How long should Offspring Lifespan be in order to obtain a proper exploration?

Antonino Di Stefano, Alessandro Vitale, Vincenzo Cutello, Mario Pavone

***Abstract*— The time an offspring should live and remain into the population in order to evolve and mature is a crucial factor of the performance of population-based algorithms both in the search for global optima, and in escaping from the local optima. Offsprings lifespan influences a correct exploration of the search space, and a fruitful exploiting of the knowledge learned. In this research work we present an experimental study on an immunological-inspired heuristic, called OPT-IA, with the aim to understand how long must the lifespan of each clone be to properly explore the solution space. Eleven different types of age assignment have been considered and studied, for an overall of 924 experiments, with the main goal to determine the best one, as well as an efficiency ranking among all the age assignments. This research work represents a first step towards the verification if the top 4 age assignments in the obtained ranking are still valid and suitable on other discrete and continuous domains, i.e. they continue to be the top 4 even if in different order.**

I. Introduction

The terms *immunological-inspired computation* identify nowadays a wide family of successful algorithms in searching and optimization, inspired by the working of the immune system (IS). The IS defense dynamics and features are a huge source of inspiration. The IS is indeed the most robust and efficient recognition system, able to detect and recognize the invaders, and distinguish between own cells, and foreign ones (*self/nonself discrimination*).

Some other really interesting features of IS allow to design efficient solving methodologies, such as high ability in learning; memory usage; self-regulation; associative retrieval; threshold mechanism, and the ability to perform parallel, and distributed cognitive tasks. In light of the above, immunological heuristics are mainly focused on three main theories: (1) clonal selection [6], [5] (2) negative selection [13]; and (3) immune networks [16]. Such algorithms have been successfully employed in a variety of different application areas.

It is well known that to have a successful population-based algorithm, is important and crucial to have a proper balancing between exploration and exploitation mechanisms, usually represented by the perturbation and selection operators. However, one aspect that is rarely considered but is also very crucial and strictly related to their good balancing is determining for how long a B cell (or offspring in general) remains, evolves, and matures within the population. Having a short life does not allow a careful search, nor learning from the knowledge gathered during the search process, with the final outcome of having a higher diversity that does not always help in finding the optimal solution. On the other hand, a life which is too long might lead to a dispersive search, and an unfruitful exploitation of the solutions, with the final outcome of having lower diversity that does not help the algorithm to jump out of local optima. Thus, in this research work we present an experimental study whose main aim is to understand how much time is enough and needed for an offspring to remain into the population so to have the right maturation and to perform a proper exploration in the search space, and a fair exploitation of the gained information. For this study we have developed an immunological algorithm based on the clonal selection principle (see section II), and presented in [2], [12], whose core components are the cloning and hypermutation operators: the first triggers the growth of a new population of high-value B cells centered on a higher affinity value; whereas the last can be seen as a local search procedure that leads to a faster *maturation* during the learning phase.

For carrying out this study in a way which is as general as possible, it is crucial to develop an algorithm not tailored to a specific problem; in a nutshell, to maintain the algorithm unaware on the knowledge of the domain. On the other hand, it is well known in literature that for tackling and solving a generic and complex combinatorial optimization problem, any evolutionary algorithm must incorporate local search methodologies that, even in refining the solutions and improving the fitness function, they are strongly based on the features and knowledge of the problem itself, and consequently they make the algorithm unsuitable and inapplicable to other problems. But in this way, studies will lose general validity. Therefore, to overcome this limitation and make our results as general as possible, we conducted our studies tackling the classic *One–Max* (or *One–Counting*) problem [15], [7].

One–Max is a well-known toy problem, used for understanding the dynamics and searching ability of a stochastic algorithm [14]. 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 [3]. It is worth emphasizing that a toy problem gives us a *failure bound*, because a

A. Di Stefano and A. Vitale are with the Department of Electric, Electronics and Computer Science, University of Catania.

V. Cutello and M. Pavone (*Member, IEEE*) are with the Department of Mathematics and Computer Science, University of Catania, V.le A. Doria 6, I-95125 Catania, Italy (email: {cutello, mpavone}@dmi.unict.it).

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 $\vec{x}$ of length ℓ:

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

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

The goal of this research work is basically to answer three main questions: (i) is the lifespan related to the number of offspring generated?; (ii) is the lifespan related to the population size?; and lastly, in case of negative answer to the previous two questions, (iii) how long must the lifespan of an offspring be to carry out a proper exploration?

The paper is structured as follow: in Sect. II we describe opt-IA, the developed immunological-inspired algorithm, and its features, with the main emphasis on the cloning, hypermutation, and aging operators; in Sect. III we describe and present the several age options (11 overall) for the assignment; in Sect. IV we present a large set of experiments conducted in order to determine primarily the efficiency ranking; and finally, Sect. V contains the concluding remarks.

II. The Immunological Algorithm

In this research work we have developed an immunological algorithm inspired by the clonal selection theory, which belongs to a special class of the immunological heuristics family called *Clonal Selection Algorithms* (CSA) [4], [9], [10], [11]. The main features of this kind of heuristics are the operators: (i) cloning, (ii) inversely proportional hypermutation and (iii) aging. The first operator generates a new population of B cells centered on the higher affinity values; the second one explores the neighborhood of each point in the search space, perturbing each solution via an inversely proportional law to its fitness function; and the last one eliminates old solutions from the current population so to introduce diversity and avoid, possibly, local minima during the evolutionary search process.

