A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem

Vincenzo Cutello, Giuseppe Nicosia, and Mario Pavone

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

Abstract. We present a new Immune Algorithm that incorporates a simple local search procedure to improve the overall performances to tackle the graph coloring problem instances. We characterize the algorithm and set its parameters in terms of Information Gain. Experiments will show that the IA we propose is very competitive with the best evolutionary algorithms.

Keywords: Immune Algorithm, Information Gain, Graph coloring problem, Combinatorial optimization.

1 Introduction

In the last five years we have witnessed an increasing number of algorithms, models and results in the field of Artificial Immune Systems [1,2]. Natural Immune System provide an excellent example of bottom up intelligent strategy, in which adaptation operates at the local level of cells and molecules, and useful behavior emerges at the global level, the immune humoral response. From an information processing point of view [3] the Immune System (IS) can be seen as a problem learning and solving system. The *antigen* (Ag) is the problem to solve, the *antibody* (Ab) is the generated solution. At the beginning of the *primary response* the antigen-problem is recognized by poor candidate solution. At the end of the primary response the antigen-problem is defeated-solved by good candidate solutions. Consequently the primary response corresponds to a training phase while the *secondary response* is the testing phase where we will try to solve problems similar to the original presented in the primary response [4].

Recent studies show that when one faces the Graph Coloring Problem (GCP) with evolutionary algorithms (EAs), the best results are often obtained by hybrid EAs with local search and specialized crossover [5]. In particular, the random crossover operator used in a standard genetic algorithm performs poorly for combinatorial optimization problem and, in general, the crossover operator must be designed carefully to identify important properties, building blocks, which must be transmitted from parents population to offspring population. Hence the design of a *good* crossover operator is crucial for the overall performance of the

EAs. The drawback is that is might happen to recombine good individuals from different regions of the search space, having *different symmetries*, producing poor offspring [6]. For this reason, we use an Immunological Algorithm (IA) to tackle the GCP. IAs do not have a crossover operator, and the crucial task of designing an appropriate crossover operator is avoided at once. The IA we will propose makes use of a particular mutation operator and a local search strategy without having to incorporate specific domain knowledge.

For sake of clarity, we recall some basic definitions. Given an undirected graph $G = (V, E)$ with vertex set V, edge set E and a positive integer $K \leq\mid V \mid$, the Graph Coloring Problem asks whether G is K–colorable, i.e. whether there exists a function $f : V \rightarrow \{1, 2, ..., K\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$. The GCP is a well-known NP–complete problem [7]. Exact solutions can be found for simple or medium instances [8,9]. Coloring problems are very closely related with *cliques* [10] (complete subgraphs). The size of the maximum clique is a lower bound on the minimum number of colors needed to color a graph, $\chi(G)$. Thus, if $\omega(G)$ is the size of the maximum clique: $\chi(G) \geq \omega(G)$.

2 Immune Algorithms

We work with a simplified model of the natural immune system. We will see that the IA presented in this work is very similar to De Castro, Von Zuben's algorithm, CLONALG [11,12] and to Nicosia *et al.* immune algorithm [4,13]. We consider only two entities: Ag and *B cells*. Ag is the problem and the B cell receptor is the candidate solution. Formally, Ag is a set of variables that models the problem; and, B cells are defined as strings of integers of finite length $\ell = \mid V \mid$. The input is the antigen–problem, the output is basically the candidate solutions–B cells that solve–recognize the Ag.

By $P^{(t)}$ we will denote a population of d individuals of length ℓ, which represent a subset of the space of feasible solutions of length ℓ, S^{ℓ}, obtained at time t. The initial population of B cells, i.e. the initial set $P^{(0)}$, is created randomly. After initialization, there are three different phases.

