Clonal Selection Algorithm with Dynamic Population Size for Bimodal Search Spaces

V. Cutello[1], D. Lee[2], S. Leone[1], G. Nicosia[1], and M. Pavone[1,2]

[1] Department of Mathematics and Computer Science
University of Catania
Viale A. Doria 6, 95125 Catania, Italy
{vctl, nicosia, mpavone}@dmi.unict.it
[2] *IBM-KAIST* Bio-Computing Research Center
Department of BioSystems, KAIST
373-1, Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea
{dhlee, mario}@biosoft.kaist.ac.kr

Abstract. In this article an Immune Algorithm (IA) with dynamic population size is presented. Unlike previous IAs and Evolutionary Algorithms (EAs), in which the population dimension is constant during the evolutionary process, the population size is computed adaptively according to a cloning threshold. This not only enhances convergence speed but also gives more chance to escape from local minima. Extensive simulations are performed on trap functions and their performances are compared both quantitatively and statistically with other immune and evolutionary optmization methods.

Keywords: Immune algorithm, dynamic population size, aging operator, trap functions.

1 The Clonal Selection Algorithm

Size of the population is one of the most important aspects in Evolutionary Algorithms (EAs). To choose the population size too small, can drive EAs towards a fast convergence; whilst if it is too large, EAs need of wide computational resources. Then setting a correct population size can be a critical task in many applications. Several researchers have been interesting to investigate the setting of the optimal population size, for EAs. In this research paper is presented a modified version of the CSA proposed in [1] characterized by dynamic population size, and it is called DYN-IMMALG (*dynamic immune algorithm*). To evaluate the search ability of DYN-IMMALG, and study its evolutionary behavior was used a well-known test-bed: the trap functions, which are complex toy problems often used in evolutionary computation to assess the search ability of a given EA.

In the designed IA, the initial population of candidate solutions is randomly generated using uniform distribution in the relative domains; the algorithm chooses randomly the values 0 or 1 in a binary representation. DYN-IMMALG creates the initial population with a fixed number of B cells receptor, by parameter d. At each time step $t > 0$, DYN-IMMALG presents a population $P^{(t)}$ of dynamic size, like occurs in nature: this is the main feature of the designed immune algorithm.

Static cloning operator clones each B cell *dup* times producing an intermediate population $P^{(clo)}$, assigning each clone the same *age* of its parent. Parameter *Age* of B cells, determines their life span into the population: when a B cell will have maximum age allowed, then it will die, i.e. will be eliminated from the population. After the hypermutation phase, a cloned B cell which undergoes a successfully mutation, i.e. improving the fitness value, called *constructive mutation*, will be considered to have age equal to 0. Such a scheme, as proposed in [10, 9, 4], intends to give an equal opportunity to each new B cell to effectively explore the landscape. Static cloning operator represents a good strategy working on constant population size, as showed in [5, 1], rather than on variable size. In this last case is more easy to have an exponential growth of the population size just in the first generations ($|P^{(t)}| > 10000$). This feature is called *explosion of the population* (or *Malthusian Catastrophe*). To avoid the explosion phenomenon, a new cloning operator, *dynamic clonal operator*, was designeded, which is based on a cloning threshold θ that determines the percentage of clones created by each B cell: only $\theta \times |P^{(t)}|$ best B cells in the current population will be clones. Nevertheless, using this cloning operator is possible to have *the extinction of the current population.* With respect to the explosion phase, choosing a correct balance between θ, *dup* and age's parameter were obtained better performances by DYN-IMMALG.

The hypermutation operator acts on the B cell receptor of $P^{(clo)}$. The proposed DYN-IMMALG uses *inversely proportional hypermutation* operator, which try to mutate each B cell receptor M times without the explicit usage of a mutation probability. The main feature of the inversely proportional hypermutation operator is that the number of mutations is inversely proportional to the fitness value: as the fitness function value of the current B cell increases, the number of mutations performed by DYN-IMMALG decreases. This operator performs at most $M_i(f(\boldsymbol{x})) = (c \times \ell)\frac{E^*}{f(\boldsymbol{x})}$ mutations, where E^* is the optimum of the given trap function and c is mutation rate. If $M_i(f(\boldsymbol{x})) \geq \ell$, then we have performed ℓ mutations. Moreover, DYN-IMMALG use the *stop at the First Constructive Mutation* (*FCM*) strategy [9]: if a constructive mutation occurs, the mutation procedure will move on to the next B cell. We adopted such a mechanism to slow down (premature) convergence, exploring more accurately the search space.

