Noname manuscript No.
(will be inserted by the editor)

Enhancing Distributed EAs by a Proactive Strategy

Carolina Salto, Francisco Luna, Enrique Alba

the date of receipt and acceptance should be inserted later

Abstract In this work we propose a new distributed evolutionary algorithm that uses a proactive strategy to adapt its migration policy and the mutation rate. The proactive decision is carried out locally in each subpopulation based on the entropy of that subpopulation. In that way, each subpopulation can change their own incoming flow of individuals by asking their neighbors for more frequent or less frequent migrations in order to maintain the genetic diversity at a desired level. Moreover, this proactive strategy is reinforced by adapting the mutation rate while the algorithm is searching for the problem solution. All these strategies avoid the subpopulations to get trapped into local minima. We conduct computational experiments on large instances of the NK landscape problem which have shown that our proactive approach outperforms traditional dEAs, particularly for not highly rugged landscapes, in which it does not only reaches the most accurate solutions, but it does the fastest.

Keywords proactive behaviour · distributed EAs · migration period

Carolina Salto
Universidad Nacional de La Pampa - CONICET
General Pico, Argentine
E-mail: saltoc@ing.unlpam.edu.ar

Francisco Luna
Departamento de Informática
Universidad Carlos III de Madrid
Leganés, Madrid, Spain E-mail: fluna@inf.uc3m.es

Enrique Alba
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, Spain E-mail: eat@lcc.uma.es

1 Introduction

In distributed evolutionary algorithms (dEAs), the population of tentative solutions for a given optimization problem is structured into a usually small number of independent subpopulations (or islands) that evolve in semi-isolation [3,19]. Subpopulations interact among them after these isolation periods by using a migration operator. In addition to the traditional parameters (population size, mutation probability, and so on), dEAs requires new parameters to control the migration operator. They are usually grouped in the so-called migration policy, that fully defines the communication among the different subpopulations usually with:

- Migration period: the period of isolated evolution of the subpopulations.
- Migration rate: the number of individuals that are transferred.
- Selection/replacement of migrants: the strategies for selecting the individuals that are sent in a migration operation and for replacing the local individuals with the incoming ones.
- Topology: this parameter defines the neighbor of each subpopulation.

It is well known that the performance of evolutionary algorithms, and in particular dEAs, in general is strongly influenced by the parameter choice and that the optimal parameterization varies during the evolution process [10,11]. Consequently, our goal here is to adaptively control the migration policy of a dEA using the novel idea of the algorithm being proactive based on entropy information. These measure provides dynamic information about the stage of the evolutionary search process and the degree of diversity of each subpopulation. Let us further detail the contributions.

From the point of view of adaptation, the parameter control approach [11, **?**] is adopted, which works by starting the algorithm with a given initial configuration that changes during the execution of the algorithm (i.e., online adaptation). These changes in the migration policy are performed based on the entropy [24] of each subpopulation. In order to better analyze the results, only one single migration parameter is under study: the migration period at which the individuals are exchanged among subpopulations. The motivation of centering our proposal on the adaptation of only one parameter of the migration policy is based in that the migration period is the most critical and influential for the migration policy [**?**], because not only determines the coupling between the subpopulations, but also the parallel performance of the dEA.

The adaptation of the migration period is triggered when entropy of each subpopulation satisfies a given condition and is aimed at keeping a good diversity. Thereby, whenever the entropy falls below a given threshold, these subpopulations proactively look for new genetic material by reducing the migration period of the subpopulation that provides this island with more migrants, implying more frequent exchanges of solutions among them. That is, if the topology of the migration strategy is a ring, the island i modifies the migration period of the precedent island $i-1$ asking for more frequent migrations. Analogously, the proposed approach also acts when too much diversity is detected by increasing the migration period belonging to the island $i-1$, thus enlarging the isolation period of the island. The resulting algorithm is a has been called Proact and its novelty relies on its proactivity.

To the best of our knowledge, in the published material on self-adaptation of the migration policy in dEAs, the induced changes solely affect the parameters of the local island, i.e., no intervention is undertaken out of the scope of each island. Our idea here is to proactively reach a controlled entropy (and hence diversity) that enhances the search, but based on new, promising individuals already evolved from a neighboring subpopulation. There is a vast literature on self-adaptation in EAs ([10] has 580 cites in Scopus© by itself), but it is drastically reduced when considering dEAs.

The proactive strategy of Proact has been enhanced by the adaptation of the mutation probability (p_m), as another critical parameter to control the diversity of the population and, therefore, the performance of a dEA. By maintaining the mutation probability into adequate values avoids the premature convergence and the loss of genotypic diversity, important to provide diverse solutions to the subpopulation $(i+1)$. In this case, we also use the entropy information to provide a measure of the diversity present in the subpopulation in order to act proactively in the adaptation of the mutation probability of a given population.

The goal of this paper is then to evaluate Proact in terms of its effectiveness with respect to its homogeneous counterparts, in which the migration period and the mutation probability is fixed and preprogrammed during all the search process . As a testbed, we have used a group instances of the NK landscape problem since they allow us to easily tune the ruggedness of their landscape [1]. This article is an extended version of "Heterogeneity through Proactivity: Enhancing Distributed EAs" presented in First International Workshop on Soft Computing Techniques in Cluster and Grid Computing Systems (SCCG), part of the Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC) [23]. The newly developed research improves upon [23] by the proactive strategy which is now enhanced with the adaptation of the mutation probability to better control the population diversity. The results on NK have shown that the resulting algorithm is able to outperform both their homogeneous counterparts and the previous version of the algorithm.