The developed algorithm takes into account the two main entities: antigen (Ag) and B Cell receptor. The first is the problem to tackle, while the second represents points in the search space of the problem to be tackled. For simplicity, hereafter, we call the algorithm as OPT-IA. At each time step t OPT-IA maintains a population of B cells $P^{(t)}$ of size d (i.e., d candidate solutions), which is initialized at the time step $t = 0$, by randomly generating solutions using uniform distribution in the corresponding domain (i.e. $\{0, 1\}$). Once the population is initialized, the next step is evaluate the fitness function for each B cell $\vec{x} \in P^{(t)}$ using the function *Evaluate_Fitness*($P^{(t)}$). A summary of OPT-IA is presented in the pseudocode shown in Algorithm 1.

The first immunological operator applied is the *cloning operator*, which simply copies dup times each solution (i.e. each B cell) producing hence an intermediate population $P^{(clo)}$ of size $d \times dup$, where dup is a user–defined parameter. To any clone, or copy, we assigne an age that determines its lifetime into the population: when a clone reaches the maximum age (τ_B) (user-defined parameter) the aging operator removes it from the population. The assignment of the age, together with the aging operator, has the purpose to reduce premature convergences, and keep high diversity into the population. It should be pointed out that the choice of which age to assign plays a central role on the performances of OPT-IA, and of any evolutionary algorithm in general, since the evolution and maturation of the solutions depend on it. What age value to assign to each clone is the focus of this research work (see section III). The cloning operator, coupled with the hypermutation operator, performs a local search around the cloned solutions. The introduction of blind mutation produces individuals with higher affinity (i.e. higher fitness function values), which will be then selected to form the improved mature progenies.

Algorithm 1 Immunological Algorithm (d, dup, ρ, τ_B)

```
t ← 0;
P^(t) ← Initialize_Population(d);
Evaluate_Fitness(P^(t));
repeat
  Increase_Age(P^(t));
  P^(clo) ← Cloning (P^(t), dup);
  P^(hyp) ← Hypermutation(P^(clo), ρ);
  Evaluate_Fitness(P^(hyp));
  (P_a^(t), P_a^(hyp)) ← Aging(P^(t), P^(hyp), τ_B);
  P^(t+1) ← (μ + λ)-Selection(P_a^(t), P_a^(hyp));
  t ← t + 1;
until (termination criterion is satisfied)
```

The *hypermutation operator* acts on each solution of population $P^{(clo)}$ performing M mutations, whose number is determined by an *inversely proportional law*: the higher is the fitness function value, the lower is the number of mutations performed on the B cell. It should be noted that the hypermutation operator works without using mutation probability. In particular, in OPT-IA, the number of mutations M to perform over $\vec{x}$ is determined by the following *potential mutation*:

$$\alpha = e^{-\rho \hat{f}(\vec{x})},$$

where α represents the *mutation rate*, and $\hat{f}(\vec{x})$ the fitness function value normalized in $[0, 1]$. Therefore, the number of mutations M is given by

$$M = \lfloor (\alpha \times \ell) + 1 \rfloor,$$

with ℓ the length of the B cell. Using this equation, at least one mutation is guaranteed on each B cell; and this happens exactly when the solution is very closer to the optimal one. It is worth emphasizing that during normalization of the fitness function value, in order not to use any *a priori* knowledge about problem, we use the best current fitness value decreased by a *user-defined threshold* θ, rather than the global optima (often not known). The hypermutation operator used is basically the classical *bit-flip mutation without redundancy*: in any $\vec{x}$ B cell, an element x_i is randomly

chosen without repetition, and its value is inverted (from 0 to 1, or from 1 to 0).

The last immunological operator to be performed is the *aging operator*, whose task is avoid premature convergences and getting trapped into local optima; and produce high diversity into the current population. This operator, simply, eliminates the old B cells from the populations $P^{(t)}$ and $P^{(hyp)}$: every B cell is allowed to remain in the current population for a fixed number of generations τ_B (user-defined parameter); as soon as a B cell is old $\tau_B + 1$ it is removed from the population of belonging independently from its fitness value, included the best solution found so far. The parameter τ_B, hence, indicates the maximum number of generations allowed to any B cell to remain into the population. There exists a variant of this operator that makes an exception on the removal of the best solution found so far: i.e. the best current solution is always kept in the population, even it is older than τ_B+1. This variant is called *elitist aging operator*.

After the three immunological operators have done their work, a new population $P^{(t+1)}$ is created for the next generation by using $(\mu+\lambda)$-*Selection operator*, which selects the best d survivors to the aging step from the populations $P_a^{(t)}$ and $P_a^{(hyp)}$. Such an operator, with $\mu = d$ and $\lambda = (d \times dup)$, reduces the offspring B cell population of size $\lambda \geq \mu$ – created by cloning and hypermutation operators – to a new parent population of size $\mu = d$. The selection operator identifies the d best elements from the offspring set and the old parent B cells, thus guaranteeing monotonicity in the evolution dynamics. Nevertheless, due to the aging operator, it could happen that only $d_1 < d$ B cells survived; in this case, the selection operator randomly generates new $d - d_1$ B cells.

Finally, the algorithm terminates its execution when the termination criterion is satisfied. In this research work a maximum number of the fitness function evaluations T_{max} has been considered for all experiments performed.

