HOW TO ESCAPE TRAPS USING CLONAL SELECTION ALGORITHMS

V. Cutello, G. Narzisi, G. Nicosia, M. Pavone, G. Sorace
Department of Mathematics and Computer Science - University of Catania
V. le A. Doria 6, 95125 Catania Italy

Keywords: Evolutionary Algorithms, Immune Algorithms, Clonal Selection Algorithms, Trap Functions.

Abstract: This paper presents an experimental study on clonal selection algorithms (CSAs) to optimize simple and complex trap functions. Several settings of the proposed immune algorithms were tested in order to effectively face such a hard computational problem. The key feature to solve the trap functions, hence escape traps, is the usage of the hypermacromutation operator couple with a traditional perturbation immune operator. The experimental results show that the CSAs we designed are very competitive with the best algorithms in literature.

1 CLONAL SELECTION THEORY

Artificial Immune Systems (AIS) are computational models, inspired by biological immune systems in a broader sense, that have often been shown to be effective for difficult combinatorial optimization (Cutello V., 2002), learning and solving problems (Timmis J., 2001) and many other areas appearing in various industrial, economical and academic domains (de Castro L. N., 2002b). In this paper, we use Clonal Selection Algorithms (CSAs) to solve two trap functions, i.e. for extracting the characteristic behavior when facing these computational problems. The trap functions are complex *toy problem*, that can help in understanding the efficiency of algorithms' search ability. Toy problems (e.g., ones-counting, Basin-with-a-barrier, Hurdle-problem) play a central role in understanding the dynamics of algorithms (Prugel-Bennett A., 2001). They allow algorithm designers to devise new tools for mathematical analysis and modelling. One can tackle toy problems to build-up a fruitful intuition about the algorithm workings. Moreover, toy problems can be used to show the main differences between different algorithms. In the present experimental research work, we will consider the main differences between clonal selection algorithms, and we will show which particular algorithms and implementation to use to solve effectively the trap functions, a paradigmatic example of toy problem.

Clonal selection algorithms are special kind of Immune Algorithms (de Castro L. N., 2002b; Cutello V., 2004) which use the clonal expansion and the affinity maturation as the main forces of the evolutionary process. The theory of clonal selection (Burnet, 1959), suggests that among all possible cells with different receptors circulating in the host organism, only those who are actually able to recognize the antigen will start to proliferate by duplication (cloning). Hence, when a B cell is activated by binding an antigen, it produces many clones, in a process called *clonal expansion*. The resulting cells can undergo somatic hypermutation, creating offspring B cells with mutated receptors. Antigens compete for recognition with these new B cells, their parents and with other clones. The higher the affinity of a B cell to available antigens, the more likely it will clone. This results in a Darwinian process of variation and selection, called *affinity maturation*. Two key features of the clonal selection theory need to be taken into account: the hypermutation mechanism and the clonal expansion. Hypermutation can be seen as a local search procedure that leads to a fast "maturation" during the learning phase. The clonal expansion phase triggers the growth of a new population of high-value B cells centered on a higher affinity value. We will describe, in what follows, the class of immune algorithms (IA) based on the theory of clonal selection. To this end, we will abstract a simplified model of the IS. We will consider only two entities: antigens

Cutello V., Narzisi G., Nicosia G., Pavone M. and Sorace G. (2004).
HOW TO ESCAPE TRAPS USING CLONAL SELECTION ALGORITHMS.
In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 322-326
DOI: 10.5220/0001142503220326

(Ag's) and B cells. The input is the Ag (i.e.,the problem to tackle, the function to optimize); the output is basically the candidate solutions, the B cells, that have solved/recognized the Ag. All IAs based on the clonal selection theory are population based. Each individual of the population is a candidate solution belonging to the combinatorial fitness landscape of a given computational problem. Using the cloning operator, an immune algorithm produces individuals with higher affinities (higher fitness function values), introducing blind perturbation (by means of a hypermutation operator) and selecting their improved mature progenies. We will describe two different examples of Clonal Selection Algorithms. We start with the algorithm CLONALG (de Castro L. N., 2002a), which uses fitness values for proportional cloning, inversely proportional hypermutation and a birth operator to introduce diversity in the current population along with a mutation rate to flip a bit of a B cell memory. Extended algorithms use also threshold values to clone the best cells in the present population. We will, then, describe an immune algorithm that uses a static cloning operator, hypermutation and hypermacromutation operators, without memory cells and an aging phase, a deterministic elimination process; we will refer to the algorithm using the acronym opt-IA.