The rest of the paper is organized as follows. Section 2 describes state of the art. Section 3 describes the principal characteristics of distributed EAs, meanwhile Section 4 introduces essential characteristics of our proactive approach and provides an overview of the algorithms that we considered. Section 5 describes the testbed problem and discusses our evaluation methodology. Section 6 presents the results of experimentation conducted. Finally section 7 concludes the paper and discusses future work.

2 Related work

As stated in previous section, little amount of works related to self-adaptation in dEAs can be found. However, we want to highlight several recent contributions that are related to the ideas presented here. With respect to maintaining the diversity in a dEA, Araujo and Merelo introduced the multikulti algorithm in [6]. It also uses a ring topology in which each subpopulation N_i sends back to subpopulation N_{i-1} a genotype that represents its population (by using either the best individual in N_i or the so-called consensus sequence). When N_{i-1} receives this information, it responds by sending an individual different enough from the individual that represents N_i. No adaptation of the migration policy parameters is carried out. In [14], the authors have evaluated an adaptive migration scheme with three different topologies in which the operation occurs in response

to the current process of the distributed search that is controlled by a central manager. A third relevant work is developed by Osorio *et al.* in [22], in which the migration period is automatically adapted based on a predictive model of the takeover time. To the best of our knowledge, the use of the entropy as the condition to self-adapt the migration policy of a dEA has not been used yet. Entropy has actually been adopted for adapting other parameters in dEAs such as the probability of applying the mutation and crossover operators [30,26], a local search [29], or even the subpopulation sizes [26].

Researchers from the metaheuristics community have also been motivated on the control the mutation probability (p_m). It is very difficult, though not impossible, to find an appropriate parameter setting for p_m in order to reach an optimal performance, because the rates vary along with the problem at hand, even within different stages of the genetic process in a problem. Traditionally, determining what probability of mutation should be used is usually done by means of experience or trial-and-error [8,9,13,21], and these probability values are usually used in dEAs. Some works in the field propose the adaptation of the mutation rate in distributed EAs. The majority of them use the mean fitness of the population as a measure control in the adaptive strategy [18,15,26]. To the best of our knowledge, the combined adaptation of the migration period of a remote island and the local mutation probability to control the population diversity based on entropy has been never addressed in the literature.

3 Distributed EAs

It is widely known that when EAs [7,20] work with a population of individuals of considerable length for a complex problem it usually means a significant utilization of computational resources. A straightforward way to decrease these requirements is to resort to parallel families of EAs. Parallel EAs modify the typical behavior of the equivalent sequential (panmictic) algorithm since they are actually using a structured population in most cases. Among the most well known types of structured EAs, *distributed* (dEA) [28] and *cellular* (cEA) [25,2] algorithms are very popular optimization procedures. In short, separating one single population (panmixia) into several subpopulations (decentralization) is a healthy line of research since not only the resulting techniques are faster in terms of numerical cost (visited points in the problem landscape), but they also can be easily parallelized to perform a larger number of steps per time unit in a cluster of workstations or any other multiprocessor system (parallel environment).

Procedure 1 Elementary Distributed EA (dEA$_i$)

```
t = 0; {current evaluation}
initialize(P_i(t));
evaluate(P_i(t));
while (t < max_generations) do
  P'_i(t) = evolve(P_i(t)); {recombination and mutation}
  evaluate (P'_i(t));
  P'_i(t) = send/receive individuals from dEA_j; {interaction
  with neighbors}
  P_i(t+1) = select new population from P'_i(t) ∪ P_i(t);
  t = t + 1;
end while
```

In particular, in this work we use distributed EAs (dEAs) [28], which is a multi-population (island) model performing sparse exchanges of individuals (migration) among the elementary populations P_i (where i is the identifier of an island). The migration policy must define the island topology, when migration occurs, which individuals are being exchanged, the synchronization among the subpopulations, and the kind of integration of exchanged individuals within the target subpopulations. In this work, we have always in mind asynchronous communications [5] so that when a migration operation takes place, the subpopulations send the corresponding individuals and proceed right to the next iteration without waiting for an incoming solution to arrive.

Algorithm 1 outlines the structure of an elementary evolutionary algorithm (dEA_i) [27]. Each dEA_i randomly generates an initial population P_i of μ solutions, and evaluates them afterwards. Then, the population goes into the evolutionary loop, which means the application of genetic operators, to create λ offspring. This loop includes an additional phase of individual exchange with a set of neighboring algorithms, denoted as dEA_j. Finally, each iteration ends by selecting μ individuals to build up the new population from the population ($\mu + \lambda$). The best solution is identified as the best individual ever found which maximizes the fitness function.