III. Age Assignment Lifespan

As previously highlighted, the age assignment to each clone is crucial for the performances of the algorithm, as proven in [12] (see table 16, page 29), where using different age assignments criteria the algorithm shows different performances. Thus, we have conducted an experimental study in order to understand what is the best age assignment, in term of performance, convergence and success.

In this research work, several age options have been taken into account, and are reported in table I. Here we report a short description of the age considered, and types and symbols used for showing the results obtained (section IV).

So, overall we have studied 11 different types of age assignment:

1) age 0 (zero) for each clone;
2) random age chosen in the range $[0, \tau_B]$;
3) random age chosen in the range $[0, \frac{2}{3}\,\tau_B]$. In this way it is guaranteed to each B cell to evolve at least for a fixed number of generations (in the worst case $\frac{1}{3}\,\tau_B$);
4) random age chosen between 0 (zero) and age of parent that for simplicity we call *"inherited"*. In this way each offspring has the same age of the parent in the worst case;
5) random age chosen in the range $[0, \frac{2}{3}\,inherited]$. In this way for each offspring is guaranteed a lower age than the parent;
6) to each clone is assigned the same age of the parent (inherited). However, if after the M mutations performed on one clone, its fitness value improves, then its age is updated with:
   (a) zero;
   (b) randomly chosen in the range $[0, \tau_B]$;
   (c) randomly chosen in the range $[0, \frac{2}{3}\,\tau_B]$;
   (d) randomly chosen in the range $[0, inherited]$;
   (e) randomly chosen in the range $[0, \frac{2}{3}\,inherited]$;
7) same age of parent less one $(inherited - 1)$.

TABLE I
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

It is worth to note that assigning the same age of the parent $(inherited)$ to each clone leads to loss of clones and parents in the same generation, as showed in fig. 1 (an example performed with $\ell = 1000$) with the outcome to waste the gained learning, as well as the best result found up to that time step. In light of this, we have considered the option $(inherited - 1)$ because it guarantees at least one life generation more than the parent. It is also very interesting to observe, by inspecting this figure, that OPT-IA, though it looses all individuals (clones and parents) in the same generation, resulting in turn in the loss of all the information gained during the evolution, is able to gain again the same information and in the same time interval, starting from new individuals, randomly generated. This confirm us the robustness and efficiency of OPT-IA.

The goal of this study is to have an efficiency ranking

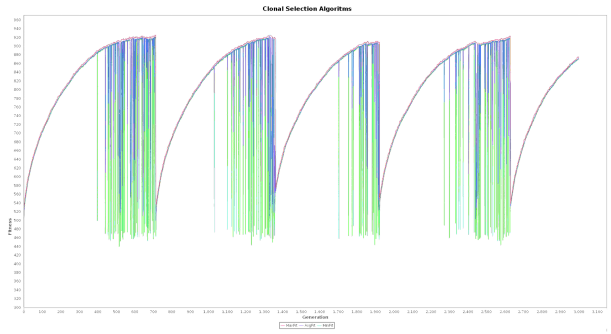

Fig. 1. Convergence dynamics of OPT-IA, using the same age of the parent for each clone.

between these options, rather than only to determine the best one, and to check as well if among the top $3-4$ positions the same options appear always, although in different order. To achieve our goals, we need first to understand if the age assignment depends on the number of offspring considered; or on the population size.

IV. Results

In this section we present the experimental results obtained by variations of OPT-IA with the aim to understand which are the top $3-4$ age assignments that show better overall performances. We have considered the classical One-Max toy problem for our experiments, and in order to make more complex the testbed we have considered a bit string of length $\ell = 10,000$ as problem dimension. However, as already highlighted, we first need to understand if the age assignment is closely related to the number of offspring, or to the population dimension. Thus, in order to answer these questions, and the two main other – (i) what is the best age assignment, and (ii) what is the efficiency ranking – we have performed several experiments, varying of the parameters as follows: $d = \{50, 100\}; dup = \{2, 5, 10\}$; and $\tau_B = \{5, 10, 15, 20, 50, 100, 200\}$. Furthermore, the two variants of OPT-IA, with and without elitism, have been tested and studied, with a total of 924 overall experiments. Each experiment has been performed on 100 independent runs, and fixing $T_{max} = 10^6$.

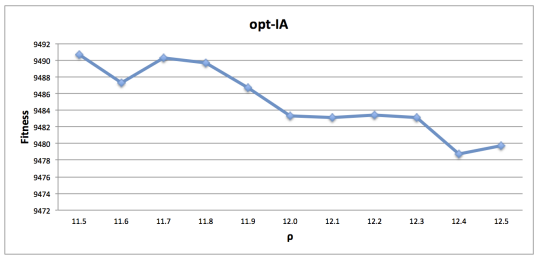

Fig. 2. The OPT-IA performances at varying of ρ parameter: a zoom.

The evaluation measures considered for our experiments are, in order: (a) *success rate* (SR), i.e. how many times OPT-IA finds the optimal solution in 100 runs; (b) average number of fitness evaluations to the optimal solution (AES); (c) best solution found on 100 independent runs (when $SR = 0$); (d) $mean$ of best solutions found on 100 independent runs; (e) σ standard deviation; and (f) mean of the fitness of the population, averaged on the overall generations, and 100 runs (avg_fit).