CLONALG. CLONALG (de Castro L. N., 2002a) is characterized by two populations: a population of antigens Ag and a population of antibodies Ab (denoted with $P^{(t)}$). The individual antibody, Ab, and antigen, Ag, are represented by string attributes $m = m_L, \ldots, m_1$, that is, a point in an $L-$dimensional real-valued shape space $S, m \in S^L \subseteq \Re^L$. The Ab population is the set of current candidate solutions, and the Ag is the environment to be recognized. After a random initialization of the first population $P^{(0)}$, the algorithm loops for a predefined maximum number of generations (N_{gen}). In the first step, it determines the fitness function values of all Abs in relation to the Ag. Next, it selects n Abs that will be cloned independently and proportionally to their antigenic affinities, generating the clone population P^{clo}. Hence, the higher the affinity-fitness, the higher the number of clones generated for each of the n Abs with respect to the following function: $N_c = \sum_{i=1\ldots n} \lfloor (\beta * n)/i \rfloor$ where β is a multiplying factor to be experimentally determined. Each term of the sum corresponds to the clone size of each Ab. The hypermutation operator performs an affinity maturation process inversely proportional to the fitness values generating the matured clone population P^{hyp}. After computing the antigenic affinity (i.e., the fitness function) of the population P^{hyp}, CLONALG creates randomly d new antibodies that will replace the d lowest fit Abs in the current population (for the pseudo-code of CLONALG see (de Castro L. N., 2002a)).

opt-IA. The opt-IA algorithm uses only two entities: antigens (Ag) and B cells like CLONALG. At each time step t, we have a population $P^{(t)}$ of size d. The initial population of candidate solutions, time $t = 0$, is generated randomly. The function *Evaluate(P)* computes the affinity (fitness) function value of each B cell $\vec{x} \in P$. The designed IA , like all immune algorithms based on the clonal selection principle, is characterized by clonal expansion, the cloning of B cells with higher antigenic affinity. The implemented IA uses three immune operators, cloning, hypermutation and aging. The cloning operator, simply, clones each B cell dup times producing an intermediate population P^{clo} of size $d \times dup$. The hypermutation operator acts on the the B cell receptor of P^{clo}. The number of mutations M is determined by a *mutation potential*. It is possible define various mutation potential. We tested our IA using static, and inversely proportional hypermutation operators, hypermacromutation operator, and combination of hypermutation operators and hypermacromutation. The two hypermutation operators and the Hypermacromutation perturbs the receptors using different mutation potentials, depending upon a parameter c In particular, it is worthwhile to note here, that all the implemented operators try to mutate each B cell receptor M times without using probability mutation. The mutation potentials used in this research work are the following: *Static Hypermutation (H1)*: the number of mutations is independent from the fitness function f, so each B cell receptor at each time step will undergo at most $M_s(\vec{x}) = c$ mutations. *Inversely Proportional Hypermutation (H2)*: the number of mutations is inversely proportional to the fitness value, that is it decrease as the affinity function of the current B cell increases. So at each time step t, the operator will perform at most $M_i(f(\vec{x})) = ((1 - \frac{E^*}{f(\vec{x})}) \times (c \times \ell)) + (c \times \ell))$ mutations. In this case, $M_i(f(\vec{x}))$ has the shape of an hyperbola branch. *Hypermacromutation (M)*: the number of mutations is independent from the fitness function f and the parameter c. In this case, we choose at random two integers, i and j such that $(i + 1) \leq j \leq \ell$ the operator mutates at most $M_m(\vec{x}) = j - i + 1$ directions, in the range $[i, j]$. The aging operator eliminates old B cells, in the populations $P^{(t)}$, $P^{(hyp)}$ and/or $P^{(macro)}$, to avoid premature convergence. To increase the population diversity, new B cells are added by the *Elitist_Merge function*. The parameter τ_B sets the maximum number of generations allowed to B cells to remain in the population. When a B cell is $\tau_B + 1$ old it is erased