4 A proactive dEA

This Section is aimed at presenting the two algorithmic proposals devised in this work. These proposals include proactive strategies in order to modify the migration period and/or the mutation rate during the evolution. The proactive strategy controls these parameters in order to maintain the genetic diversity at desired levels in each subpopulation, avoiding the converge to a local optimum or the stagnation. This strategy is guided by the entropy information, which measures the genetic diversity present in a population. For that reason, the

sections are organized to follow certain structure in order to explain the following topics: what parameters to adapt, on what evidence to adapt and how to adapt. First, the details of Proact, which proactively adapt the migration period, are motivated and introduced. Then, we present its enhanced version, Proact-PmAdap in which the adaptation of the mutation rate comes into play in order to better control the population diversity.

4.1 Proact

In this section, we explain the proactive strategy followed by Proact, which introduce the adaptation of the migration period. Traditionally, the subpopulations in a dEA exchange information, usually individuals (but any other kind of population statistics can be transferred [4]), in a *blind* way, i.e., in a fixed number of steps. According to the results presented by Tanese [28], performance degrades if migration happens too frequently or too infrequently, such that the migration period is a critical parameter for dEAs. It also occurs, as stated by [10], that the best parameter setting of an algorithm is different depending on the stage of the evolution. These two factors lead to engineering algorithms whose parameterization is automatically modified according to an *intelligent* adaptive strategy. The decisions of this strategy can be guided by some run-time characteristics of the search performed by the algorithm.

Since the subpopulations are searching using their own information, it is easy to converge to a local optimum or to stagnate. These states could be avoided if each subpopulation asks for new genetic material to its neighbors when the beginning of some of those situations is detected.

Considering the previous observations, we propose here the dynamical adjustment of the migration period, *mig_period* (what to adapt), using information from the current state of the search in terms of the (Shannon) entropy metric ($H \in [0,1]$) [24]. The remaining parameters of the migration operation of our approach remains fix during all the evolution. Proact uses an unidirectional ring topology so that each subpopulation p_i only receives/sends individuals from/to subpopulation p_{i-1}/p_{i+1}. The selection and replacement strategies are, respectively, sending the best and always replacing the worst. The migration rate (or the number of individuals involved in each operation) is one.

The working principle of our proactive strategy is as follows. When the H value is close to 1.0 in a given island, we assume that the island has a good and diverse genetic material, so therefore the search has to be further intensified (increase exploitation with less frequent incoming individuals). On the other hand, the strategy tries to promote the exploration (more frequent migrations) when H is close to zero. Let $H(g)_i$ be the entropy value of p_i at generation g. When p_i detects an decreasing diversity (low value of $H(g)_i$), it asks p_{i-1} to send individuals with higher frequency by updating the *mig_period* at p_{i-1}. That is, p_i receives new genetic material proactively by taking into account its actual needs. In this way, p_i acts in advance due to a diversity loss. This is the evidence on what Proact adapts the migration period. These decisions are proactive in the sense that they anticipate some of the scenarios previously described and, in consequence, introduce changes to the migration period. These are the design goals of Proact.

Up to now, we have generally talked about "high diversity" and "low diversity" ("high entropy" or "low entropy", respectively). It is time to define these concepts accurately. Our proactive scheme uses an upper and a lower bound of $H(g)_i$, $\overline{H}$ and $\underline{H}$ respectively, in order to modify the migration period. Therefore, if $H(g)_i > \overline{H}$, Proact increases the value of the migration period in a value equal to the population size (μ value); analogously, if $H(g)_i < \underline{H}$, Proact decreases the period in μ units. Finally, if $\underline{H} \leq H(g)_i \leq \overline{H}$, we assume the search has a controlled entropy and, consequently, the migration period remains without modification. Procedure 2 sketches the proactive strategy followed by each subpopulation of Proact. Of course, the migration periods are assumed to be discrete values in the range $[mig\_period_{min}, mig\_period_{max}]$. Thus, Proact can directly measure and control the migration period.

The execution is initialized by randomly assigning the *mig_period* parameter with multiples of *pop_size* in the range [1,*max_migration_period*] to each island. The bounding cases are the total communication (migration period equals to one) or almost isolated execution, as given by *max_migration_period*. If $mig\_period = 1$ and Proact needs to change to the next lower one, the algorithm does not make any change in the migration period at all. The same action is performed when the $mig\_period = max\_migration\_period$ and the algorithm needs to increase the value to the next one. Each subpopulation has a set of bounds initially determined in a fixed way, which is adaptively adjusted regarding the problem characteristics and the search difficulty.

Regarding the characteristics of each problem and instance, the adaptation of the upper and lower bound ($\overline{H}$ and $\underline{H}$ respectively) values are adapted during the run. The bound values are decreased//increased by some given factor b at the end of each generation. If the $H(g)_i$ value is three consecutive times lower than the lower bound $\underline{H}$, then $\underline{H}$ is decreased in a factor of b. The reason of this update is that the population

Procedure 2 adapt_migration(mig_period);

```
if (H(g)_i > H̄) then
  mig_period = mig_period − μ;
else
  if (H(g)_i < H̲) then
    mig_period = mig_period + μ;
  end if
end if
```