The aging operator, used by the algorithm, eliminates old B cells in the populations $P^{(t)}$, and $P^{(hyp)}$, maintaining high diversity in the current population, in order to avoid premature convergence. The maximum number of generations the B cells are allowed to remain in the population is determined by the τ_B parameter: when a B cell is $\tau_B + 1$ old it is erased from the current population, independently what is its fitness value.

After the application of the immune operators the surviving B cells from the populations $P_a^{(t)}$ and $P_a^{(hyp)}$, will constitute the population of the next genera-

Table 1. Pseudo-code of DYN-IMMALG

```
DYN-IMMALG($d, dup, \theta, \tau_B, c$)
$t := 0$;
$P^{(t)}$ := Initial_Population($d$);
Evaluate($P^{(t)}$);
while ( ¬ Termination_Condition() ) do
    Increase_Age($P^{(t)}$);
    $P^{(clo)}$ := Dynamic_Cloning ($P^{(t)}, dup, \theta$);
    $P^{(hyp)}$ := Hypermutation ($P^{(clo)}, c$);
    Evaluate($P^{(hyp)}$);
    ($P_a^{(t)}$, $P_a^{(hyp)}$):= Aging($P^{(t)}, P^{(hyp)}, \tau_B$);
    $P^{(t+1)}$ := ($P_a^{(t)} \cup P_a^{(hyp)}$);
    $t := t + 1$;
end_while
```

tion. Tackling the trap functions, to maintain high diversity into the population, no redundancy is allowed: each B cell receptor is unique.

The function *Evaluate($P^{(*)}$)* computes the fitness function value of each B cell $\boldsymbol{x} \in P^{(*)}$. Moreover, function *Termination_Condition()* is a boolean function, which returns true if a maximum number of fitness function evaluations (T_{max}) is reached, or the optima solution is found.

In table 1 is showed the pseudo-code of DYN-IMMALG.

Trap Functions. In this research work are used the traps functions, complex toy problem, to assess if DYN-IMMALG is able to reach an optimal solution starting from a randomly initialized population. Our study is restricted on *unitation functions*, which depend entirely upon the number of ones in a bit string but not on their position:

$$f(\boldsymbol{x}) = \hat{f}(u(\boldsymbol{x})) = \hat{f}\left(\sum_{k=1}^{\ell} x_k\right)$$

where ℓ is the length of the bitstring, such that $\boldsymbol{x} \in \mathbb{B}^{\ell}$, and $\hat{f} : \mathbb{B}^{\ell} \rightarrow \mathbb{R}$ is the *fitness function* that maps the bit string to a real number. To evaluate, mainly the search ability and, then goodness of the results obtained by DYN-IMMALG, were used two different trap functions: basic trap function and complex trap function [2]. The basic (or simple) trap function is characterized by a global optimum, obtained when the input is a string of all 0's, and a local optimum, obtained when the input is a string of ones. The formal definition is:

$$\hat{f}(u) = \begin{cases} \frac{a}{z}(z-u), & \text{if } u \leq z \\ \frac{b}{\ell-z}(u-z), & \text{otherwise.} \end{cases} \tag{1}$$

where a, b and z are parameters (see [2]). The complex trap function is more difficult to investigate, having two different slopes that lead to the local optimum.

It also avoids that bit-flipping operators, which change/reverse the values of all the bits in a string, may find the global optimum easily. This may happen in the simple trap presented above, where the two optima are bit-wise complements of each other. The formal definition is:

$$\hat{f}(u) = \begin{cases} \frac{a}{z_1}(z_1 - u), & if \quad u \leq z_1 \\ \frac{b}{\ell - z_1}(u - z_1), & if \quad z_1 < u \leq z_2 \\ \frac{b(z_2 - z_1)}{\ell - z_1}\Big(1 - \frac{1}{\ell - z_2}(u - z_2)\Big) & otherwise. \end{cases} \tag{2}$$

The values of the parameters chosen for both functions are the same as are used in [2]. This choice simplifies a comparison with works published previously. Using these parameters, are obtained several different functions, for each kind of trap function. Therefore, was used the following syntax: $S(type)$ and $C(type)$, where S and C mean respectively simple and complex trap function and *type* varying with respect the parameter values used (see table 2).

