Optimizing the Individuals Maturation for Maximizing the Evolutionary Learning

T. Calenda[1], A. Vitale[2], A. Di Stefano[2], V. Cutello[1] and M. Pavone[1]

[1] Department of Mathematics and Computer Science
University of Catania
V.le A. Doria 6, I-95125 Catania, Italy
{cutello, mpavone}@dmi.unict.it

[2] Department of Electric, Electronics and Computer Science
University of Catania
v.le A. Doria 6, I-95125, Catania

1 Extended Abstract

As it is well known in the natural computing field, one of the major successful factors in evolutionary algorithms is the design and development of the exploration and exploitation mechanisms. A good balancing between these two phases is crucial since it strictly affects the efficiency and robustness of the evolutionary algorithms performances. While the aim of the exploration mechanism is to search for new solutions in new regions by using the mutation operator, the second mechanism has the purpose to exploit in the best possible way all information gathered using the selection process. Both phases, hence, help the algorithm in discovering, gaining and learning new information, and, subsequently, in exploiting all gained promising regions so to generate better populations.

However, what allows to take advantage of the acquired information is truly given by how long each individual lives and in doing so influencing the evolution and maturation of the population. Besides, this lifetime affects, also, the exploration phase, allowing having a better and deep search process. Thus, the time an individual remains in the population becomes crucial in the performances of any evolutionary algorithm, and it is strictly related to the good balancing between the exploration and exploitation processes. Indeed, letting individuals live for a long time produces a dispersive search, and, then, an unfruitful learning, with the final outcome of increasing the probability to easily get trapped in local optima due to the low diversity that is generated. On the other hand, allowing a short lifetime often does not help to have enough overall learning of the knowledge discovered, and it neither allows a careful search within the solutions space, producing instead high diversity into the population, which, in turn, negatively affects the convergence towards a global optimum.

A first research work on this aspect was conducted in [3], where the authors presented an experimental study whose main aim was to understand the right lifetime of any individual/solution in order to perform a proper exploration within the search space, as well as a fair exploitation of the gained information. Such experimental analysis was conducted on an immunological algorithm, whose core components are the cloning, hypermutation and aging operators.

Table 1. Age assignment options.

Type	Symbol	Description
0	$[0:0]$	age zero
1	$[0:\tau_B]$	randomly chosen in the range $[0:\tau_B]$
2	$[0:(2/3\ \tau_B)]$	randomly in the range $[0:(2/3\ \tau_B)]$
3	$[0:inherited]$	randomly in the range $[0:inherited]$
4	$[0:(2/3\ inherited)]$	randomly in the range $[0:(2/3\ inherited)]$
5	$inherited$ or $[0:0]$	inherited; but if constructive mutations occur then type 0
6	$inherited$ or $[0:\tau_B]$	inherited; but if constructive mutations occur then type 1
7	$inherited$ or $[0:(2/3\ \tau_B)]$	inherited; but if constructive mutations occur then type 2
8	$inherited$ or $[0:inherited]$	inherited; but if constructive mutations occur then type 3
9	$inherited$ or $[0:(2/3\ inherited)]$	inherited; but if constructive mutations occur then type 4
10	$inherited-1$	same age of parents less one

In the cited research work, eleven different options about the lifetime of each individual were studied (see table 1), with the main goal to answer the three main questions: (i) *"is the lifespan related to the number of offspring generated?"*; (ii) *"is the lifespan related to the population size?"*; and in case of negative answer to the two previous ones, (iii) *"how long must the lifespan of an offspring be to carry out a proper exploration?"*. Once these questions were answered , an efficiency ranking was produced, from which clearly emerged that a too short lifetime (parent's age less 1 - "$type10$" of table 1) is always the worst choice; whilst the best 4 are, respectively: "$type0$;" "$type4$;" "$type3$;" and "$type2$;". Thus, following the above described study, in this research work we want to check if the achievements produced on the immunological algorithm (IA) are still valid, and work, on a genetic algorithms (GA). Of course, what we do not expect to get the same efficiency ranking, but rather we would like to check if the top 4 for IA still appear as the top 4 for GA, even if in different ranking order, and, moreover, if the worst for IA continues to still be the worst for GA.

The tackled Problem: to validate and generalize the obtained results, it is crucial to develop an algorithm which is not tailored to a specific problem, by keeping it unaware of any knowledge about the domain. As it is well-known in literature, to tackle and solve generic and complex combinatorial optimization problems, any evolutionary algorithm must incorporate local search methodologies, used as refinement and improvement of the fitness function, and this means that they have to add knowledge about the features of the problem and application domain. This, consequently, makes the algorithm unsuitable and inapplicable to any other problem. To overcome this limitation and make the outcomes as general as possible, in this study we tackle the classic *One–Max* (or *One–Counting*) problem [5, 2] (as done in [3]). *One–Max* is a well-known toy problem, used to understand the dynamics and searching ability of a stochastic algorithm

[4]. Although it is not of immediate scientific interest, it represents a really useful tool in order to well understand the main features of the algorithm, for example: what is the best tuning of the parameters for a given algorithm; which search operator is more effective in the corresponding search space; how is the convergence speed, or the convergence reliability of a given algorithm; or what variant of the algorithm works better [1]. It is worth emphasizing that a toy problem gives us a *failure bound*, because a failure occurs in toy problems at least as often as it does in more difficult problems. *One-Max* is simply defined as the task to maximize the number of 1 of a bit-string $\boldsymbol{x}$ of length ℓ:

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell} x_i,$$

with $x_i \in \{0, 1\}$. In order to validate our studies and our outcomes we have set $\ell = 10,000$ in all experiments.