Procedure 3 adapt_bounds($\underline{H}, \overline{H}$)

```
if (H(g)_i > H̄) then
  #_higher + +;
  if (#_higher == 3) then
    H̄ + = H̄ × b;
    #_higher = 0;
  end if
else
  if (H(g)_i < H̲) then
    #_lower + +;
    if (#_lower == 3) then
      H̲ − = H̲ × b;
      #_lower = 0;
    end if
  else
    #_middle + +;
    if (#_middle == 3) then
      H̄ − = H̄ × b;
      H̲ + = H̲ × b;
      #_middle = 0;
    end if
  end if
end if
```

has little genotype diversity so Proact will need to emphasize their needs of new genetic material from its neighbors. On the contrary, if the $H(g)_i$ value is three consecutive times higher than the upper bound $\overline{H}$ then it is increased in a factor of b. In the case that the $H(g)_i$ value falls three consecutive times within the upper and lower bounds, then both bounds are modified: the upper bound is decreased meanwhile the lower one is increased. Procedure 3 shows the mechanism followed by our dEAs to adapt the bounds.

Summarizing, each subpopulation p_i of the Proact sends:

- its migration period to the left neighboring subpopulation in the ring (island p_{i-1}) in a proactive way by using the Procedure 2, and
- the best solution found so far to the right neighboring subpopulation in the ring at a frequency indicated by the p_{i+1} island.

Consequently, there is a double sense of information flow between the islands. Figure 1 outlines this scheme. Moreover, locally each subpopulation p_i adapts the bounds used to make the changes to the migration period. The refined distributed genetic algorithm incorporating this proactive strategy is described in Procedure 4.

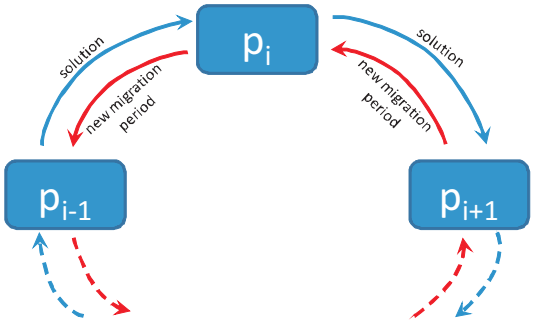

Fig. 1 Outline of the ring topology of Proact.

Procedure 4 Elementary Proact algorithm (Proact$_i$)

```
t = 0; {current evaluation}
initialize(P_i(t));
evaluate(P_i(t));
while (t < max_generations) do
  P'_i(t) = evolve(P_i(t)); {recombination and mutation}
  evaluate (P'_i(t));
  P'_i(t) = send/receive individuals from Proact_j at
  mig_period_toSend rate; {interaction with neighbors}
  P_i(t + 1) = select new population from P'_i(t) ∪ P_i(t);
  adapt_migration(mig_period);
  send/receive the mig_period to/from Proact_{i−1/i+1};
  adapt_bounds(H̲, H̄);
  t = t + 1;
end while
```

4.2 Proact-PmAdap: Proact improved by adaptive mutation

In this work we also considered another parameter to introduce into the proactive strategy: the p_m value. The reason is that it will allow the algorithm to better control the population diversity in a simple, straightforward and efficient way. The p_m value is updated depending on the genetic diversity present in the population (entropy value), in order to improve the performance of the search process during run time. The goal is also to proactively act when a low diversity is expected to appear. The strategy tries to include more genetic diversity in the island i with the purpose of disposing a diverse population from which solutions can be selected in each migration step.

The strategy gives a positive or negative reinforcement to the mutation probability values. In this way, a decrease in the entropy values indicate a lost of diversity, in consequence the mutation probability value will be increased in order to introduce some genetic diver-

sity. Otherwise, the p_m value is decreased. The variation introduced to the p_m value is shown in Equation 1

$$p_m = p_m - \delta * (H(g)_i - H(g-1)_i) \quad (1)$$

where δ is included for providing smooth parameter adjustments, as it is required for any effective dynamic parameter control strategy. Its value is fixed at 0.01.

The algorithm controls that p_m value belongs to $[\underline{p_m}, \overline{p_m}]$, to prevent the overshooting of p_m; where the lower bound of p_m is $\underline{p_m}$=0.001 and the upper bound is $\overline{p_m}$ =0.1. These bounds are selected taking into the account the values suggested in literature [9,13].

The adaptation of p_m has a strong negative correlation with the migration period as it gets increased when the entropy falls below the threshold, whereas the *mig_period* is decreased. Analogously, p_m decreases when a very high entropy is detected, while *mig_period* is increased.

5 Experimentation

In this section we present the necessary information to reproduce the experiments that have been carried out in this research. First we will introduce the problem used to assess the performance of our proposal: the NK-Landscapes, a classical combinatorial optimization problem. Second, we will present the parameters used by our Proact.

5.1 NK-Landscapes

An NK-landscape is a fitness function $f : \{0,1\}^N \rightarrow \Re$ on binary strings [16], where N is the bit string length and K is the number of bits in the string that epistatically interact with each bit. Each gene x_i, where $1 \leq x_i \leq N$, contributes to the total fitness of the genotype depending on the value of its allele and on those of each of the K other genes to which it is linked. Thus K must fall between 0 and N-1. For $K = 0$, there are no interaction among genes and a single-peak landscape is obtained; in the other extreme (for $K = N - 1$), all genes interact each other in constructing the fitness landscape, so a completely random landscape is obtained (a maximally rugged landscape). Varying K from 0 to $N - 1$ gives a family of increasingly rugged multi-peaked landscapes.