2 The Algorithm Dynamics

The common population explosion phenomenon is showed in left plot of figure 1, where is possible to see how population size increases very quickly in the first 8 generations. In this plot, one can see how the constructive mutations number is higher than destructive ones, i.e. when a mutation worsen the fitness value. Therefore, the major of the hypermutated cells not will die (each constructive mutation will be assign age 0), staying into the current population and increasing its size. For this reason, DYN-IMMALG reaches the maximum number of fitness function evaluations allowed (T_{max}), in the first 8 generations on each independently run.

Using the threshold ($\theta = 50\%$), instead, the destructive mutations increase more than the constructive, as showed in the right plot of figure 1. Of course, the population increase proportionally to an higher mean life value, like a *"natural"* selection process.

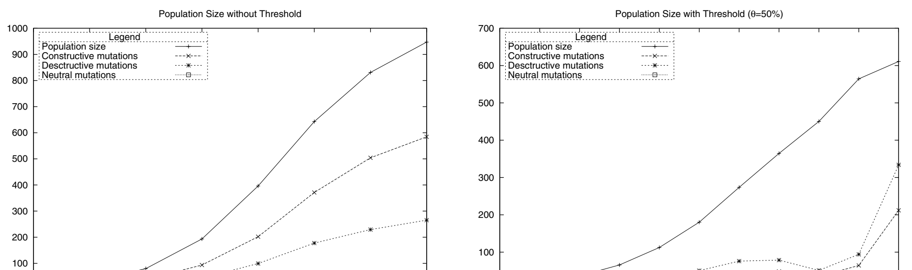

Fig. 1. Explosion of the population (left plot) without cloning threshold, $c = 0.1$, and $\tau_B = 10$. No explosion of the population (right plot) with cloning threshold $\theta = 50\%$, $c = 0.1$, and $\tau_B = 10$.

A set of experiments is performed on the simple and complex trap functions in order to analyse the population dynamics and performances of dynamic cloning with the threshold. All results from the set of experiments presented in this article are obtained with a population size of $d = 10$ and averaged over 1000 independent runs.

Figure 2 reports four snapshoots of Success Rate (SR) obtained by DYN-IMMALG for $\tau_b = (1, 2, 10, 50)$, tackling the simple trap function $S(II)$. From an overall view of such figure, one can see how the best performances, in term of SR, move towards low threshold values with the increase of τ_B parameter (from $SR = 2\%$ in (a) plot to $SR = 96\%$ in (d) plot). In fact, just (a) plot shows worst performances of the IA. Worst performances showed in (a) and (b) plots are obtained because for low values of τ_B : increasing the τ_B parameter value increases the success rate for low value of cloning threshold θ and high mutation rates c.

As written above, using a not correct setting of the parameters, is more easy to obtain either the explosion of the population or its extinction. However, were obtained several *special cases*, where DYN-IMMALG obtained together either success (optima solution), extinction, and failures (i.e. not able to find optima solution), or success and extinction, on 1000 independent runs. These special cases are showed in the figures 3, 4 and 5, using static and dynamic cloning operator, respectively.

Figure 3 shows the interesting case where on 1000 independent runs DYN-IMMALG was able to converge to optima solution, in some runs, and in others no, either because it reached T_{max}, or occurred the population's extinction. This feature is reported for static cloning (left plot) and dynamic cloning with the threshold θ (right plot). Left plot shows a peak in correspondence of the 8*th* generation, where the population size reach the maximum number of B cells (about $|P^{(t)}| = 1000$): around of this point is crucial for the performance of the proposed algorithm, because DYN-IMMALG either take the correct path towards global optima solution (success), or its opposite path towards local optima (failures), reaching, in this case, T_{max} fitness function evaluations (stop criterion). If DYN-IMMALG overtakes this around, then population size decreases, until to extinction (13*th* generation). Using the threshold (right plot), DYN-IMMALG presents a steady population size, until to an unexpected increase. During this increase, the algorithm converge towards global optima solution.