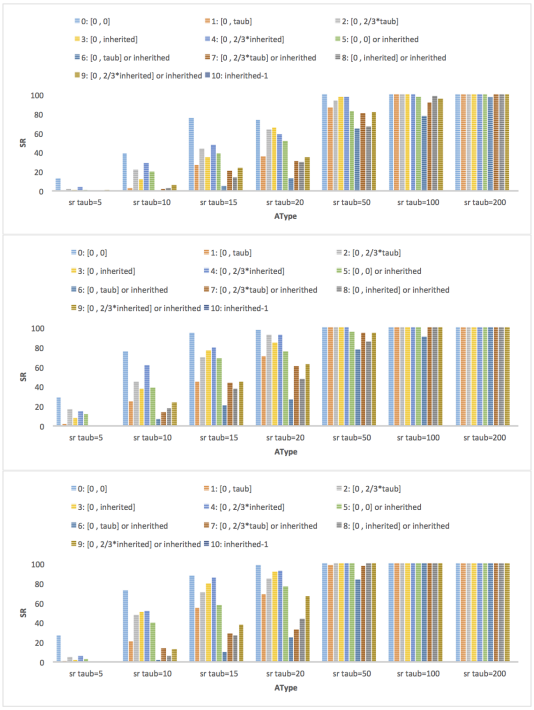

Fig. 3. SR versus $Aging\,Type$: results obtained by the elitist version of OPT-IA ($d = 100$, $dup = \{2, 5, 10\}$ and $\tau_B = \{5, 10, 15, 20, 50, 100, 200\}$).

Before starting the experimental study we need to tune the only parameter that does not affect the age assignment, i.e. ρ, which determines the mutation rate to perform. This parameter indeed is closely related only to the problem dimension. Thus, OPT-IA was tested by varying the parameter ρ in the range of real-valued $[6, 15]$. From these experiments OPT-IA showed the better performances for $\rho \in \{11.5, \ldots, 12.5\}$, as showed in figure 2, and the best one was obtained for $\rho = 11.5$. Hereafter, all experiments showed in this section have been performed using this setting for ρ.

Figs. 3 – 4 show the results obtained by opt-IA on the 11 options of age assignment, in term of SR by varying the τ_B parameter. In particular, fig. 3 shows the results obtained by using elitism, and fig. 4 the results without elitism. Analysing the results with the use of elitism, it is possible to see how the age assignment "$type\,0$" (see table I) shows always the higher success rate with respect to all other options. This means that every offspring needed a good maturation time in order to well explore the search space. These good performances are more obvious for $\tau_B = 5$, where in all three experiments OPT-IA produces a $SR > 10$.

In the overall from fig. 3 we may deduce the following top 4 age assignments: "$type\,0$"; "$type\,4$"; "$type\,2$"; and "$type\,3$". It is worth to note that assigning the same age of the parent

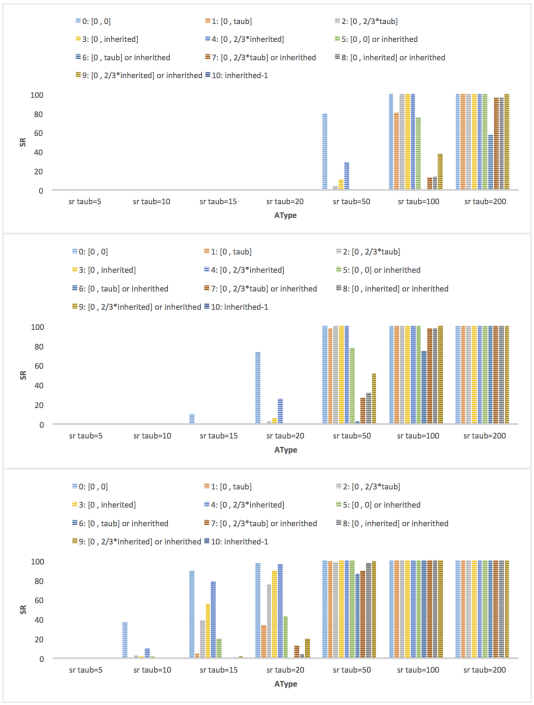

Fig. 4. SR versus $Aging\,Type$: results obtained by the version of OPT-IA without elitism ($d = 100$, $dup = \{2, 5, 10\}$ and $\tau_B = \{5, 10, 15, 20, 50, 100, 200\}$).

minus 1 ("$type10$") is not a good choice, since in this case OPT-IA is never able to find the optimal solution ($SR = 0$). What it is instead really interesting, by inspecting this figure, is that there exists a threshold for the maximum number of generations allowed (τ_B), above which OPT-IA shows an overall form of elitism (strengthening of the elitism), regardless not only to the age assignment, but also to all parameters, with the result of driving all B cell solutions toward the optimal one. The existence of this threshold is found in all experiments done although with different values. Of course, this feature is related to the type of the problem tackled. A better understanding of this dynamic will be the subject of future research.

Figure 4 shows the results obtained by OPT-IA without elitism. Is possible to see that the algorithm for small dup values is not able to find the optimal solution, except for high τ_B values. In these experiments, for $dup = 2$ is also possible to note how the threshold appears less effective than in the previous experiments in fig. 3, as well as for the other dup values. Also in these experiments the age assignment "$type0$" seems to be the best choice, followed by "$type4$;" "$type3$;" and "$type2$;" whilst "$type10$" also in these experiments proves to be a bad choice.