In the *Interaction phase* the population $P^{(t)}$ is evaluated. $f(\boldsymbol{x}) = m$ is the fitness function value of B cell receptor $\boldsymbol{x}$. Hence for the GCP, the fitness function $f(\boldsymbol{x}) = m$ indicates that there exists a m–coloring for G, that is, a partition of vertices $V = S_1 \cup S_2 \cup \ldots \cup S_m$ such that each $S_i \subseteq V$ is a subset of vertices which are pairwise not adjacent (i.e. each S_i is an *independent set*).

The *Cloning expansion phase* is composed of two steps: *cloning* and *hypermutation*. The cloning expansion events are modeled by *cloning potential* V and *mutation number* M, which depend upon f. If we exclude all the adaptive mechanisms [14] in EA's (e.g., adaptive mutation and adaptive crossover rates which are related to the fitness function values), the immune operators, contrary to standard evolutionary operators, depend upon the fitness function values[15]. Cloning potential is a truncated exponential: $V(f(\boldsymbol{x})) = e^{-k(\ell - f(\boldsymbol{x}))}$, where the parameter k determines the sharpness of the potential. The cloning operator generates the population P^{clo}. The mutation number is a simple straight line:

$M(f(\boldsymbol{x})) = 1 - (\ell/f(\boldsymbol{x}))$, and this function indicates the number of swaps between vertices in $\boldsymbol{x}$. The mutation operator chooses randomly $M(f(\boldsymbol{x}))$ times two vertices i and j in $\boldsymbol{x}$ and then swaps them. The hypermutation function from population P^{clo} generates the population P^{hyp}. The cell receptor mutation mechanism is modeled by the mutation number M, which is inversely proportional to the fitness function value. The cloning expansion phase triggers the growth of a new population of high–value B cells centered around a higher fitness function value.

In the *Aging phase,* after the evaluation of P^{hyp} at time t, the algorithm eliminates old B cells. Such an elimination process is stochastic, and, specifically, the probability to remove a B cell is governed by an exponential negative law with parameter τ_B, (expected mean life for the B cells): $P_{die}(\tau_B) = (1 - e^{(-\ln(2)/\tau_B)})$. Finally, the new population $P^{(t+1)}$ of d elements is produced. We can use two kinds of Aging phases: *pure aging phase* and *elitist aging phase.* In the elitist aging, when a new population for the next generation is generated, we do not allow the elimination of B cells with the best fitness function. While in the pure aging the best B cells can be eliminate as well. We observe that the exponential rate of aging, $P_{die}(\tau_B)$, and the cloning potential, $V(f(\boldsymbol{x}))$, are inspired by biological processes [16].

Sometimes it might be useful to apply a *birth phase* to increase the population diversity. This extra phase must be combined with an aging phase with a longer expected mean life τ_B. For the GCP we did not use the birth phase because it produced a higher number of fitness function evaluation to solutions.

Assignment colors. To assign colors, the vertices of the solution represented by a B cell are examined and assigned colors, following a deterministic scheme based on the order in which the graph vertices are visited. In details, vertices are examined according to the order given by the B cell and assigned the first color not assigned to adjacent vertices. This method is very simple. In literature there are more complicated and effective methods [5,6,10]. We do not use those methods because we want investigate the learning and solving capability of our IA. In fact, the IA described does not use specific domain knowledge and does not make use of problem-dependent local searches. Thus, our IA can be improved simply including *ad hoc* local search and immunological operators using specific domain knowledge.

2.1 Termination Condition by Information Gain

To analyze the learning process, we use the notion of *Kullback information,* also called *information gain* [17], an entropy function associated to the quantity of information the system discovers during the learning phase. To this end, we define the B cells distribution function $f_m^{(t)}$ as the ratio between the number, B_m^t, of B cells at time t with fitness function value m, (the distance m from the antigen–problem) and the total number of B cells:

$$f_m^{(t)} = \frac{B_m^t}{\sum_{m=0}^{h} B_m^t} = \frac{B_m^t}{d}. \tag{1}$$

It follows that the information gain can be defined as:

$$K(t, t_0) = \sum_m f_m^{(t)} \log(f_m^{(t)}/f_m^{(t_0)}). \tag{2}$$

The gain is the amount of information the system has already learned from the given Ag–problem with respect to initial distribution function (the randomly generated initial population $P^{(t_0=0)}$). Once the learning process starts, the information gain increases monotonically until it reaches a final steady state (see figure 1). This is consistent with the idea of a *maximum information-gain principle* of the form $\frac{dK}{dt} \geq 0$. Since $\frac{dK}{dt} = 0$ when the learning process ends, we use it as a termination condition for the Immune Algorithms. We will see in section 3 that the information gain is a kind of entropy function useful to understand the IA's behavior and to set the IA's parameters.

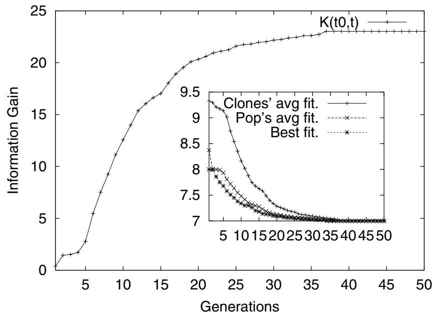

Fig. 1. Information Gain versus generations for the GCP instance QUEEN6_6.

In figure 1 we show the information gain when the IA faces the GCP instance QUEEN6_6 with vertex set $\mid V \mid= 36$, edge set $\mid E \mid= 290$ and optimal coloring 7. In particular, in the inset plot one can see the corresponding average fitness of population P^{hyp}, the average fitness of population $P^{(t+1)}$ and the best fitness value. All the values are averaged on 100 independent runs. Finally, we note that our experimental protocol can have other termination criteria, such as maximum number of evaluations or generations.

2.2 Local Search

Local search algorithms for combinatorial optimization problems generally rely on a definition of neighborhood. In our case, neighbors are generated by swapping vertex values. Every time a proposed swap reduces the number of used colors, it is accepted and we continue with the sequence of swaps, until we explore the neighborhood of all vertices. Swapping all pair of vertices is time consuming, so we use a reduced neighborhood: all $n =\mid V \mid$ vertices are tested for a swap, but only with the closer ones. We define a neighborhood with radius R. Hence we swap all vertices only with their R nearest neighbors, to left and to right. A possible value for radius R is 5. Given the large size of neighborhood and n, we found it convenient to apply the previous local search procedure only on the population's best B cell. We note that if $R = 0$ the local search procedure is not executed. This case is used for simple GCP instances, to avoid unnecessary fitness function evaluations. The local search used is not critical to the searching process. Once a maximum number of generations has been fixed, the local search procedure increases only the success rate on a certain number of independent runs and, as drawback, it increases the average number of evaluations to solutions. However, if we omit it, the IA needs more generations, hence more fitness function evaluations, to obtain the same results of IA using local search.

Table 1. Pseudo–code of Immune Algorithm

```
Immune Algorithm($d, dup, \tau_B, R$)
1. $t := 0$;
2. Initialize $P^{(0)} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_d}\} \in S^\ell$
3. while ( $\frac{dK}{dt} \neq 0$ ) do
4.     Interact(Ag, $P^{(t)}$);                 /* Interaction phase */
5.     $P^{clo}$ := Cloning ($P^{(t)}, dup$);      /* First step Cloning expansion */
6.     $P^{hyp}$ := Hypermutation ($P^{clo}$);   /* Second step Cloning expansion */
7.     Evaluate ($P^{hyp}$);                    /* Compute $P^{hyp}$ fitness function */
8.     $P^{ls}$:=Local_Search($P^{hyp}, R$);      /* LS procedure */
9.     $P^{(t+1)}$:=aging($P^{hyp} \sqcup P^{(t)} \sqcup P^{ls}, \tau_B$); /* Aging Phase */
10.    $K(t, t_0)$:=InformationGain();         /* Compute $K(t, t_0)$ */
11.    $t := t + 1$;
12. end_while
```