by the current population, no matter what its fitness value is. We call this strategy, *static pure aging*. During the cloning expansion, a cloned B cell takes the age of its parent. After the hypermutation phase, a cloned B cell which successfully mutates, that is the new receptor will have a better fitness value, will be considered to have age equal to 0. Such a scheme intends to give an equal opportunity to each "new receptor" to effectively explore the landscape. We note that for τ_B greater than the maximum number of allowed generations, the IA works essentially without aging operator. In such a limit case the algorithm uses a strong elitist selection strategy. The best B cells which"survived" the aging operator, are selected from the populations $P^{(t)}$, $P^{(hyp)}$ and/or $P^{(macro)}$, in such a way each B cell receptor is unique, i.e. each B cell receptor is different from all other receptors. In this way, we obtain the new population $P^{(t+1)}$, of d B cells, for the next generation $t+1$. If only $d' < d$ B cells survived , the *Elitist_Merge* function creates $d - d'$ new B cells (*Birth phase*). The boolean function *Termination_Condition()* returns true if a solution is found, or a maximum number of fitness function evaluations (T_{max}) is reached. Next we show the pseudo-code of the proposed Immune Algorithm.

```
   opt-IA(ℓ, d, dup, τ_B, c, h, hm)
1.  t := 0
2.  P^(t) := Initial_Pop()
3.  Evaluate(P^(0))
4.  while (¬ Termination_Condition()) do
5.     P^(clo) := Cloning (P^(t), dup)
6.     if (h is TRUE) then
7.        P^(hyp) := Hypermutation(P^(clo), c, ℓ)
8.        Evaluate(P^(hyp))
9.     if (hm is TRUE) then
10.       P^(macro) := Hypermacro(P^clo)
11.       Evaluate (P^(macro))
12.    Pure_Aging(P^(t), P^(hyp), P^(macro), τ_B)
13.    P^(t+1) :=Merge (P^(t), P^(hyp), P^(macro))
14.    t := t + 1
15. end_while
```

The boolean variables h, hm control, respectively, the hypermutation and the hypermacromutation operator.

2 ESCAPING TRAPS

In this section, we will describe the behavior and performances of the Clonal selection algorithms presented in previous section. First, we briefly illustrate the two trap functions, simple and complex trap functions, faced by CLONALG and opt-IA, and then we report the experimental results obtained with different

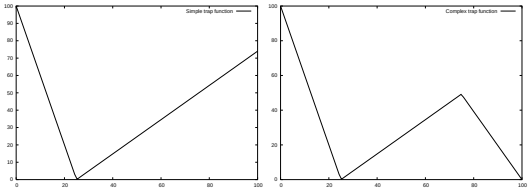

Figure 1: Simple and Complex trap functions.

algorithmic settings. The trap functions (Nijssen S., 2003), simply, take as input the number of 1's of bit strings of length ℓ :

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

For our experiments we will use two trap functions: a *simple trap function* and a *complex trap function*. In the following we give the definition of the simple trap function:

$$\widehat{f}(u) = \begin{cases} \frac{a}{z}(z-u), & \text{if } u \leq z \\ \frac{b}{\ell - z}(u-z), & \text{otherwise.} \end{cases} \tag{2}$$

There are many choice for parameters a, b and z, we will use the parameter values used in (Nijssen S., 2003): $z \approx (1/4)\ell$; $b = \ell - z - 1$; $1.5b \leq a \leq 2b$; a a multiple of z.