In figure 4 is showed the population size of the described algorithm versus the generations; both plots present the special case where DYN-IMMALG either obtains the global optima (success) or the population size decrease until $|P^{(t)}| = 0$ (extinction). Never DYN-IMMALG have consumed all T_{max} fitness function evaluations. Left plot shows the behavior obtained not using the threshold: in the first generations, in correspondence of the peak, DYN-IMMALG reaches the optimal solution, increasing the SR value. After the 6*th* generation, i.e. after the peak, when DYN-IMMALG not follows the right path, the population size have a quickly decrease until the extinction. Even in this plot, the around of the peak represents a bound for the performances of the proposed algorithm.

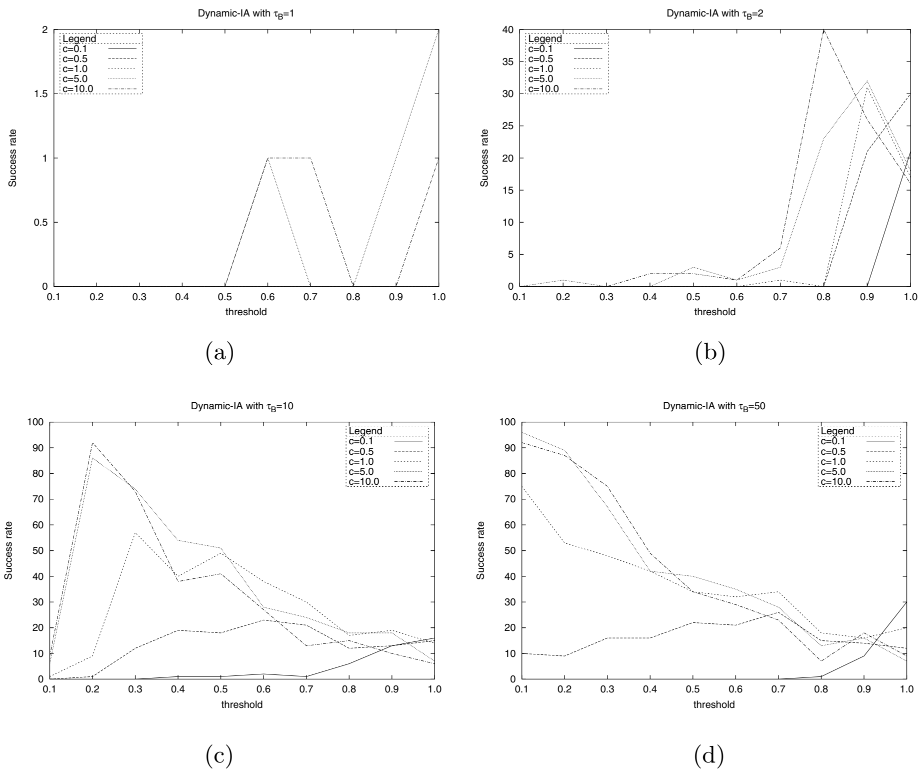

Fig. 2. Success Rate (SR) versus cloning threshold θ, and various mutation rates c for $\tau_b = 1, 2, 10, 50$. The plots concern the behaviour of DYN-IMMALG tackling the simple trap function $S(II)$.

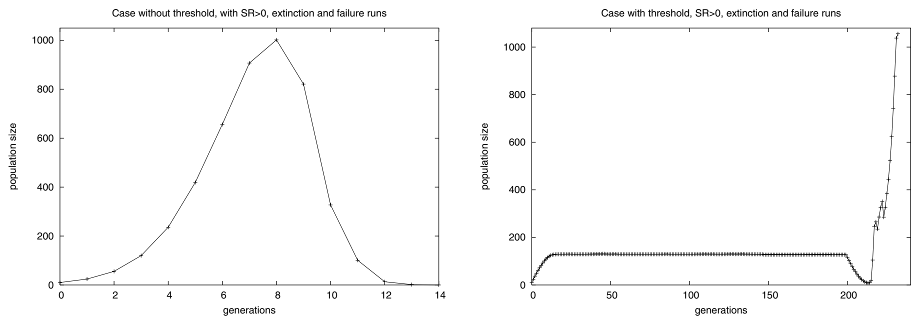

Fig. 3. Special case where DYN-IMMALG obtained together success (i.e. find optima solution), extinction and failures (i.e. is not able to reach the optima solution). These cases are showed both, without (left plot) and with (right plot) threshold θ.