In figure 2 we show the fitness function value dynamics. In both plots, we show the dynamics of average fitness of population P^{hyp}, $P^{(t+1)}$, and the best fitness value of population $P^{(t+1)}$. Note that the average fitness of P^{hyp} shows the diversity in the current population, when this value is equal to average fitness of population $P^{(t+1)}$, we are close a premature convergence or in the best case we are reaching a sub–optimal or optimal solution. It is possible to use the difference between P^{hyp} average fitness and $P^{(t+1)}$ average fitness, $\mid avg_{fitness}(P^{hyp}) - avg_{fitness}(P^{(t+1)} \mid = Pop_{div}$ as a standard to measure population diversity. When Pop_{div} rapidly decreases, this is considered as the primary reason for premature convergence. In the left plot we show the IA dynamic when we face the DSCJ250.5.COL GCP instance ($\mid V \mid = 250$ and $\mid E \mid = 15,668$). We execute the algorithm with population size $d = 500$, duplication parameter $dup = 5$, expected mean life $\tau_B = 10.0$ and neighborhood's radius $R = 5$. For this instance we use *pure aging* and obtain the optimal coloring. In the right plot

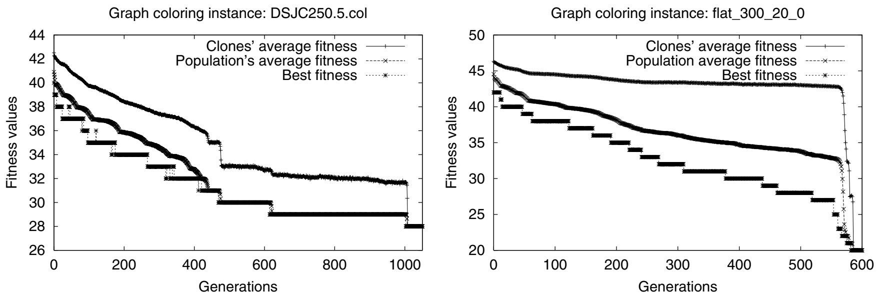

Fig. 2. Average fitness of population P^{hyp}, average fitness of population $P^{(t+1)}$, and best fitness value vs generations. Left plot: IA with pure aging phase. Right plot: IA with elitist aging

we tackle the FLAT_300_20 GCP instance ($| V |= 300$ and $| E |= 21,375$), with the following IA's parameters: $d = 1000$, $dup = 10$, $\tau_B = 10.0$ and $R = 5$. For this instance the optimal coloring is obtained using *elitist aging*. In general, with elitist aging the convergence is faster, even though it can trap the algorithm in a local optimum. Although, with pure aging the convergence is slower and the population diversity is higher, our experimental results indicate that elitist aging seems to work well. We can define the ratio $S_p = \frac{1}{dup}$ as the *selective pressure* of the algorithm: when $dup = 1$, obviously we have that $S_p = 1$ and the selective pressure is low, while increasing *dup* we increase the IA's selective pressure. Experimental results show that high values of d denote high clones population average fitness and, in turn, high population diversity but, also, a high computational effort during the evolution.

3 Parameters Tuning by Information Gain

To understand how to set the IA parameters, we performed some experiments it with the GCP instance QUEEN6_6. Firstly, we want to set the B cell's mean life, τ_B. We fix the population size $d = 100$, duplication parameter $dup = 2$, local search radius $R = 2$ and total generations $gen = 100$. For each experiment we performed $runs = 100$ independent runs.