By inspecting these two first figures, it is possible already to assert that the age assignment seems to be independent from the number of offspring used.

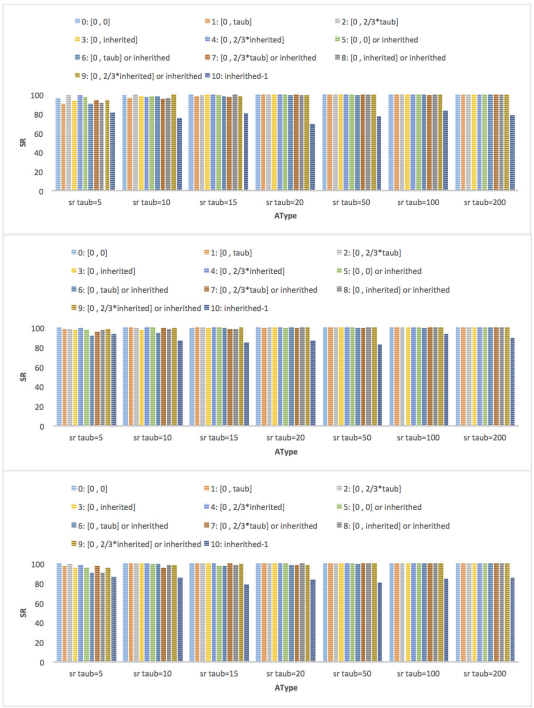

Fig. 5. SR versus $Aging\,Type$: results obtained by the elitist version of OPT-IA ($d = 50$, $dup = \{2, 5, 10\}$ and $\tau_B = \{5, 10, 15, 20, 50, 100, 200\}$).

Where, instead, the existence of the threshold appears to be very effective is in the experiments performed for $d = 50$ and using the elitism version of OPT-IA, as reported in figure 5. Using small population size and the elitist aging operator, the outcome is to make stronger the elitism, and this, although helps OPT-IA in reaching always the best solution for this problem independently by parameters and age assignments used, may not occur in other complex problems.

An almost opposite behavior occurs instead by running OPT-IA without elitism, as shown in fig. 6. Also in these experiments the top 4 age assignment options are "$type0$;" "$type4$;" "$type3$;" and "$type2$". All in all, by inspecting all 4 figures, it is possible to assert that the age assignment is not related to the number of offspring nor to the population size, and the top 4 age assignments seem to be, in the order, "$type0$;" "$type4$;" "$type3$;" and "$type2$". Moreover, all experiments clearly prove that the age assignment "$type10$" shows the worst performances, not reaching the optimal solution, and consequently, not allowing a proper maturation of the B cell.

To confirm these claims, we show some of the most relevant results in tables II, and III, for both versions of OPT-IA when varying the population size. In these tables we show the results for all 11 aging type using the evaluation measure described above. Table II shows the results obtained by the two versions of OPT-IA (elitism and no elitism) with the following parameters setting: $d = 100$, $dup = 5$ and $\tau_B = 20$. These results were performed on 100 independently runs. We highlight in bold face the best result. By inspecting

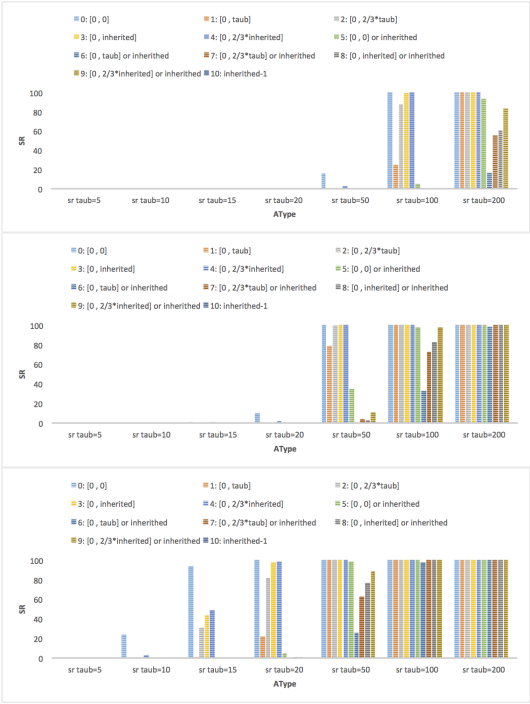

Fig. 6. SR versus $Aging\,Type$: results obtained by the version of OPT-IA without elitism ($d = 50$, $dup = \{2, 5, 10\}$ and $\tau_B = \{5, 10, 15, 20, 50, 100, 200\}$).

this table, it is possible to confirm the above statements, as well as to note that the two best options ("$type0$" and "$type4$") show similar performances in both versions of OPT-IA, proving that a proper setting of the age helps significantly in performing a right exploration, and exploitation (mainly in absence of elitism). It is also possible to note how the "$type10$" age reaches a best solution far from the optimal one, albeit it works on a population with high average fitness (avg_fit). This is due, as we expected, to a very low diversity produced by this kind of age assignment, that, on one hand, helps the algorithm to keep B cell with high affinity, but on the other hand it doesn't help the algorithm to jump out from the local optima.

Figure 7 shows the AES dynamic behavior, i.e. the average number of fitness evaluations to find the optimal solution ($SR \neq 0$) when varying the parameter τ_B (third column of table II). The top plot reports the performances of the elitist version of OPT-IA, whilst the bottom one shows the AES obtained without using elitism. In figure 8, instead, are reported the average fitness population (avg_fit) behavior for each age assignment options considered, for both elitist (top) and no elitist (bottom) versions.