The simple trap function is characterize by a global optimum (for a bit string of all 0's) and a local optimum (for a bit string of all 1's) that are the complement *bit-wise* of each other. The complex trap function, how we will see, is more difficult to investigate, in fact there are two directions to get trapped :

$$\widehat{f}(u) = \begin{cases} \frac{a}{z_1}(z_1-u), & \text{if } u \leq z_1 \\ \frac{b}{\ell - z_1}(u-z_1), & \text{if } z_1 < u \leq z_2 \\ \frac{b(z_2-z_1)}{\ell - z_1}\left(1 - \frac{1}{\ell - z_2}(u-z_2)\right) & \text{otherwise.} \end{cases} \tag{3}$$

We note that for $z_2 = \ell$ the complex trap function becomes the simple trap function. In this case the values of parameter z_2 are determines by the following equation $z_2 = \ell - z_1$, with this trick an *ad hoc* operator that mutates all the bit of string does not obtain the global maximum of the complex trap function (see figure 1). The tables of the next section reporting the experimental results will label the trap function with the following syntax: $S(type)$ and $C(type)$; where S and C mean respectively Simple and Complex trap function, while $type$ varying with respect the parameter values used by simple and complex trap functions: type I ($\ell = 10, z = 3, a = 12, b = 6$), type II ($\ell = 20, z = 5, a = 20, b = 14$), type III ($\ell = 50, z = 10, a = 80, b = 39$), type IV ($\ell = 75, z = 20, a = 80, b = 54$), type V ($\ell = 100, z = 25, a = 100, b = 74$). For the complex trap function $z_1 = z$ and $z_2 = \ell - z_1$.

Table 1: The best results obtained by CLONALG$_1$ with population size $N = 10$, varying $d \in \{1, 2, 3, 4, 5\}$.

Trap	$\left(\frac{1}{\rho}\right) e^{(-f)}$			$e^{(-\rho * f)}$		
	SR	AES	(β, ρ)	SR	AES	(β, ρ)
S(I)	100	1100.4	(.5,3)	100	479.7	(.8,2)
S(II)	100	27939.2	(.8,8)	24	174563.4	(.1,4)
S(III)	0	-	-	0	-	-
S(IV)	0	-	-	0	-	-
S(V)	0	-	-	0	-	-
C(I)	100	272.0	(.7,3)	100	251.0	(.9,4)
C(II)	100	17526.3	(1,8)	10	191852.7	(.2,1)
C(III)	0	-	-	0	-	-
C(IV)	0	-	-	0	-	-
C(V)	0	-	-	0	-	-

Table 2: The best results obtained by CLONALG$_2$ with population size $N = 10$, varying $d \in \{1, 2, 3, 4, 5\}$.

Trap	$\left(\frac{1}{\rho}\right) e^{(-f)}$			$e^{(-\rho * f)}$		
	SR	AES	(β, ρ)	SR	AES	(β, ρ)
S(I)	100	725.0	(.9,4)	100	539.2	(.7,2)
S(II)	30	173679.8	(.1,6)	31	172191.2	(.1,4)
S(III)	0	-	-	0	-	-
S(IV)	0	-	-	0	-	-
S(V)	0	-	-	0	-	-
C(I)	100	254.0	(.3,3)	100	218.4	(.5,4)
C(II)	29	173992.6	(.1,6)	24	172434.2	(.1,4)
C(III)	0	-	-	0	-	-
C(IV)	0	-	-	0	-	-
C(V)	0	-	-	0	-	-