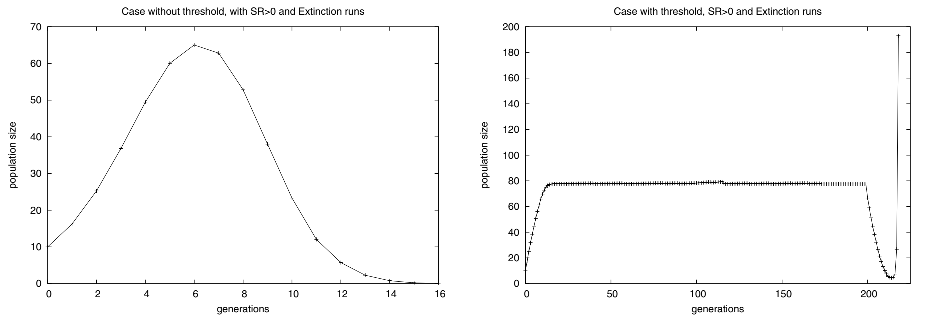

Fig. 4. Special case where DYN-IMMALG obtained both success (i.e. find optima solution) and population's extinction, on 1000 independent runs. This feature is showed using (right plot) or not (left plot) the threshold θ.

Instead, the right plot, shows the population size dynamics using the threshold θ. Like in figure 3 (right plot), the population size stays in a steady state, until the generation number is less than τ_B (in this experiment τ_B was fixed to 200). After $(\tau_B + 1)$ generations, the plot shows two different steps: in the first step, a decrease of the population size, which corresponds to extinction process; in the second step, a quickly increase, where the algorithm reaches the optimal solution.

Figure 5 shows the extinction process, which was obtained in all 1000 independent runs, with and without threshold. In particular, right plot shows a steady state of the population size, for the first τ_B generations ($\tau_B = 200$). Once the generations number is equal to τ_B, the population decreases until the extinction. This extinction phenomenon is obtained to low values of $dup = 1$ and $\theta = 1\%$ ($\theta = 0.01$). It is important to highlight, how using the threshold θ helps DYN-IMMALG to converge towards the right solution in more generations, handling better the population size, and to increase the SR value.

Analyzing deeply the obtained results, using dynamic clonal operator with threshold θ emerge an interesting feature, that is for each kind of trap functions

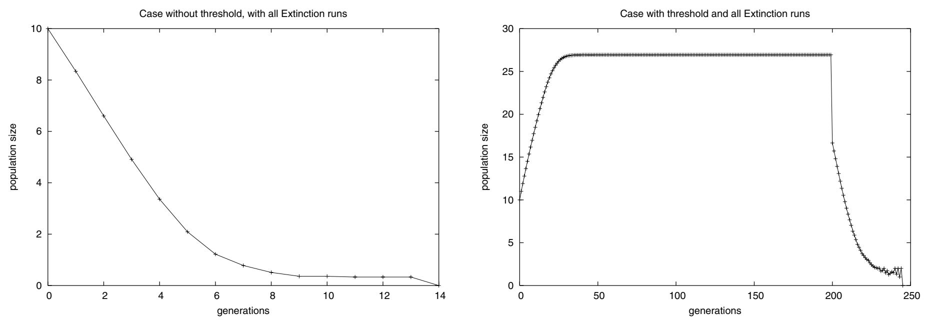

Fig. 5. Extinction. Special case where DYN-IMMALG obtained only extinction on all 1000 independent runs. This case is showed both, without (left plot) and with (right plot) threshold θ.

