

Comparison of Different Approaches for Adapting Mutation Probabilities in Genetic Algorithms

Natalia Stark¹, Gabriela Minetti¹, Carolina Salto^{1,2}

¹Laboratorio de Investigación en Sistemas Inteligentes
Facultad de Ingeniería – UNLPam
Calle 110 N° 390 – General Pico, La Pampa
²CONICET
e-mail { nstark, minettig, saltoc }@ing.unlpam.edu.ar

Abstract- Traditionally in Genetic Algorithms, the mutation probability parameter maintains a constant value during the search. However, an important difficulty is to determine a priori which probability value is the best suited for a given problem. In this paper we compare three different adaptive algorithms that include strategies to modify the mutation probability without external control. One adaptive strategy uses the genetic diversity present in the population to update the mutation probability. Other strategy is based on the ideas of reinforcement learning and the last one varies the probabilities of mutation depending on the fitness values of the solution. All these strategies eliminate a very expensive computational phase related to the pre-tuning of the algorithmic parameters. The empirical comparisons show that if the genetic algorithm uses the genetic diversity, as the strategy for adapting the mutation probability outperforms the other two strategies.

1 Introduction

Adaptation is an alternative to determine the optimal value of the parameters to use in a genetic algorithm (GA) during the search. This is a good option, because during the design process, we need to instantiate several parameters to specific values before the run of the algorithm, in order to obtain a robust search technique. Even for an expert user, the parameter configuration for an optimal performance is hard to find.

The parameter control [9] manages information that influences the parameter values during the search and defines a set of criteria to produce changes. In this way, the user does not need to make non-trivial decisions beforehand, which are a very expensive task. Many relevant approaches have been proposed for adjusting GA parameters such as genetic operator probabilities [14,17,18,19], selection methods [5, 11], population size [4, 10, 21,23] during the search. The mutation probability (p_m) is one of the critical parameters in the GA performance. Large values of p_m transform the GA into a purely random search algorithm, while some mutation is required to prevent the premature convergence of the GA to suboptimal solutions.

In this work, we compare the adaptive method proposed by Stark et al. [22] with other two different methods, present in the literature, to adapt the p_m during the GA run. The aim of this comparison is to determine which of them allows to obtain good

quality of results with a low computational effort. In the method proposed by Stark et al. [22], the mutation probability varies depending on the genetic diversity present in the population, by using a control strategy based on the current population entropy. In the comparison the adaptive method proposed by Srinivas et al.[17] is considered, where the probabilities of mutation are changed depending on the fitness values of the solution. This proposal protects high fitness solutions, but solutions with subaverage fitness are totally disrupted. These methods have been selected since they present different ways to vary the p_m and also exhibit good performance. The other method included in the comparison is the one proposed by Riff et al. [14], which is based on the ideas of reinforcement learning: an operator receives a reward in its probability of application when the new generated individual is better than its parents. Analogously, it receives a penalty when the offspring has a worse fitness value than its parents. Both the rewards and the penalties strongly depend on the improvement/degradation of the evaluation function value.

The outline of the paper is as follows. Section 2 details the adaptive algorithms included in this work and a brief comparison among them. Section 3 addresses the optimization problem and the parameter configuration. In the next section, we analyze the results from a computational point of view. Finally, we summarize the conclusions and discuss several lines for future research in Section 5.

2 Adaptive GAs

In this section, we present the characteristics of the adaptive algorithm proposed in [22], denominated Adaptive Mutation Probability Genetic Algorithms (APmGA), which dynamically adjusts the mutation probability (p_m) during the evolutionary process. To assess their behavior we considered two other algorithms from the literature, which also include adaptive control strategies to change p_m . They are: the Adaptive Genetic Algorithms (AGA) proposed by Srinivas, et al. [17] and the Adaptive Control strategy (AcGA) proposed by Riff et al. [14]. These algorithms are the most representative in this area, showing a good performance.

All adaptive control strategies considered can be embedded as a procedure control into a traditional GA, thus not requiring any change of the algorithm itself. The generic structure of the GA used in this work is shown in Algorithm 1. The algorithm creates an initial population P of μ solutions in a random way, and then evaluates these solutions. The population goes into a cycle where it evolves by means of the operator applications such as selection, recombination, and mutation to compute a whole generation of new λ individuals. The new population is built up by selecting μ individuals from the set of $(\mu+\lambda)$ existing ones. After that, each iteration ends with the computation of the statistics of the current population and the application of the adaptive mechanism to dynamically set the mutation probability. Finally, the best solution is identified as the best individual ever found that maximizes the fitness function.