The fitness value for the entire genotype is given as the average of the fitness contribution of each locus f_i by:

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots, x_{i_K}) \quad (2)$$

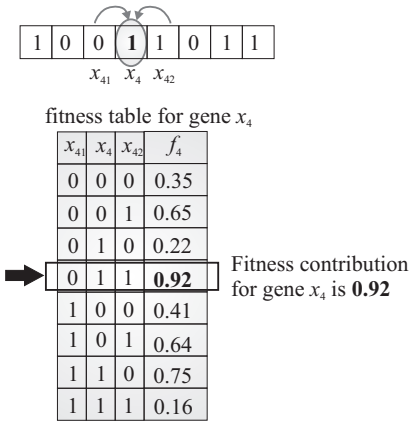

Fig. 2 An example of a genotype and a fitness table associated to gene x_4 for a problem with N=8 and K=2.

where $\{x_{i_1}, \ldots, x_{i_K}\} \subset \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, N\}$ are the K genes interacting with gene x_i in the genotype x. The other K epistatic genes could be chosen in any number of ways from the N genes in the genotype. Kauffman [17] investigated two possibilities: *adjacent neighborhoods*, where K genes nearest to gene x_i on the chromosome are chosen, particularly a gene interacts with $K/2$ left and $K/2$ right adjacent genes; and *random neighborhoods*, where these K other genes are chosen randomly on the chromosome. We adopted the first type of neighborhood and considered circular genotypes to avoid boundary effects. The fitness contribution f_i of x_i is taken at random from a uniform distribution [0.0, 1.0] and depends upon its value and those present in K other genes among the N. Each gene has associated a $fitnesstable$, mapping each of the 2^{K+1} possible combinations of alleles to a random, real value number in the range [0,1]. Figure 2 gives an example of the fitness function $f_4(x_4, x_{4_1}, x_{4_2})$ associated to gene x_4 for $N = 8$ and K=2. Gene x_4 is linked to two other genes, in this case, genes to either side, x_{4_1} and x_{4_2}.

In the experiments of this work, we have used NK instances with N=192 bits varying the epistatic relations from K=8 to K=96. For each combination of N and K we generated 30 random problem instances. Each algorithm were tested with each one of these instances.

5.2 Algorithm parameterization and methodology

In order to evaluate the proactive strategy, we have compared the Proact versions to several homogeneous, non-proactive dEAs that have the migration period parameter fixed. They have been named Hom$<$ *mig_period* $>$, with *mig_period* $\in \{1, 32, 64, 128, 256, 512\}$, that is, from

Table 1 Experimental Parameters

Population size	512 individuals
Number of islands	8 islands
Selection of parents	Binary tournament
Recombination	two-point, $pc = 0.65$
Bit mutation	Bit-flip, $pm = 1/L$
Replacement	Rep_always

maximum coupling among islands to fairly isolated search. For each of the homogeneous algorithms, we also considered a variant considering the dynamical adaptation of the mutation rate, denoted as $< alg_variant >$PmAdap. The common settings for all the algorithms is as follows. The whole population is composed of 512 individuals. They all use a configuration with 8 islands, so each island has a subpopulation with 64 tentative solutions. The maximum number of generations is fixed to 5000.

The tentative solutions for the NK-landscapes are encoded as binary strings. The genetic operators used within the evolutionary loop are binary tournament selection, two point crossover, and bit flip mutation. The crossover rate is set to 0.65. In the case the mutation rates is not adapted, this value is set to $1/L$, where L is the length of the solutions ($L = 192$). Proportional selection is used to build up the next population from the set of $(\mu + \lambda)$ individuals. The values for the lower and upper bound that triggers the proactive actions of Proact, i.e., $\underline{H}$ and $\overline{H}$, has been set to 0.3 and 0.6, respectively. Moreover, the allowed range of movement for the bound values are $[\underline{H}/\overline{H} - 0.1, \underline{H}/\overline{H} + 0.1]$.

All the algorithms considered in this work are stochastic, so a thorough analysis has been performed in order to provided the results with statistical confidence and, therefore, obtain meaningful conclusions. First, 30 independent runs are carried out. We use the non-parametric Kruskal Wallis test, to distinguish meaningful differences among the mean of the results for each algorithm. We have considered a level of significance of $\alpha = 0.05$, in order to indicate a 95% confidence level in the results.

The algorithms are implemented inside MALLBA [12], a C++ software library fostering rapid prototyping of hybrid and parallel algorithms. Our computing system is a cluster of 8 machines with AMD Phenon X3 at 2.66GHz and 2 GB RAM, linked by Fast Ethernet, under Linux with 2.6.27.7 kernel version. Each island is physically run on a separate processor.

6 Results and discussion

In order to organize the presentation, we divide this section in two parts. As the main contribution of this paper with respect to our previous work is the adaptation of the mutation rate, the first section is precisely devoted to analyze the impact produced by that adaptation (as explained in Section 4.2) into the performance of the dEAs. The second part shows the benefits of the proactive approaches when compared with best performing homogeneous dEAs.