3.1 B Cell's Mean Life, τ_B

In figure 3 we can see the best fitness values (left plot) and the Information Gain (right plot) with respect the following τ_B values {1.0,5.0,15.0,25.0,1000.0}. When $\tau_B = 1.0$ the B cells have a shorter mean life, only one time step, and with this value the IA performed poorly. With $\tau_B = 1.0$ the maximum information gain obtained at generation 100 is about 13. As τ_B increases, the best fitness values decreases and the Information Gain increases. The best value for τ_B is 25.0. With $\tau_B = 1000.0$, and in general when τ_B is greater than a number of fixed

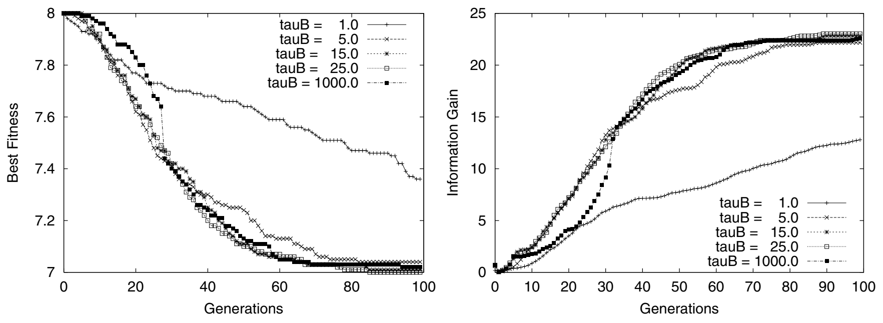

Fig. 3. Best fitness values and Information Gain vs generations.

generations *gen*, we can consider B cells mean life infinite and obtain a pure elitist selection scheme. In this special case, the behavior of IA shows slower convergence in the first 30 generations in both plots. For values of τ_B greater than 25.0 we obtain slightly worse results. Moreover, when $\tau_B \leq 10$ the success rate (SR) on 100 independent runs is less than 98 while when $\tau_B \geq 10$ the IA obtains a SR=100 with a lower Average number of Evaluations to Solution (AES) located when $\tau_B = 25.0$.

3.2 Duplication Parameter Dup

Now we fix $\tau_B = 25.0$ and vary *dup*. In fig.4 (left plot) we note that the IA obtains quickly more Information Gain at each generation with $dup = 10$, moreover it reaches faster the best fitness value with $dup = 5$. With both values of dup the

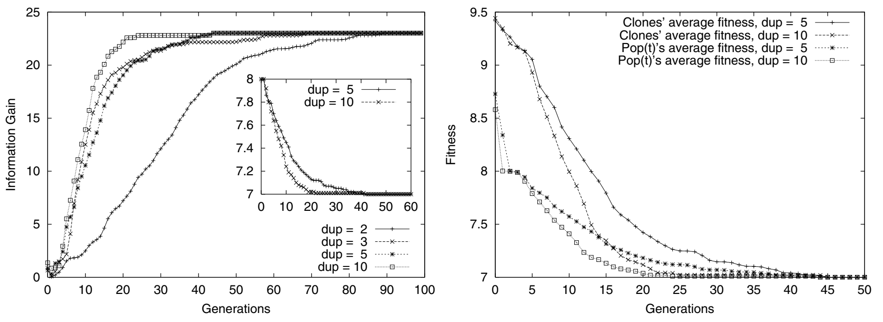

Fig. 4. Left plot, Information Gain and Best fitness value for *dup*. Right plot, average fitness of Clones and $Pop^{(t)}$ for $dup \in \{5, 10\}$.

largest information gain is obtained at generation 43. Moreover, with $dup = 10$ the best fitness is obtained at generation 22, whereas with $dup = 5$ at generation 40. One may deduce that $dup = 10$ is the best value for the cloning of B cells since we obtain faster more information gain. This is not always true. Indeed, if we observe figure 4 (right plot) we can see how the IA with *dup* = 5 obtains a larger amount of clones average fitness and hence a greater diversity. This characteristic can be useful in avoiding premature convergence and in finding more optimal solutions for a given combinatorial problem.