Algorithm 1 Traditional GA()

```
t = 0; {current generation}
initialize(P(t));
evaluate(P(t));
while{(t < maxgenerations) do
    P'(t) = selection(P(t));
    P'(t) = recombine(P'(t));
    P'(t) = mutate(P'(t), pm);
    evaluate (P'(t));
    P(t+1) = Replace(P'(t) U P(t));
    statistics(P(t+1));
    AdaptivePmControl(pm);
    t = t + 1;
end while
return best_solution
```

2.1 APmGA

The objective of the adaptive strategy of APmGA is to increase p_m if the genetic diversity is gradually lost in order to maintain a population distributed in the search space. Otherwise, the p_m value is reduced when an increase in the population diversity is observed. Therefore, these changes in the p_m value are also an additional source for a good balance between exploration and exploitation, intrinsic to the adaptive criterion. Furthermore, this strategy monitors the genotypic diversity present in the population using the Shannon entropy metric [20], following the ideas presented in [2]. When the entropy value is closed to 0, the population contains identical individuals; otherwise it is positive and maximal when all the individuals in the population are different.

The control strategy calculates the population entropy at the end of each *epoch* (a certain number of consecutive generations). Then, it computes the variation of the current entropy at the k epoch (H_k) respect the entropy from the previous $k-1$ epoch (H_{k-1}), denoted as $\Delta H_k = H_k - H_{k-1}$. This variation, ΔH_k , is compared with the one observed in the previous epoch ($\Delta H_{k-1} = H_{k-1} - H_{k-2}$), as shown in the Algorithm 2. In this way, if the entropy variation decreases at least in a factor of ε , this indicates the lost of diversity, in a consequence the mutation probability value will be changed by adding a constant factor (α). Otherwise, the p_m value is decreased by subtracting an α value. In order to prevent the overshooting of p_m , the strategy controls that p_m belongs to $[p_m\text{LB}, p_m\text{UB}]$; where the lower bound of p_m is $p_m\text{LB} = 0.001$ and the upper bound is $p_m\text{UB} = 0.1$. These bounds are selected considering the values suggested in literature. We consider the factor α as 0.001 and ε as 0.5. These values have been “optimized” through a previous hand-tuning process comparing it versus other different values.

Algorithm 2 AdaptivePmControl($\Delta H_k, \Delta H_{k-1}, p_m$) function

```
if  $\Delta H_k < (1+\varepsilon) \cdot \Delta H_{k-1}$  then
     $p_m = p_m + \alpha$ 
else
     $p_m = p_m - \alpha$ 
```

2.2 AGA

Srinivas et al. present an adaptive strategy to control the p_m in order to prevent the GA to be stuck at a local optimum. The value of p_m varies depending on the value of $f_{max} - \bar{f}$ (where \bar{f} is the average fitness value of the population and f_{max} the maximum fitness value of the population) and on the fitness value f of a solution. If f is close to f_{max} , p_m should be set to a smaller value in order to not disrupt good solutions. Consequently, they proposed to use different p_m values depending on the solution quality, as shown in Algorithm 3. For the solution with the maximum fitness, p_m is zero. For a solution with $f = \bar{f}$, $p_m = k$. For solutions with sub average fitness value ($f < \bar{f}$), p_m might assume values larges to 1.0, so the authors impose the following constraint: $p_m = k$ if $f < \bar{f}$, where $k \leq 1.0$. The k value has to be less than 1.0 to constrain p_m to the range $[0.0, 1.0]$ [17]. In particular, we have set k equal to 0.5, taking into the account the value used in the original work.

2.3 AcGA

This strategy computes the mutation probability value for the next generation (pm_{t+1}) considering the performance of the operator, which is measured by the average of the improvements/degradations ($\overline{Sa(O)}$) obtained after its application considering the last l generations. The improvements/degradations are considered as a proportional factor depending on the maximum improvement (max_imp) or degradation (max_deg). The success measure for an operator ($Sa(O)$) in its i^{th} application is computed as the difference between the fitness of the generated child and the average fitness of its parents. This procedure is shown in Algorithm 4, where r is included for providing smooth parameter adjustments, as it is required for any effective dynamic parameter control strategy [14]. This value is fixed at 0.5.