6.1 Adaptation vs no adaptation of the mutation rate

This section shows how the adaptation of the mutation rate can impact in the quality of dEAs results. Let us begin with the analysis of the results of the homogeneous dEAs with fixed migration period. From the analysis of the best solutions reached by each algorithm, we can conclude that the dEAs maintaining the probability rate fixed (Hom$< mig_period >$PmAdap) variants perform better than (Hom$< mig_period >$), regardless of the landscape ruggedness (a table detailing the values is not shown because of the homogeneity of the results). These differences are validated by the statistical test carried out, where $p-$values far above 0.05 were obtained. Even it is not the goal of this work to propose a standalone adaptive mutation scheme, this results have been included just to highlight the later synergy induced when this mechanism is endowed with the proactive migration strategy.

In the following, we carry out an analysis of the performance of our proactive approach with fixed mutation rate, Proact, and its enhanced version with adaptive p_m, Proact-PmAdap. Table 2 encloses the results. The most relevant aspects that were measured in this comparison are the following ones: the best fitness value obtained (column *best*), the average objective values of the best found feasible solutions along their standard deviations (column *mean* $\pm\sigma$), the average number of evaluations needed to reach the best value (numerical effort) (column gen_{best}), and the time spent in the search in seconds (column *time*). The last row in Table 2 shows average results over all the instances obtained by each algorithm just as a summary of the trends. The maximum *best* values are printed in bold.

We want to remark that the Proact-PmAdap has significantly outperformed Proact in both aspects: solution quality and numerical effort (p-values well below 0.05). This could be somehow unexpected because is a very different scenario as observed in the previous analysis. In this case, the adaptation of the mutation rate is able to generate a valuable genetic diversity which is useful to the search. This new material is exchanged between subpopulation improving the quality of final solutions, which are found in a lower number of generations (except for instance with K=64 and K=96).

Table 2 Experimental results for the two Proact variants.

	Proact-PmAdap				Proact			
K	*best*	*mean* $\pm\sigma$	gen_{best}	*time*	*best*	*mean* $\pm\sigma$	gen_{best}	*time*
8	**0.7659**	0.7525 $\pm$0.0064	4054.37	17.48	0.7643	0.7297 $\pm$0.0185	4125.33	14.19
16	0.6368	0.6273 $\pm$0.0046	486.80	15.56	**0.7194**	0.6995 $\pm$0.0084	4049.52	14.84
32	**0.7065**	0.6833 $\pm$0.0078	3279.43	18.34	0.6978	0.6753 $\pm$0.0103	3342.10	14.97
48	**0.6828**	0.6671 $\pm$0.0072	2866.47	18.77	0.6775	0.6585 $\pm$0.0099	3080.93	15.36
64	**0.6683**	0.6558 $\pm$0.0054	2782.37	19.35	0.6671	0.6476 $\pm$0.0089	2658.43	15.74
80	**0.6582**	0.6414 $\pm$0.0069	2660.20	19.77	0.6461	0.6360 $\pm$0.0056	2815.33	16.48
96	0.6509	0.6300 $\pm$0.0068	2734.83	17.00	**0.6572**	0.6362 $\pm$0.0086	2318.40	15.49
Mean	0.6887	0.6717	3062.94	18.45	0.6850	0.6639	3056.75	15.37

That is, as stated before, the two mechanisms proposed (proactive migrating strategy and mutation rate), when working together, work much better than when they are applied separately. Finally, as expected the time spent in the search by Proact-PmAdap is a little longer than the case of Proact, mainly because of the cost incurred when the new p_m value is computed each generation.

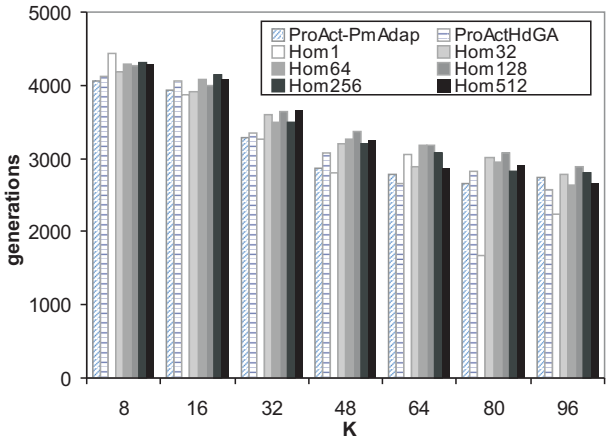

Fig. 3 Number of generations to converge for Proact and homogeneous dEAs.

6.2 Proactive vs homogeneous dEAs

This section compares the two proactive algorithms, Proact and Proact-PmAdapt, and the homogeneous algorithms without the adaptive mutation rate (which have performed the best in the previous section). In a first analysis, the quality of a solution is measured by the percentage gap (see Equation 3) with respect to the fitness of the best solution reached by Proact-PmAdap ($\text{best}_{Proact-PmAdap}$), i.e., the relative distance between the two fitness values. Table 3, which shows the gap values, has been split into two parts: the upper one displays the aforementioned gap with respect to the best solution obtained in the 30 runs (gap_{best}), whereas the lower part includes the gap with respect to the average values (gap_{mean}). Positive values mean that Proact-PmAdap has reached better (higher) fitness values than that the homogeneous counterparts or Proact, while negative values point out the opposite situation.