3.3 Dup and τ_B

In 3.1 we saw that for $dup = 2$, the best value of τ_B is 25.0. Moreover, in 3.2 experimental results show better performance for $dup = 5$. If we set $dup = 5$ and vary τ_B, we obtain the results in fig.5. We can see that for $\tau_B = 15$ we reach the maximum Information Gain at generation 40 (left plot) and more diversity (right plot). Hence, when $dup = 2$ the best value of τ_B is 25.0, i.e. on average we need 25 generations for the B cells to reach a mature state. On the other hand, when $dup = 5$ the correct value is 15.0 Thus, increasing dup the average time for the population of B cells to reach a mature state decreases.

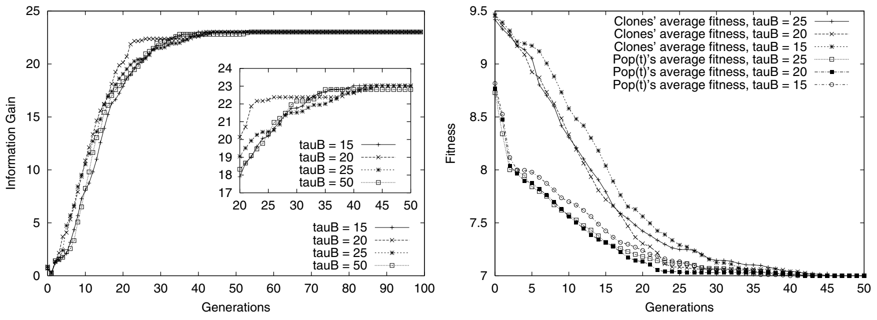

Fig. 5. Left plot Information Gain for $\tau_b \in \{15, 20, 25, 50\}$. Right plot average fitness of population P^{hyp} and population $P^{(t)}$ for $\tau_b \in \{15, 20, 25\}$

3.4 Neighborhood's Radius R, d and Dup

Local search is useful for large instances (see table 2). The cost of local search, though, is high. In figure 6 (left plot) we can see how the AES increases as the neighborhood radius increases. The plot reports two classes of experiments performed with 1000 and 10000 independent runs. In figure 6 (right plot) we show the values of parameters *d* and *dup* as functions of the Success Rate (SR). Each point has been obtained averaging 1000 independent runs. How we can see there is a certain relation between *d* and *dup* in order to reach a $SR = 100$. For the QUEEN6_6 instance, for low values for the population we need a high value of dup to reach $SR = 100$. For $d = 10$, $dup = 10$ is not sufficient to obtain the maximum SR. On the other hand, as the population number increases, we need smaller values for dup. Small values of dup are a positive factor.

Table 2. Mycielsky and Queen graph instances. We fixed $\tau_B = 25.0$, and the number of independent runs 100. *OC* denotes the Optimal Coloring.

Instance G	$\mid V \mid$	$\mid E \mid$	OC	(d,dup,R)	Best Found	AES
Myciel3	11	20	4	(10,2,0)	4	30
Myciel4	23	71	5	(10,2,0)	5	30
Myciel5	47	236	6	(10,2,0)	6	30
Queen5_5	25	320	5	(10,2,0)	5	30
Queen6_6	36	580	7	(50,5,0)	7	3750
Queen7_7	49	952	7	(60,5,0)	7	11,820
Queen8_8	64	1,456	9	(100,15,0)	9	78,520
Queen8_12	96	2,736	12	(500,30,0)	12	908,000
Queen9_9	81	1,056	10	(500,15,0)	10	445,000
School1_nsh	352	14,612	14	(1000,5,5)	15	2,750,000
School1	385	19,095	9	(1000,10,10)	14	3,350,000

We recall that dup is similar to the temperature in Simulated Annealing [18]. Low values of dup corresponds to a system that cools down slowly and has a high *EAS*.