2.4 Comparison between the adaptive GA approaches

Both APmGA and AcGA use control strategies at a population level. Thus, the mutation probability values are the same for all of the individuals in the current

Algorithm 3: Adaptive Genetic Algorithms (f_{max}, \bar{f}, pm) function

```

if  $f \geq \bar{f}$  then
     $pm = \frac{k(f_{max} - f)}{f_{max} - \bar{f}}$ 
else
     $pm = k$ 

```

Algorithm 4: AdaptiveControl($\overline{Sa(O)}$, max_imp, max_deg, Pm_t) function

```

if  $\overline{Sa(O)} \geq 0$  then
     $pm_{t+1} = pm_t + r * \frac{\overline{Sa(O)}}{max\_imp}$ 
else
     $pm_{t+1} = pm_t + r * \frac{\overline{Sa(O)}}{max\_deg}$ 

```

population. In the case of AGA, a different p_m is computed for each solution depending on its fitness value. Another difference is the way to update the p_m : AcGA and AGA consider the fitness values of individuals or population, but ApmGA uses the genetic diversity of the population.

These methods can be implemented without adding any significant overhead and without introducing any major changes to its original algorithm design. The decisions made during the search are based on the current information available from a monitoring process, allowing the algorithm to trigger changes when deemed necessary, based on the potential advantages that such changes could bring.

3 Experimental Setup

In this section we present the necessary information to reproduce the experiments that have been carried out in this work. First, we will introduce the problem used to assess the performance of our proposals: the NK landscape, which has been the center of several theoretical and empirical studies both for the statistical properties of the generated landscapes and for their GA-hardness [1, 8]. Second, we will justify the parameters that the adaptive GAs will use.

3.1 NK-Landscapes

An NK-landscape [12] is a fitness function $f: \{0,1\}^N \rightarrow R$ on binary strings, where N is the bit string length and K is the number of bits in the string that epistatically interact with each bit, i.e., K stands for the number of other genes that epistatically affect the contribution of each gene to the overall fitness value of the string. 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 the average of the fitness contribution of each locus f_i by Equation 1:

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

where $\{x_{i_1}, \dots, x_{i_k}\} \subseteq \{x_1, \dots, x_{i-1}, x_{i+1}, \dots, N\}$ 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 [13] investigated two possibilities: *adjacent neighbourhoods*, 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 neighbourhoods*, where these K other genes are chosen randomly on the chromosome. In this work, we adopted the first type of neighbourhood and considered circular genotypes to avoid boundary effects.

3.2 Parameters

The parameter setting applied to the GAs is the following. The whole population (μ) is composed of 32 individuals. By default, the initial population is randomly generated. In each iteration, the number of created offsprings (λ) is 64. The maximum number of generations is fixed at 10000. Each parent is selected by binary tournament selection. The uniform crossover operator is applied with a probability of 0.65. Bit-flip mutation is used. Fitness proportional selection and elitism are used to build up the next population from the set of $(\mu + \lambda)$ individuals. These parameters (population size, stop criterion, probabilities, etc.) are chosen after an examination of some values previously used with success in [1]. The initial probability mutation is $1/N$ for all algorithms. In Table 1, we summarize the whole setting of parameters values used in these experiments.

We conduct our study on NK instances with $N=96$ bits varying the epistatic relations from $K=0$ to $K=48$ in increments of 4, adding up to 13 instances. We use landscapes with adjacent epistatic patterns among genes. For each combination of N and K we have generated 30 random problem instances.

Due to the stochastic nature of the algorithms, the final results are obtained after averaging the running times of 30 independent runs. A statistical analysis has been performed in order to provide the results with statistical confidence and, therefore, obtain meaningful conclusions. We use the non-parametric Kruskal-Wallis test, to distinguish meaningful differences between the mean results of all algorithms. We have considered a level of significance of $\alpha = 0.01$, in order to indicate a 99% confidence level in the results.

The algorithms are implemented inside MALLBA [3], a C++ software library fostering rapid prototyping of hybrid and parallel algorithms. The strategy control proposed by Riff [14] and Srinivas et al. [17] were implemented by replicating the ideas and configurations proposed in their works. Our computing system is an Intel CI7 2600 at 3.40GHZ and 4 G RAM, under SUSE Linux with 3.1.0-1.2 kernel version.

4 Experimental Results

In this section, we analyze the performance of the three adaptive GA considered in the comparison: APmGA [22], AGA [17] and AcGA [14]. Our aim is to offer an analysis from the accuracy (quality) and performance points of view. For this

Table 1: Parametric values used for the different GAs

<i>Parameter</i>	<i>Value</i>
Parents selection	Binary tournament
μ	32
λ	64
Crossover Operator	UX
Crossover Probability	0.65
Mutation Operator	Bit-Flip
Replacement Selection	The best $(\mu + \lambda)$
Stop conditions	10.000 generations

purpose, this section consists of two parts. First, we analyze the solution quality obtained by the studied algorithms. In the second part, we study the computational effort of each algorithm.