$$gap_{best} = \frac{best_{Proact-PmAdap} - best_{alg\_variant}}{best_{Proact-PmAdap}} \ . \qquad (3)$$

From the values included in Table 3, we can first observe that the gap_{best} is positive in 43 out of 49 combinations between dEA variants and NK instances. There are, however, negative values (i.e., a homogeneous dEA or Proact has computed a more accurate best solution than Proact-PmAdap), but most of them are very close to zero, meaning that the best solutions obtained by the algorithms are very similar in these cases. When considering the gap_{mean}, Proact-PmAdap has reached the best average values in 47 out of 49 comparisons. The statistical tests have pointed out that Proact-PmAdap have significant differences with the rest of the algorithms for most of the instances (in 4 out of 7 instances). The tendency in a column-wise comparison is clear: the lower the K value, the larger the gap. That is, the higher diversity induced by Proact-PmAdap has largely improved the search, with a deeper impact on not very rugged landscapes, but always allowing to improve upon Proact and all the homogeneous dGAs. The reason for the better performance of the Proact-PmAdap on not very rugged landscapes is that the diversity values induced by the proactive mechanism are too high for very rugged landscapes that prevent the algorithm to converge very fast.

All the previous claims state that the devised algorithm in this paper, Proact-PmAdap, has outperformed its previous version Proact and its homogeneous counterparts in terms of solution quality. But it also has an additional benefit: its convergence speed. Figure 3 shows the average number of generations that all the algorithms considered in this analysis need to compute their best solution. It can be seen that two Proact versions have reported a low number of generations for the majority of the NK instances considered, together with Hom1. The statistical tests have pointed out that differences are significant, except for instances with K=64

Table 3 Gap values for Proact variants and homogeneous dEAs.

K	ProAct	Hom1	Hom32	Hom64	Hom128	Hom256	Hom512
				gap_{best}			
8	0.208	2.416	1.552	3.418	5.089	3.741	4.702
16	1.454	1.236	2.009	-0.080	1.031	2.847	3.419
32	1.240	3.015	1.827	1.620	3.081	3.649	3.671
48	0.773	2.224	-0.201	0.555	0.367	2.238	1.718
64	0.176	0.908	1.684	-0.175	0.524	0.782	1.167
80	1.828	1.881	1.869	-1.618	1.069	0.506	1.652
96	-0.980	0.941	1.442	1.754	1.409	0.301	-0.492
				gap_{mean}			
8	3.035	2.132	2.750	3.295	4.125	3.749	4.034
16	2.164	1.460	1.594	1.837	1.907	2.354	2.847
32	1.171	2.283	1.538	1.155	1.685	2.036	1.675
48	1.298	2.112	1.509	0.730	1.398	1.891	1.782
64	1.241	1.812	1.185	0.994	1.427	1.665	1.825
96	0.196	0.676	0.912	0.813	0.201	-0.255	-0.096

and K=96. The tests also indicate that Proact-PmAdap and Hom1 have significant differences regarding the convergence velocity. Refining a little bit more the discussion, on the one hand, Proact-PmAdap has clearly converged much faster than Proact, so a compromise between solution quality and converged speed arises here. Many scenarios may be considered in which saving as many function evaluations as possible is a must (e.g., when the fitness function involve tasks demanding high computational tasks such as simulations, or optimization problems that require very short response times). On the other hand, our proactive algorithm has not only been the algorithm that reached the best averaged solutions, as mentioned previously, but also the one that reaches them using a low number of evaluations. This, again, points out the suitability of Proact-PmAdap to deal with not very rugged landscapes because of the strategies to control the diversity in its populations.

Regarding the execution time, the results presented in Figure 4 (averaged over the 30 independent runs) show that Proact approach presents slightly longer execution times to the majority of the homogeneous approaches. The longest run times are reported by Proact-PmAdap. The statistical tests have provided these results with confidence.

Summarizing, the results of Proact-PmAdap are very encouraging, since it can compute accurate solutions with high fitness values in a low number of generations.

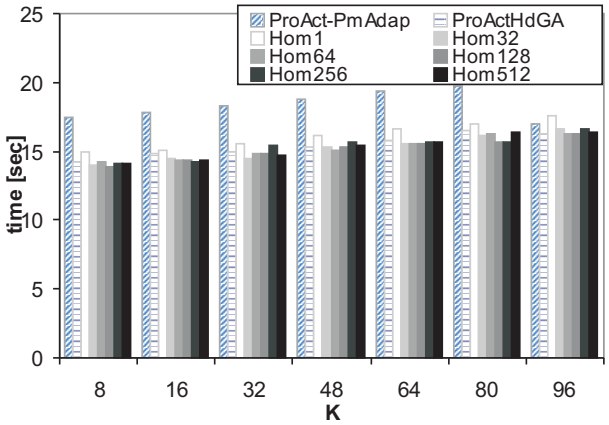

Fig. 4 Execution times of homogeneous dEAs and Proact.