CLONALG VS TRAPS. In this paper we use the CLONALG version for multimodal optimization (de Castro L. N., 2002a), varying the same parameters (N, n, β, d, ρ) where ρ controls the shape of the mutation rate with respect to the following two equations:

$$\alpha = e^{(-\rho * f)} \tag{4}$$

$$\alpha = \left(\frac{1}{\rho}\right) e^{(-f)} \tag{5}$$

where α represents the mutation rate , and f is the fitness function value normalized in $[0.1]$ (i.e. antigenic affinity). The number of mutations of the clone with fitness function value f is equal to $\lfloor L * \alpha \rfloor$ where L is the length of the clone receptor. The potential mutation 4 has been proposed in (de Castro L. N., 2002a), the original mutation law used by CLONALG algorithm; while the potential mutation 5 has been introduced in (de Castro L. N., 2002b), a hybrid artificial immune system that combines clonal selection principle and immune network, the opt-aiNET algorithm. Moreover, using the CLONALG version for optimization tasks, the affinity proportionate cloning is not useful, we use the same law defined in (de Castro L. N., 2002a):

$$N_c = \sum_{i=1}^{n} round\,(\beta * N) \tag{6}$$

in 6 N_c represents the total number of clones created at each generation where each antibody (or B cell) produces the same number of clones. We vary β in the range $\{0.1, 0.2, ..., 1.0\}$, ρ in the range $\{1.0, 2.0, ..., 10.0\}$, and the population size N in the set $\{10, 100\}$, assigning $n = N$, that is all Ab's will be selected for cloning.

We conducted our experimental study with two versions of CLONALG, CLONALG$_1$ and CLONALG$_2$, and using the two potential mutations above defined (equations 4 and 5).

Table 3: $opt - IA$ with HyperMacromutation Operator with population size $d = 10$.

Trap	T_{max}	SR	AES	(dup, τ_B)
S(I)	10^5	100	1495.9	$(1, 1)$
S(II)	2×10^5	28	64760.25	$(1, 1)$
S(III)	3×10^5	23	19346.09	$(4, 13)$
S(IV)	4×10^5	28	69987	$(10, 12)$
S(V)	5×10^5	27	139824.41	$(7, 1)$
C(I)	10^5	100	737.78	$(5, 3)$
C(II)	2×10^5	100	27392.18	$(5, 3)$
C(III)	3×10^5	54	115908.61	$(4, 7)$
C(IV)	4×10^5	7	179593.29	$(2, 9)$
C(V)	5×10^5	2	353579	$(1, 15)$

- CLONALG$_1$: each Ab at generation t will be substituted at the next generation $(t + 1)$ by the best individual of its set of $\beta * N$ mutated clones.
- CLONALG$_2$: the population at the next generation $(t + 1)$ will be formed by the n best Ab's of the mutated clones at time step t.

All the experimental results reported in this and the next sections have been averaged over 100 independent runs, varying all the parameter values $(\beta \in \{0.1, 0.2, ..., 1.0\}, \rho \in \{1.0, 2.0, ..., 10.0\}$.

Tables 1 and 2 show the best results obtained respectively by CLONALG$_1$ and CLONALG$_2$ in terms of Success Rate (SR) and Average number of Evaluations to Solutions (AES), varying the rate of newcomers $d \in \{1, 2, 3, 4, 5\}$ while the population size has been set to the minimal value $N = 10$. The third column of the tables reports the best parameter values that allowed the hypermutation operators to reach the best results. In terms of problem solving ability, the results show clearly that facing toy problems is not an easy game.

Table 4: $opt - IA$ with Inversely Proportional Hypermutation Operator with population size $d = 10$ and $dup = 1$.

Trap	T_{max}	SR	AES	(τ_B, c)
S(I)	10^5	100	504.76	$(5, 0.3)$
S(II)	2×10^5	97	58092.7	$(20, 0.2)$
S(III)	3×10^5	0	-	-
S(IV)	4×10^5	0	-	-
S(V)	5×10^5	0	-	-
C(I)	10^5	100	371.15	$(10, 0.2)$
C(II)	2×10^5	100	44079.57	$(10, 0.2)$
C(III)	3×10^5	0	-	-
C(IV)	4×10^5	0	-	-
C(V)	5×10^5	0	-	-

Table 5: $opt - IA$ with Inversely Proportional Hypermutation and HyperMacromutation Operators with population size $d = 10$, and $dup = 1$.

