, and with two operators; exchange mutation and remake, for series 50.000 iterations long. We have found that tai64c has, as we expected, a high level of correlation, revealing a smooth fitness landscape. Most of the instances with a similar correlation structure were best solved by CD (lipa60a, tai60a) and in some of them ZZ's performance was particularly inefficient (lipa60b, sko64, tai60b, wil50). On the other hand, els19 has one of the lowest measures of correlation, pointing out a rugged fitness landscape. The same happens with another instance well solved by ZZ, chr20c. However, the mapping between strategies and fitness landscape is not that clear in this case, since there are some rugged instances that are slightly better solved by CD (rou20, tai20a), and most them are solved similarly by all strategies (chr12a, chr18b, had12, had20, nug15, nug20,

scr20). That could be caused by the inappropriate placing of center in ZZ. Further work is needed before concluding a definitive relationship.

Analyzing MX, we found that it worked exactly as expected when solving ste36b: Diversity descends progressively as fitness rises, and forgetting periods allows to escape from local optima to reach a better one. Even though these general-purpose diversity levels worked relatively well with most instances, they were not high enough to deal with tai64c. Another drawback of this strategy is that it does not consider the converging rate, i.e. even if fitness is rising firmly, when the time to forget is come, the algorithm starts to explore, losing the opportunity to improve the current best solution. The number of generations before entering a forgetting period is a sensible parameter, whose value depends on the problem. Something similar happens with the frequency of diversity change in ZZ.

## 6 Conclusion

In this paper we have presented a method to control multiple EA's parameters. Our purpose was to create an abstraction of algorithmic details in order to allow the definition of high-level control strategies, applicable to a wide range of EAs, regardless of the operators used and the problems being solved.

We have discussed several criteria that should be considered when defining general strategies, and three different schemes, all of them absolutely independent from EA's operators, have been studied. We have considered strategies: that keep a somewhat stable value of EEB, and others that continuously move this value.

An interesting relation between the performance of these strategies and the shape of the fitness landscape has been suggested. This could be used to automatically choose which strategy to apply when faced to a particular problem. A learning component could analyze performance of control strategies over different problems and parameters.

We have considered application rate parameters. It would be interesting to apply this method to other kind of parameters, such as selection pressure or population size.

Future work would also extend to other problems in order to assess the generality of our approach.

## References

1. Michalewicz, Z.: Genetics Algorithms + Data Structures = Evolution Programs (1996)
2. Maturana, J., Saubion, F.: Towards a generic control strategy for EAs: an adaptive fuzzy-learning approach. In: IEEE Congress on Evolutionary Computation (CEC 2007), IEEE, Los Alamitos (2007)
3. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in EAs. In: Parameter Setting in EAs, pp. 19–46. Springer, Heidelberg (2007)

4. Kee, E., Airey, S., Cyre, W.: An adaptive genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 391–397 (2001)
5. Nannen, V., Eiben, A.: Relevance estimation and value calibration of EA parameters. In: Proc. of 20th Int.Joint Conf. on Artificial Intelligence IJCAI-2007, pp. 975–980 (2007)
6. Thierens, D.: Adaptive Strategies for Operator Allocation. In: Parameter Setting in EAs, Springer, Heidelberg (2007)
7. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: Current framework and new trends. Fuzzy Sets and Systems 141(1) (2004)
8. Kulkarni, A.: 3. In: Fuzzy Logic Fundamentals, pp. 61–103. Prentice-Hall, PTR (2001)
9. Piegat, A.: Fuzzy Modeling and Control. Springer, Heidelberg (2001)
10. Buckley, J.J.: Sugeno type controllers are universal controllers. Fuzzy Sets and Systems 53, 299–303 (1993)
11. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man and Cybernetics 22(6), 1414–1427 (1992)
12. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica 25, 53–76 (1957)
13. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the TSP. In: Proceedings of the 2nd Int.Conf.on GAs and their application, USA (1987)
14. Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics 63, 325–336 (1990)