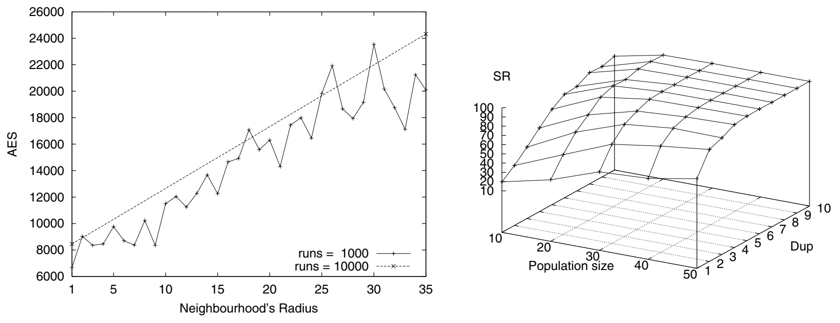

Fig. 6. Left plot: Average number of Evaluations to Solutions versus neighborhood's radius. Right plot: 3D plot of *d*, *dup* versus Success Rate (SR).

4 Results

In this section we report our experimental results. We worked with classical benchmark graph [10]: the Mycielski, Queen, DSJC and Leighton GCP instances. Results are reported in Tables 2 and 3. In these experiments the IA's best found value is always obtained $SR = 100$. For all the results presented in this section, we used *elitist aging*. In tables 4 and 5 we compare our IA with two of the best evolutionary algorithms, respectively Evolve_AO algorithm [19] and the

Table 3. Experimental results on subset instances of DSJC and Leighton graphs. We fixed $\tau_B = 15.0$, and the number of independent runs 10.

Instance G	$\mid V \mid$	$\mid E \mid$	OC	(d,dup,R)	Best Found	AES
DSJC125.1	125	736	5	(1000,5,5)	5	1,308,000
DSJC125.5	125	3,891	12	(1000,5,5)	18	1,620,000
DSJC125.9	125	6,961	30	(1000,5,10)	44	2,400,000
DSJC250.1	250	3,218	8	(400,5,5)	9	1,850,000
DSJC250.5	250	15,668	13	(500,5,5)	28	2,500,000
DSJC250.9	250	27,897	35	(1000,15,10)	74	4,250,000
le450_15a	450	8,168	15	(1000,5,5)	15	5,800,000
le450_15b	450	8,169	15	(1000,5,5)	15	6,010,000
le450_15c	450	16,680	15	(1000,15,10)	15	10,645,000
le450_15d	450	16,750	9	(1000,15,10)	16	12,970,000

HCA algorithm [5]. For all the GCP instances we ran the IA with the following parameters: $d = 1000$, $dup = 15$, $R = 30$, and $\tau_B = 20.0$. For these classes of experiments the goal is to obtain the best possible coloring, no matter the value of AES. Table 4 shows how the IA outperform the Evolve_AO algorithm, while is similar in results to HCA algorithm and better in SR values (see table 5).

Table 4. IA versus Evolve_AO Algorithm. The values are averaged on 5 independent runs.

Instance G	$\chi(G)$	Best–Known	Evolve_AO	IA	Difference
DSJC125.5	12	12	17.2	18.0	+ 0.8
DSJC250.5	13	13	29.1	28.0	-0.9
flat300_20_0	≤ 20	20	26.0	20.0	-6.0
flat300_26_0	≤ 26	26	31.0	27.0	-4.0
flat300_28_0	≤ 28	29	33.0	32.0	-1.0
le450_15a	15	15	15.0	15.0	0
le450_15b	15	15	15.0	15.0	0
le450_15c	15	15	16.0	15.0	-1.0
le450_15d	15	15	19.0	16.0	-3.0
mulsol.i.1	–	49	49.0	49.0	0
school1_nsh	≤ 14	14	14.0	15.0	+1.0

5 Conclusions

We have designed a new IA that incorporates a simple local search procedure to improve the overall performances to tackle the GCP instances. The IA presented has only four parameters. To set correctly these parameters we use the Information Gain function, a particular entropy function useful to understand

Table 5. IA versus Hao *et al.*'s HCA algorithm. The number of independent runs is 10.