In table III are reported the results obtained by OPT-IA without elitism fixing the parameters: $d = 50$, $dup = 10$ and $\tau_B = 15$. Also from this table is possible to verify the same ranking (approximately) of the above results. In figures 9 and 10 are reported respectively the AES and avg_fit dynamic behaviors when varying the τ_B parameter for all the 11 age options. Also in these plots, as well as in the previous ones,

TABLE II

OPT-IA ON ONE-MAX PROBLEM WITH $\ell = 10,000$. THE RESULTS HAVE BEEN OBTAINED BY SETTING: $d = 100, dup = 5, \tau_B = 20$, WITH AND WITHOUT ELITISM.

type	SR	AES	$best$	$mean$	σ	avg_fit
With Elitism						
0	**98**	$\mathbf{4.3 \times 10^6}$	**10000**	**9999.98**	**0.14**	**8964.70**
1	69	4.6×10^6	10000	9999.69	0.46	8957.78
2	85	4.4×10^6	10000	9999.85	0.36	8964.36
3	95	4.4×10^6	10000	9999.95	0.22	8964.25
4	97	4.4×10^6	10000	9999.97	0.17	8965.08
5	77	4.3×10^6	10000	9999.75	0.48	8485.66
6	25	4.7×10^6	10000	9998.79	0.96	8458.66
7	33	4.5×10^6	10000	9999.17	0.69	8468.74
8	48	4.5×10^6	10000	9999.34	0.71	8421.47
9	59	4.4×10^6	10000	9999.53	0.62	8460.44
10	0	//	9992	9983.61	3.71	8943.93
Without Elitism						
0	**97**	$\mathbf{4.3 \times 10^6}$	**10000**	**9999.97**	**0.17**	**8964.36**
1	34	4.6×10^6	10000	9999.29	0.55	8956.5
2	76	4.5×10^6	10000	9999.76	0.43	8964.26
3	90	4.4×10^6	10000	9999.9	0.3	8964.99
4	96	4.4×10^6	10000	9999.96	0.2	8964.35
5	43	4.2×10^6	10000	9999.32	0.68	8486.99
6	0	//	9999	9997.2	1.25	8475.98
7	13	4.2×10^6	10000	9998.56	1.02	8449.93
8	4	4.3×10^6	10000	9998.36	0.77	8443.18
9	20	4.2×10^6	10000	9998.94	0.71	8451.6
10	0	//	9944	9939.36	2.21	8932.74

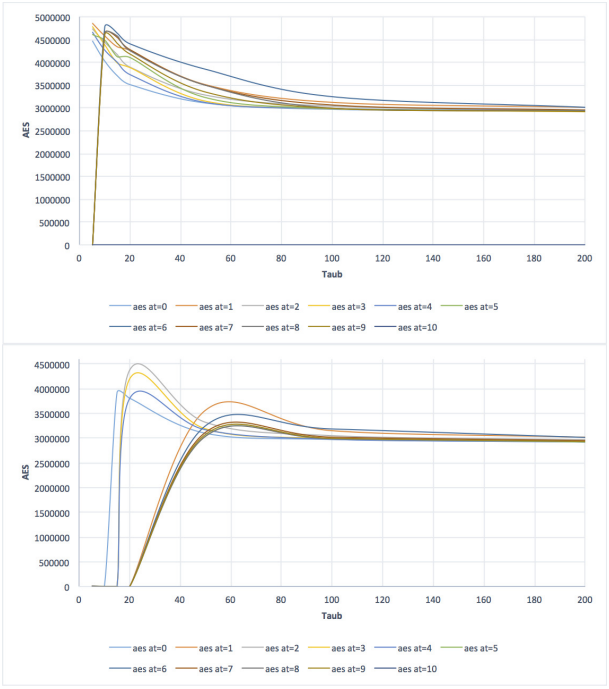

Fig. 7. AES versus τ_B: $d = 100$; $dup = 5$; elitist (top), and no elitist (bottom) versions of OPT-IA.

it is possible to see how the top 4 age assignment types need a lower number of fitness function evaluations to reach the optimal solution, obtained higher SR values.

Finally, what emerges also from all presented experiments is that the option "$type6$" is almost always the second to last in the ranking, and this is because if an offspring improves the fitness, it likely will have an age as closer as possible to