(simple and complex) the best performances of DYN-IMMALG occur in the first value of θ from which the extinction is not obtained. This feature can be better examined in figure 6. In such figure is showed the behavior of success rate and number of extinctions, obtained on 1000 runs. From both plots (fig. 6) is possible to see, as decrease of the extinction's phenomenon corresponds to increase the success rate: in particular, the first point where the extinction is null represents higher value of SR, i.e. of the best performances of DYN-IMMALG. Moreover, if the algorithm is able to obtain more $SR = 100\%$, with several theta values, then the point where the extinction is null, corresponds to $SR = 100\%$ with lowest *Average number of fitness function Evaluations to Solution* (AES).

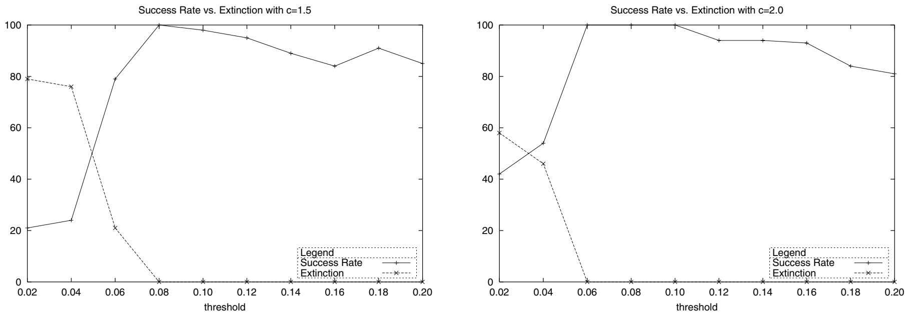

Fig. 6. Success Rate versus extinction, with $c = \{1.5, 2\}$. The plots concern the behaviour of DYN-IMMALG tackling the complex trap function $C(II)$.

3 Experimental Results

This section reports the comparisons between the proposed algorithm and others well-known immune algorithms. Were used two well-known clonal selection algorithms OPT-IA [5] and CLONALG [7]. For each instance, the experimental study was conducted using two versions of CLONALG, CLONALG_1 and CLONALG_2, which used different selection scheme [7, 5]: in CLONALG_1, at generation t, each B cell receptor will be substituted by the best individual of its set of mutated clones, whilst CLONALG_2, the population at the next generation $t + 1$ will be formed by the n best B cells' of the mutated clones at time step t.

The experimental results show the effectiveness of the designed immune algorithm. DYN-IMMALG finds the global optimum of all the examined simple and complex trap functions. The T_{max} limit, in terms of number of evaluations before the computation must terminate, is the same used often in literature [5]. Nevertheless the number of evaluations performed by the algorithm is much lower than T_{max}. The results are reported in terms of *Success Rate* (SR) and *Average number of Evaluations to Solutions* (AES). The table 2 shows the best results obtained by DYN-IMMALG on each *trap function* averaged on 1000.

In the simple trap functions, although, DYN-IMMALG and OPT-IA are competitive in terms of SR, the proposed algorithm is more able to find the optima

Table 2. DYN-IMMALG versus OPT-IA, and the two versions of CLONALG

Trap	*SR*	*AES*	*SR*	*AES*	*SR*	*AES*	*SR*	*AES*	*SR*	*AES*	*SR*	*AES*
	DYN-IMMALG		OPT-IA		CLONALG$_1$				CLONALG$_2$			
					$\left(\frac{1}{\rho}\right)$ $e^{(-f)}$		$e^{(-\rho*f)}$		$\left(\frac{1}{\rho}\right)$ $e^{(-f)}$		$e^{(-\rho*f)}$	
S(I)	100	**70**	100	477.04	100	1100.4	100	479.7	100	725.3	100	539.2
S(II)	100	**2291.57**	100	35312.29	100	27939.2	100	174563.4	30	173679.8	31	172191.2
S(III)	100	**4871.04**	100	20045.81	0	-	0	-	0	-	0	-
S(IV)	100	**15089.97**	100	42089	0	-	0	-	0	-	0	-
S(V)	100	**29507.47**	100	80789.94	0	-	0	-	0	-	0	-
C(I)	100	**63.2**	100	388.42	100	272.5	100	251.3	100	254.0	100	218.4
C(II)	100	**894.25**	100	29271.68	100	17526.3	10	191852.7	29	173992.6	24	172434.2
C(III)	**100**	18270.55	24	149006.5	0	-	0	-	0	-	0	-
C(IV)	**72.8**	828.51	2	154925	0	-	0	-	0	-	0	-
C(V)	**65.2**	2021.76	0	-	0	-	0	-	0	-	0	-

solution with lower values of fitness function evaluations. In the complex trap functions, DYN-IMMALG takes the correct path more times than OPT-IA and CLONALG, in particular for $C(III)$, $C(IV)$ and $C(V)$ complex trap functions.