Trap	T_{max}	SR	AES	(τ_B, c)
S(I)	10^5	100	477.04	$(15, 0.2)$
S(II)	2×10^5	100	35312.29	$(100, 0.2)$
S(III)	3×10^5	100	20045.81	$(2 \times 10^5, 0.1)$
S(IV)	4×10^5	100	42089	$(25, 0.2)$
S(V)	5×10^5	100	80789.94	$(50, 0.2)$
C(I)	10^5	100	388.42	$(10, 0.2)$
C(II)	2×10^5	100	29271.68	$(5, 0.2)$
C(III)	3×10^5	24	149006.5	$(20, 0.1)$
C(IV)	4×10^5	2	154925	$(15, 0.4)$
C(V)	5×10^5	0	-	-

OPT-IA VS TRAP FUNCTIONS. In this section we report the experimental results obtained by $opt - IA$ when using a population size $d = 10$, a duplication parameter $dup = 1$, and varying the parameter of maximum number of generations allowed to B cells to remain in the population $\tau_B \in \{1, \ldots, 15, 20, 25, 50, 50, 100, 200, \infty\}$ and parameter $c \in \{0.1, \ldots, 1.0\}$. All results have been averaged on 100 independent runs. Table 3 reports the results of opt-IA when using only the hypermacromutation operator. It is the first time that all cases of the simple and complex trap function have $SR > 0$. Instead table 4 reports the results of opt-IA when using only the inversely proportional hypermutation operator. If we compare these results with the results obtained by CLONALG for population size of 10 Ab's we note how opt-IA outperforms CLONALG. Finally, table 5 shows the results obtained by opt-IA using both perturbation operators. The usage of coupled operators is the key feature to effectively face the trap functions. The same experiments were performed using static hypermutation alone and static hypermutation with hypermacromutation, and the final result is equivalent to that reported in table 5. The results obtained with this setting are comparable with the results in (Nijssen S., 2003), where the authors, in their theoretical and experimental research work, use only cases C(I), C(II) and C(III) for the complex trap function.

3 CONCLUSION

This paper presented an experimental study of clonal selection algorithms to face trap functions. Different settings of the proposed IA's were analyzed in order to detect the key features to properly tackle the given computational problem. The results obtained by CSAa show clearly that facing toy problems is not an easy game. The key feature is the hypermacromutation operator coupled with a traditional perturbation immune operator (for example, static hypermutation or inversely hypermutation) to optimize both simple and trap functions.

REFERENCES

Burnet, F. M. (1959). *The Clonal Selection theory of acquired immunity*. Cambridge University Press, Cambridge.

Cutello V., Nicosia, G. (2002). An immunological approach to combinatorial optimization problems. In *IBERAMIA'02, Proceedigns of the 8th Conference Iberoamericana of Artificial Intelligence*. Springer.

Cutello V., N. G. (2004). The clonal selection principle for in silico and in vitro computing. In de Castro, L. N. and von Zuben, F. J., editors, *Recent Developments in Biologically Inspired Computing*. Idea Group Publishing, Hershey, PA.

de Castro L. N., Von Zuben, F. J. (2002a). Learning and optimization using the clonal selection principle. In *IEEE Trans. on Evolutionary Computation, vol. 6, no. 3, pp. 239-251*. IEEE.

de Castro L. N., Timmis, J. (2002b). *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, London.

Nijssen S., Back, T. (2003). An analysis of the behavior of semplified evolutionary algorithms on trap functions. In *IEEE Trans. on Evolutionary Computation, vol 7, no 1, pp. 11-22*. IEEE.

Prugel-Bennett A., Rogers, A. (2001). Modelling ga dynamics. In *Proceedings Theoretical Aspects of Evolutionary Computing*. Springer.

Timmis J., Neal, M. (2001). A resource limited artificial immune system for data analysis. In *Knowledge Based Systems, vol 14, pp. 121-130*.