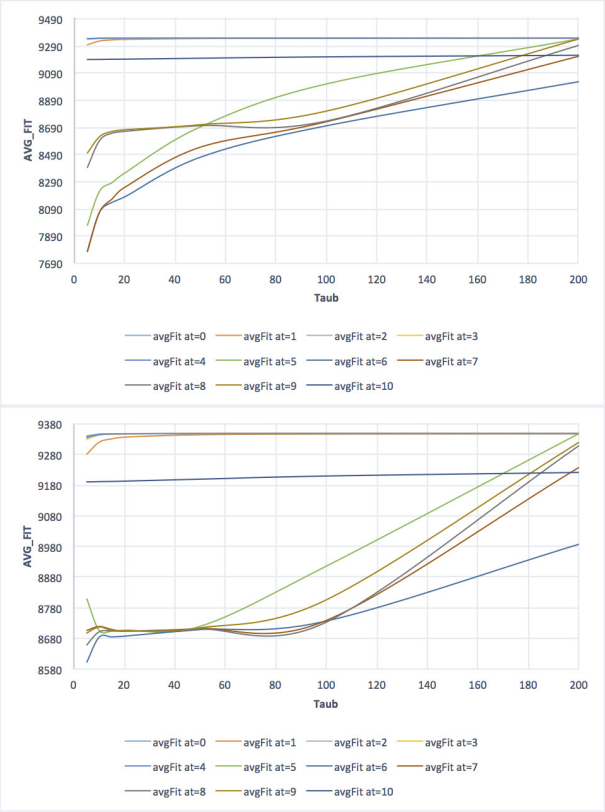

Fig. 8. Average fitness of the population (avg_fit) versus τ_B: $d = 100$; $dup = 5$; elitist (top), and no elitist (bottom) versions of OPT-IA.

TABLE III

OPT-IA ON ONE-MAX PROBLEM WITH $\ell = 10,000$. THE RESULTS HAS BEEN OBTAINED USING: $d = 50, dup = 10, \tau_B = 15$, AND NO ELITISM.

type	SR	AES	$best$	$mean$	σ	avg_fit
0	**94**	$\mathbf{3.1 \times 10^6}$	**10000**	**9999.94**	**0.24**	**9473.86**
1	1	4.7×10^6	10000	9998.47	0.56	9467.7
2	31	3.5×10^6	10000	9999.29	0.5	9473.4
3	44	3.4×10^6	10000	9999.44	0.5	9474.14
4	49	3.3×10^6	10000	9999.49	0.5	9473.61
5	0	0	9999	9997.27	1.08	8534.37
6	0	0	9995	9991.43	1.98	8513.31
7	0	0	9998	9994.51	1.46	8505.78
8	0	0	9998	9994.47	1.6	8505.55
9	0	0	9998	9995.93	1.32	8515.91
10	0	0	9937	9931.4	1.93	9417.13

τ_B, or anyhow older than its parent, and thus the occurred improvement is not exploited properly. This is likely because even the age "$type1$" does not show good performance as we would have expected.

V. CONCLUSION AND FUTURE WORK

In this research work, we presented an experimental study on what age should be assigned to a B cell in order to properly explore the search space. An immunological-inspired heuristic has been developed, and called OPT-IA, which is based on three main operators: cloning, hypermutation and aging operators. The age assignment, i.e. how many more cycles an offspring must remain into the population, plays a central role on the performance of any evolutionary

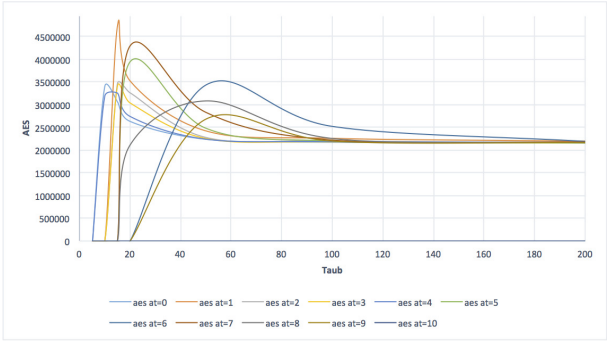

Fig. 9. AES versus τ_B: $d = 50$; $dup = 5$; and version of OPT-IA without elitism.

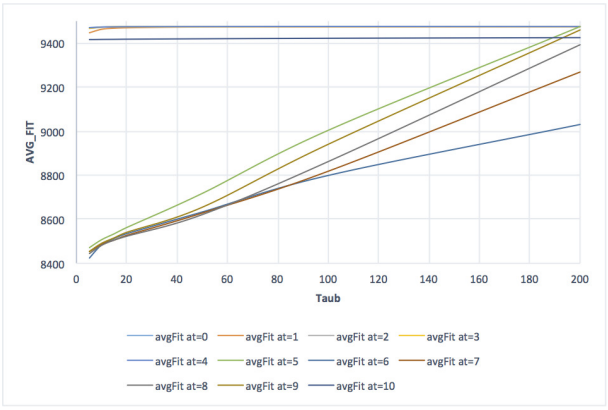

Fig. 10. Average fitness of the population (avg_fit) versus τ_B: $d = 50$; $dup = 5$; and version of OPT-IA without elitism.

algorithm, since it is crucial for having a correct balance between exploration of the solutions in the search space, and the exploitation of the knowledge gained during the search process. For this research work, we have considered the classical *One-Max* toy problem in order to design OPT-IA not tailored to a specific problem, keeping then the algorithm unaware on the problem knowledge. Toy problems are generally used to understand the dynamics and features of a generic stochastic algorithm. On the other hand, it is known that, an evolutionary algorithm works well on a complex combinatorial optimization problem if it incorporates knowledge of the problem itself, with the outcome to limit the application flexibility of the algorithm. Eleven (11) different types of age assignment are presented, and studied (when varying of the OPT-IA parameters), with the goal to determine the best one, and mainly their efficiency ranking. In order to achieve our goal we needed first to understand if the age assignment was strictly related to the number of offspring considered, or to the population size. Many experiments have been performed for an overall of 924 tests, and from the analysis of all experimental results, we may assert that the age assignment is not affected by neither the number of the copies (B cells or offspring), nor the population size used. Moreover, it emerges that assigning age zero (0) to each clone seems to be the best choice to do, whilst considering the same age of parent minus one,

doesn't help the performances of the algorithm, taking the last place of the efficiency rankings. Referring to the ranking, the top 4 age assignment types are respectively: (1) "$type0$;" (2) "$type4$;" (3) "$type3$;" and (4) "$type2$" (see table I). It is also possible to claim that in the last two positions we have the options "$type6$" and "$type10$" respectively, whose worst performances are caused by a lower diversity produced, and very less time given to the offspring to evolve and mature.