4 Conclusion

In this paper we have introduced a new immune algorithm using dynamic population size to face bimobal search spaces. Unlike most previous IAs and EAs, in which the population dimension is constant during the evolutionary process, the population size is computed adaptively according to a cloning threshold θ. The validity of the proposed algorithm was tested using a widely used test bed. Experimental results show that the IA with dynamic population size, DYN-IMMALG, overcomes premature convergence, escape many local minima and finally finds more suitable solutions than OPT-IA and CLONALG. Future work includes DYN-IMMALG application to parameter extraction of circuit design problems, numerical optimization of dynamic functions (consist of a number of peaks, changing over time in height, width and location), and study on the convergence of the proposed immune algorithm.

Acknowledgments

This work was supported by the National Research Laboratory Grant (2005-01450) from the Ministry of Science and Technology. D.L. and M.P. would like to thank CHUNG Moon Soul Center for BioInformation and BioElectronics for providing research and computing facilities.

References

1. Cutello V., Nicosia G., Pavone M., Narzisi G.; "*Real Coded Clonal Selection Algorithm for Unconstrained Global Numerical Optimization using a Hybrid Inversely Proportional Hypermutation Operator*," in 21st Annual ACM Symposium on Applied Computing (SAC), vol. 2, pp. 950–954 (2006).

2. Nijssen S., Bäck T.; "*An Analysis of the Behavior of Simplified Evolutionary Algorithms on Trap Functions*," IEEE Transaction on Evolutionary Computation, vol. 7, no. 1, pp. 11–22 (2003).
3. Cutello V., Nicosia G.; "*The Clonal Selection Principle for in silico and in vitro Computing*," in de Castro, L. N., and von Zuben, F. J. V., eds., Recent Developments in Biologically Inspired Computing, pp. 104–146, USA: Idea Group Publishing (2004).
4. Cutello V., Morelli G., Nicosia G., Pavone M.; "*Immune Algorithms with Aging Operators for the String Folding Problem and the Protein Folding Problem*," in 5th European Conference on Computation in Combinatorial Optimization (EvoCOP), pp. 80–90 (2005).
5. Cutello V., Narzisi G., Nicosia G., Pavone M.; "*Clonal selection algorithms: A comparative case study using effective mutation potentials*," in 4th International Conference on Artificial Immune Systems (ICARIS), pp. 13–28 (2005).
6. de Castro L. N., Timmis, J.; "*Artificial Immune Systems: A New Computational Intelligence Paradigm*," Berlin, Germany: Springer-Verlag (2002).
7. de Castro L. N., von Zuben F. J. V.; "*Learning and optimization using the clonal selection principle*," IEEE Transaction on Evolutionary Computation, vol. 6, no. 3, pp. 239–251 (2002).
8. Nicosia G., Cutello V., Bentley P., Timmis J.; "*Proceedings of the Third International Conference on Artificial Immune Systems*," Berlin, Germany: Springer-Verlag (2004).
9. Cutello V., Nicosia G., Pavone M.; "*Exploring the capability of immune algorithms: A characterization of hypermutation operators*," in 3rd International Conference on Artificial Immune Systems (ICARIS), pp. 263–276 (2004).
10. Cutello V., Nicosia G., Pavone, M.; "*An immune algorithm with hyper-macromutations for the 2d hydrophilic-hydrophobic model*," in Congress on Evolutionary Computing (CEC), pp. 1074–1080 (2004).
11. Nicosia G.; "*Immune Algorithms for Optimization and Protein Structure Prediction*," Ph.D. Dissertation, Department of Mathematics and Computer Science, University of Catania, Italy (2004).