Genetic Algorithm: in this work a classical genetic algorithm has been developed, which is based on the *uniform crossover*, and *flip mutation* operator, where each individual has probability $P_m = 0.4$ to have one randomly chosen bit to be flipped. However, in order to adapt the GA on this experimental study, to each individual/offspring we assigned an age that determines its lifetime into the population until it reaches the maximum age (τ_B), which is determined by a user-defined parameter. In a nutshell, each chromosome is allowed to remain for a number of generations determined by the assigned ageuntil it reaches the value τ_B. Whenever an offspring is generated, it is is assigned a given age, chosen from Table 1, which is incremented by 1 at each generation. Instead, to every individual of the initial population it is always assigned age zero, regardless of the age assignment chosen. It is important to highlight that in general, Crossover and Mutation do not affect the age of any individual, except for the $5 - 9$ options of the age assignment types, where the assigned age is updated only if its fitness value improves after the performance of the genetic operators. Once the maximum age allowed is exceeded for a chromosome, this will be removed from the population via the aging operator, regardless of its fitness function value. However, an exception is allowed only for the best solution found so far (elitist variant). This variant helps the algorithm keep track of the most promising region - which would otherwise be lost - and whose exploitation might be useful in solving some specific kinds of problems. The last operator performed is the selection operator, which identifies the best elements from the offspring set and the old parents, guaranteeing monotonicity in the evolution dynamics. Nevertheless, due to the aging operator, it could happen that the number of individuals which survived (d_s) is less than the input population size (d). In this case, the selection operator randomly generates $d - d_s$ new individuals. The age assignment together with the aging operator, have the purpose to reduce premature convergences, and keep high diversity into the population. It is worth emphasizing again that the choice of what age value to assign plays a central role on the performances of the GA designed (and of any evolutionary algorithm), since, from it, it depends the evolution and maturation of the solutions. What age to assign is then the focus of this research work.

Results and Conclusions: as described above, several experiments have been performed with the main aim to check if the outcomes highlighted and obtained on the immunological algorithm are still valid on a Genetic Algorithm. If so, this allows us to give a rough indication on the needed lifetime to a solution to have a proper balancing between exploration and exploitation in order to maximize the evolutionary learning. For these experiments we took into account the same experimental protocol used in [3], but reducing the string length to $\ell = 2000$. Unfortunately, for higher values than $\ell = 2000$ the GA is not able to reach the optimal solution. All age assignment options in table 1, have been studied varying the population size $(pop\_size = 50, 100)$ and $\tau_B = 5, 10, 15, 20, 50, 100, 200$, fixing as termination criteria $Tmax = 10^5$ (maximum number of fitness function evaluations), and, finally, each experiment was computed on 100 independent runs. Furthermore, GA was studied in both its variants: *elitist* and *no_elitist*.

By analyzing all the results it is possible to assert that the worst age assignment types on IA continue to be the worst even on GA, and in particular in the last two positions appear $''type6''$ and $''type10''$ respectively. Likely their bad performances are due to the high diversity they produce, not allowing a relevant lifetime to perform a good exploration. Moreover, we may also assert that the top 4 options produced by IA, in most cases, are still in the top 4 of the efficiency ranking produced by GA, although never in the same order.

References

1. V. Cutello, A. G. De Michele, M. Pavone: "*Escaping Local Optima via Parallelization and Migration*", VI International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO), Studies in Computational Intelligence, vol. 512, pp. 141–152, 2013.
2. V. Cutello, G. Narzisi, G. Nicosia, M. Pavone: "*Clonal Selection Algorithms: A Comparative Case Study using Effective Mutation Potentials*", 4th International Conference on Artificial Immune Systems (ICARIS), LNCS 3627, pp. 13–28, 2005.
3. A. Di Stefano, A. Vitale, V. Cutello, M. Pavone: "*Document How long should offspring lifespan be in order to obtain a proper exploration?*", 2016 IEEE Symposium Series on Computational Intelligence (SSCI), INSPEC number 16670548 2016, pp. 1–8, 2016.
4. A. Prugel-Bennett, A. Rogers: "*Modelling Genetic Algorithm Dynamics*", Theoretical Aspects of Evolutionary Computing, pp. 59-85, 2001.
5. J. D. Schaffer, L. J. Eshelman: "*On crossover as an evolutionary viable strategy*", 4th International Conference on Genetic Algorithms, pp. 61–68, 1991.