Finally, of course, we don't expect that the best obtained age assignment in this work is still valid in other problems or in another evolutionary algorithm, but rather, we are interesting in determining the top $3-4$ age assignments to take into account, not necessarily in the same order. This research work is a first step of our study, which is addressed, in the overall, in verifying if the top 4 age assignments are still valid on other problems - discrete and continuous domains - even if in different order. In a second step of our work (currently under study), we are testing OPT-IA using mathematical models as *"tunably rugged" fitness landscape*, in order to validate the generality of the outcomes obtained. Another interesting line of research we intend to pursue is to use our methodology to analyze other intelligent algorithms, such as Moth Search (MS) algorithm [17], EarthWorm optimization Algorithm (EWA) [18], Elephant Herding Optimization (EHO) [19], Monarch Butterfly Optimization (MBO) [20].

REFERENCES

[1] http://tracer.lcc.uma.es/problems/onemax/onemax.html

[2] P. Conca, G. Stracquadanio, O. Greco, V. Cutello, M. Pavone, G. Nicosia: "*Packing equal disks in a unit square: an immunological optimization approach*", 1st International Workshop on Artificial Immune Systems (AIS), IEEE Press, pp. 1–5, 2015.

[3] 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.

[4] V. Cutello, D. Lee, S. Leone, G. Nicosia, M. Pavone: "*Clonal Selection Algorithm with Dynamic Population Size for Bimodal Search Spaces*", 2nd International Conference on Natural Computation (ICNC), LNCS 4221, pp. 949–958, 2006.

[5] V. Cutello, D. Lee, G. Nicosia, M. Pavone, I. Prizzi: "*Aligning Multiple Protein Sequences by Hybrid Clonal Selection Algorithm with Insert-Remove-Gaps and BlockShuffling Operators*", 5th International Conference on Artificial Immune Systems (ICARIS), LNCS 4163, pp. 321-334, 2006.

[6] V. Cutello, G. Morelli, G Nicosia, M. Pavone, G. Scollo: "*On discrete models and immunological algorithms for protein structure prediction*", Natural Computing, vol. 10, no. 1, pp. 91-102, 2011.

[7] 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.

[8] V. Cutello, G. Nicosia, M. Pavone: "*An Immune Algorithm with Stochastic Aging and Kullback Entropy for the Chromatic Number Problem*", Journal of Combinatorial Optimization, vol. 14, no. 1, pp. 9–33, 2007.

[9] V. Cutello, G. Nicosia, M. Pavonem I. Prizzi: "*Protein Multiple Sequence Alignment by Hybrid Bio-Inspired Algorithms*", Nucleic Acids Research, vol. 39, no. 6, pp. 1980–1992, 2011.

[10] V. Cutello, G. Nicosia, M. Pavone, G. Stracquadanio: "*An Information Theoretic Approach for Clonal Selection Algorithms*", 9th International Conference on Artificial Immune Systems (ICARIS), LNCS 6209, pp. 144–157, 2010.

[11] V. Cutello, G. Nicosia, M. Pavone, J. Timmis: "*An Immune Algorithm for Protein Structure Prediction on Lattice Models*", IEEE Transaction on Evolutionary Computation, vol. 11, no. 1, pp. 101–117, 2007.

[12] M. Pavone, G. Narzisi, G. Nicosia: "*Clonal Selection - An Immunological Algorithm for Global Optimization over Continuous Spaces*", Journal of Global Optimization, vol. 53, no. 4, pp. 769–808, 2012.

[13] M. Poggiolini, A. Engelbrecht: "*Application of the feature-detection rule to the negative selection algorithm*", Expert Systems with Applications, vol. 40, no. 8, pp. 3001–3014, 2013.

[14] A. Prugel-Bennett, A. Rogers: "*Modelling Genetic Algorithm Dynamics*", Theoretical Aspects of Evolutionary Computing, pp. 59-85, 2001.

[15] J. D. Schaffer, L. J. Eshelman: "*On crossover as an evolutionary viable strategy*", 4th International Conference on Genetic Algorithms, pp. 61–68, 1991.

[16] S. Smith, J. Timmis: "*Immune network inspired evolutionary algorithm for the diagnosis of Parkinsons disease*", Biosystems, vol. 94, no. (1–2), pp. 34–46, 2008.

[17] G.G. Wang: "*Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems*", Memetic Computing, pp. 1–14, 2016.

[18] G.G. Wang, S. Deb, L.D.S. Coelho: "*Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems*", International Journal of Bio-Inspired Computation, 2015.

[19] G.G. Wang, S. Deb, L.D.S. Coelho: "*Elephant Herding Optimization*", 3rd International Symposium on Computational and Business Intelligence (ISCBI), pp. 1–5, 2015.

[20] G.G. Wang, S. Deb, Z. Cui: "*Monarch butterfly optimization*", Neural Computing and Application, pp. 1–20, 2015.