Table 2 presents the solution quality that is measured by the percentage gap, i.e., the relative distance to the best solution obtained by APmGA ($best_sol_{APmGA}$) and the best solution of each of the other algorithms, as described in Equation 2.

$$gap_{best} = \frac{best_sol_{APmGA} - GA_i}{best_sol_{APmGA}} \times 100 \quad (2)$$

From the results, APmGA is better than the other two adaptive GAs, from low to high levels of epistasis, which suggests that APmGA obtains solutions with higher fitness than the other two approaches. The non-parametric tests suggest that there are statistical differences between the APmGA and the rest of the algorithms (p -value less than $2.2e-16$). Boxplot summaries for each algorithm can be found in Figure 1, where each boxplot contains the best found solutions from 30 different runs for all instances. APmGA obtains a median value lower than the rest of the algorithm with an interquartile amplitude smaller, indicating that the best solutions are not disperse (homogeneity in the data set). Previous observations suggest that the adaptive strategy used by APmGA improves the behaviour of the GA, avoiding the premature convergence and the lost of genetic diversity.

Finally, Figure 2 shows boxplots corresponding to the evaluations to obtain the best solution for the three adaptive approaches. AGA is the algorithm with the highest numerical effort, in order to find their best solutions. The differences of the computational effort between AcGA and APmGA are not statically significant. Considering these results and the improvement in the solution quality shown by the APmGA, we can conclude that this algorithm is the best strategy to adapt the mutation probability.

Table 2: Average values of the best found solution for each instance (the best values are bolded).

Inst	APmGA	AGA	AcGA
96-0	0,04	0,08	0,04
96-4	0,04	0,12	0,07
96-8	0,06	0,15	0,08
96-12	0,04	0,12	0,06
96-16	0,06	0,11	0,07
96-20	0,03	0,09	0,05
96-24	0,05	0,09	0,06
96-28	0,05	0,07	0,05
96-32	0,04	0,07	0,05
96-36	0,06	0,08	0,06
96-40	0,03	0,04	0,04
96-44	0,05	0,06	0,05

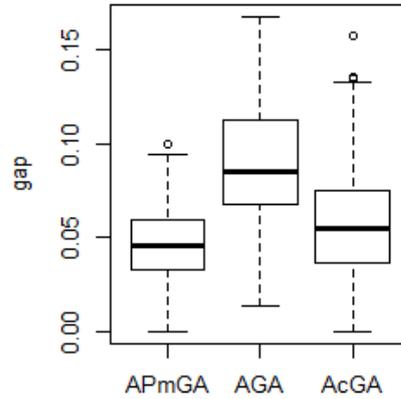


Fig. 1. Boxplot of the gap values for each algorithm

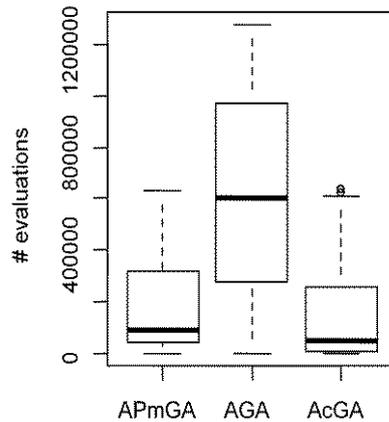


Fig. 2. Boxplot of the evaluations needed to obtain the best solution for each algorithm

Summarizing, the best results are obtained when the mutation probability adaptation is based on the genotypic diversity, as implemented in APmGA. The application of this strategy allows to increase or to keep the population diversity according to the entropy value. Furthermore, no increments in the computational effort are observed when this algorithm is run.

5 Conclusions

In this paper, we analyzed different adaptive evolutionary algorithms which add a control strategy to modify the mutation probability value during the evolution without external control. The control strategies considered differ on the information of the search considered to update the probability value. The different proposals reduce the pre-tuning time to determinate the most appropriate mutation probability value to enhance the algorithm performance.

The results obtained by APmGA compared with AcGA and AGA are very encouraging from low to high levels of epistasis, since it can obtain high quality solutions with a competitive computational effort. Thus, the control strategy using the genotypic diversity as the base to introduce changes in the mutation probability outperforms the other two using the fitness solution.