7 Conclusions and future work

In this work, we have introduced Proact-PmAdap, a distributed EA which proactively controls the migration period of the neighbouring subpopulations and the mutation probability in order to maintain a good genetic diversity within each island. The algorithm bases these proactive changes on information of the current state of the search in terms of the Shannon entropy. We have tested our technique on a set of high dimensional NK instances. In order to evaluate the performance of Proact-PmAdap, it has been compared to Proact (a dEA enhanced with a proactive strategy without the adaptation of the mutation rate) and several homogeneous dEAs using different migration periods, fixed after an empirical exploration over different configurations. The results reported have shown that our proactive algorithms are able to reach solutions for the NK-landscape instances with a similar quality than those of the best homogeneous dEAs. Proact-PmAdap has performed specially well on not highly rugged landscapes for which it clearly improves upon homogeneous dEAs, but always allowing to improve upon Proact. An additional truly interesting point is that our proposal exhibits a faster convergence speed, i.e., it has reached their best solutions in a lower number of generations that the majority of the homogeneous dEAs.

We consider that these results are encouraging and they open many promising research lines for future

work. From an algorithmic point of view, we plan to extend Proact for adapting other configuration parameters (from the migration or genetic operators) that allows the algorithm to have a better control of the search. The information used to perform this adaptation is also a matter for future research, not just the entropy, but also other statistics derived from the search history. From the application point of view, we also plan to further evaluate Proact in other domains, specially those in which a fast convergence is required.

Acknowledgements We acknowledge the Universidad Nacional de La Pampa, the ANPCYT and CONICET in Argentina from which the first author receive continuous support. The work of Francisco Luna and Enrique Alba has been partially funded by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01 TIN2011-28194 (the roadME project). Francisco Luna also acknowledges support from TIN2011-28336.

References

1. H. Aguirre and K. Tanaka. A study on the behavior of genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination. *IEICE Trans Fundam Electron Commun Comput Sci*, E86-A(9):2270–2279, 2003.
2. E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*, volume 42 of *Operations Research - Computer Science Interfaces Series*. Springer, 2008.
3. E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443 – 462, 2002.
4. E. Alba and J.M. Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.
5. Enrique Alba and José M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4):451–465, 2001.
6. L. Araujo and J. J. Merelo. Diversity through multiculturality: Assessing migrant choice policies in an island model. *IEEE Transactions Evolutionary Computation*, 15(4):456–469, 2011.
7. T. Bck, D. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. Oxford University Press, New York, 1997.
8. Thomas Bck. Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press, 1992.
9. K. DeJong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.
10. A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions Evolutionary Compututation*, 3(2):124–141, 1999.
11. A. E. Eiben, Zbigniew Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Studies in Computational Intelligence*, volume 54/2007, pages 19–46. Springer, 2007.
12. E. Alba et al. *MALLBA: A Library of Skeletons for Combinatorial Optimisation*, volume 2400 of *LNCS*, pages 927–932. Springer, 2002.
13. J. J. Greffenstette. Optimization of control parameters for genetic algorithms. *IEEE Transaction on System Man and Cybernetic*, 16(1):122128, 1986.
14. M. Hijaze and D. Corne. Distributed evolutionary algorithm topologies with adaptive migration schemes. In *IEEE Congress on Evolutionary Computation*, pages 608–615, 2011.
15. Xue jing Gong and Jian wei Xie. Migration strategy and mathematical analysis of sub-population size adaptation in parallel genetic algorithm. In *Medical Imaging, Parallel Processing of Images, and Optimization Techniques*, volume 7497, pages 1–15, 2009.
16. S. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.
17. S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11–45, 1987.
18. Halina Kwasnicka. An ensemble of cooperative genetic algorithms as an intelligent search tool. *International Journal of Computational Intelligence Research*, 3(2):165–176, 2007.
19. G. Luque and E. Alba. *Parallel Genetic Algorithms: Theory and Real World Applications*, volume 367 of *Studies in Computational Intelligence*. Springer, 2011.
20. M. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third revised edition, 1996.
21. Gabriela Ochoa, Inman Harvey, and Hilary Buxton. On recombination and optimal mutation rates. In *in Proceedings of Genetic and Evolutionary Computation Conference*, pages 488–495. Morgan Kaufmann, 1999.
22. K. Osorio, G. Luque, and E. Alba. Distributed evolutionary algorithms with adaptive migration period. In *2011 11th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 259 – 264, 2011.
23. C. Salto, F. Luna, and E. Alba. Heterogeneity through proactivity: Enhancing distributed EAs. In *1st International Workshop on Soft Computing Techniques in Cluster and Grid Computing Systems (SCCG), part of the Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 279–284, 2012.
24. C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.
25. P. Spiessens and B. Manderick. A massevily parallel genetic algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 279–286, 1991.
26. K. G. Srinivasa, K. R. Venugopal, and L. M. Patnaik. A self-adaptive migration model genetic algorithm for data mining applications. *Information Sciences*, 177(20):4295–4313, 2007.
27. G. Syswerda. Study of reproduction in generational and steady-state genetic algorithms. *Foundations of Genetic Algorithms*, pages 94–101, 1991.
28. R. Tanese. Distributed genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 434–439, 1989.
29. J. Tang, M-H. Lim, and Y-S. Ong. Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. *Soft Computing*, 11(9):873–888, 2007.
30. R. Wang, Y. Ru, and Q. Long. Improved adaptive and multi-group parallel genetic algorithm based on good-point set. *Journal of Software*, 4(4):348–356, 2009.