As a future work, we propose to include the recombination probability in this process. Furthermore, another extension to this work should be the analysis of the effectiveness of this adaptive approach over other complex problems.

Acknowledgements

We acknowledge the Universidad Nacional de La Pampa and the ANPCYT (Argentina) from which the authors receive continuous support. C. Salto thanks CONICET of Argentina.

References

- [1] H. Aguirre and K. Tanaka. Genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination, in *EvoWorkshops 2003*, LNCS 2611, pp. 131–142, 2003.

- [2] E. Alba and B. Dorronsoro. The Exploration/Exploitation Tradeoff in Dynamic Cellular Genetic Algorithms, in *Proceedings of IEEE Transactions on Evolutionary Computation*, vol. 9(2), pp. 126-142, 2005.
- [3] E. Alba, F. Almeida, M.J. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, J. Luna, L. Moreno, C. Pablos, J. Petit, A. Rojas, F. Xhafa. MALLBA: A Library of Skeletons for Combinatorial Optimisation”, in *Proceedings of the 8th International Euro-Par Conference on Parallel*, pp. 927–932, 2002.
- [4] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS A genetic algorithm with varying population size, in *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 73-78, 1994.
- [5] J. E. Baker. Adaptive selection methods for genetic algorithms, in *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, pp. 101–111, 1985.
- [6] M. Birattari, T. Stutzle, L. Paquete, and K. Varrenttrapp. A racing algorithm for configuring metaheuristics, in *Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, pp. 11–19, 2002.
- [7] K. A. De Jong. An analysis of the behavior of a class of genetic adaptive systems, Ph.D. dissertation, Univ. of Michigan, Ann Arbor, MI, 1975.
- [8] K.A De Jong, M.A. Potter, and W.M. Spears. Using problem generators to explore the effects of epistasis, in *Proceedings of 7th International Conference on Genetic Algorithms*, pp. 338-345, 1997.
- [9] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter Control in Evolutionary Algorithms, in *IEEE Transactions on Evolutionary Computation.*, vol. 3(2), pp. 124–141, 1999.
- [10] A.E. Eiben E. Marchiori, and V.A. Valko. Evolutionary Algorithms with on-the-fly pop size adjustment, in *Proceedings of Parallel Problem Solving from Nature 2004 (PPSN 2004)*, vol. 3242 of LNCS , pp. 41-50, 2004.
- [11] A.E. Eiben, M.C. Schut, and A.R. de Wilde. Boosting Genetic Algorithms with Self-Adaptive Selection, in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pp. 16-21, 2006.
- [12] S. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.
- [13] S. A. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes, in *Journal of Theoretical Biology*, vol. 128, pp. 11-45, 1987.
- [14] E. Montero and M-C Riff. On-the-fly calibrating strategies for evolutionary algorithms, in *Information Sciences*, vol. 181(3), pp. 552-566, 2011.
- [15] V. Nannen and A. Eiben, Relevance estimation and value calibration of evolutionary algorithm parameters, in *Proceedings of the Joint International Conference for Artificial Intelligence (IJCAI)*, pp. 975–980, 2006.
- [16] E. Ridge and D. Kudenko. Tuning an algorithm using design of experiments, in *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 265 –286, 2010.
- [17] M. Srinivas and L.M. Patnaik, Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, in *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24 (4), pp. 656-667, 1994.
- [18] J.E. Smith and T.C. Fogarty. Operator and parameter adaptation in genetic algorithms, in *Soft Computing*, vol. 1(2), pp. 81–87, 1997.
- [19] D. Thierens. Adaptive strategies for operator allocation, in *Parameter Setting in Evolutionary Algorithms*, Springer, pp. 77–90, 2007.
- [20] C.E. Shannon. A mathematical theory of communication, in *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.
- [21] X.H. Shi, L.M. Wan, H. P. Lee, X. W. Yang, L. M. Wang, and Y. C. Liang. An Improved Genetic Algorithm with Variable Population-Size and a PSO-GA Based Hybrid Evolutionary Algorithm, in *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, pp. 1735-1740, 2003.

- [22] N. Stark, G. Minetti, and C. Salto. A New Strategy for Adapting the Mutation Probability in Genetic Algorithms, in Proceedings of the Congreso Argentino de Ciencias de la Computación (CACIC 2012), pp. 51-59, 2012.
- [23] D. Voosen and H. Mühlhenbein. Adaptation population size by competing subpopulations, in Proceedings of the 1996 IEEE Conference on Evolutionary Computation, pp. 330-335, 1996.