Instance G	HCA's Best–Found and (SR)	IA's Best–Found and (SR)
DSJC250.5	28 (90)	28 (100)
flat300_28_0	31 (60)	32 (100)
le450_15c	15 (60)	15 (100)
le450_25c	26 (100)	25 (100)

the IA's behavior. The Information Gain measures the quantity of information that the system discovers during the learning process. We choose the parameters that maximize the information discovered and that increases moderately the information gain monotonically. To our knowledge, this is the first time that IAs, and in general the EAs, are characterized in terms of information gain. We define the average fitness of population P^{hyp} as the diversity in the current population, when this value is equal to average fitness of population $P^{(t+1)}$, we are close a premature convergence. Using a simple coloring method we have investigated the IA's learning and solving capability. The experimental results show how the proposed IA is comparable to and, in many GCP instances, outperforms the best evolutionary algorithms. Finally, the designed IA is directed to solving GCP instances although the solutions' representation and the variation operators are applicable more generally, for example Travelling Salesman Problem.

Acknowledgments. The authors wish to thank the anonymous referees for their excellent revision work. GN wishes to thank the University of Catania project "Young Researcher" for partial support and is grateful to Prof. A. M. Anile for his kind encouragement and support.

References

1. Dasgupta, D. (ed.): Artificial Immune Systems and their Applications. Springer-Verlag, Berlin Heidelberg New York (1999)
2. De Castro L.N., Timmis J.: Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer-Verlag, UK (2002)
3. Forrest, S., Hofmeyr, S. A.: Immunology as Information Processing. Design Principles for Immune System & Other Distributed Autonomous Systems. Oxford Univ. Press, New York (2000)
4. Nicosia, G., Castiglione, F., Motta, S.: Pattern Recognition by primary and secondary response of an Artificial Immune System. Theory in Biosciences **120** (2001) 93–106
5. Galinier, P., Hao, J.: Hybrid Evolutionary Algorithms for Graph Coloring. Journal of Combinatorial Optimization Vol. 3 **4** (1999) 379–397
6. Marino, A., Damper, R.I.: Breaking the Symmetry of the Graph Colouring Problem with Genetic Algorithms. Workshop Proc. of the Genetic and Evolutionary Computation Conference (GECCO'00). Las Vegas, NV: Morgan Kaufmann (2000)

7. Garey, M.R., Johnson, D.S.: Computers and Intractability: a Guide to the Theory of NP-completeness. Freeman, New York (1979)
8. Mehrotra, A., Trick, M.A.: A Column Generation Approach for Graph Coloring. INFORMS J. on Computing 8 (1996) 344–354
9. Caramia, M., Dell'Olmo, P.: Iterative Coloring Extension of a Maximum Clique. Naval Research Logistics, 48 (2001) 518–550
10. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge. American Mathematical Society, Providence, RI (1996)
11. De Castro, L. N., Von Zuben, F. J.: The Clonal Selection Algorithm with Engineering Applications. Proceedings of GECCO 2000, Workshop on Artificial Immune Systems and Their Applications, (2000) 36–37
12. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Trans. on Evolutionary Computation Vol. 6 **3** (2002) 239–251
13. Nicosia, G., Castiglione, F., Motta, S.: Pattern Recognition with a Multi–Agent model of the Immune System. Int. NAISO Symposium (ENAIS'2001). Dubai, U.A.E. ICSC Academic Press, (2001) 788–794
14. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. on Evolutionary Computation, Vol. 3 **2** (1999) 124–141
15. Leung, K., Duan, Q., Xu, Z., Wong, C.W.: A New Model of Simulated Evolutionary Computation – Convergence Analysis and Specifications. IEEE Trans. on Evolutionary Computation Vol. 5 **1** (2001) 3–16
16. Seiden P.E., Celada F.: A Model for Simulating Cognate Recognition and Response in the Immune System. J. Theor. Biol. Vol. 158 (1992) 329–357
17. Nicosia, G., Cutello, V.: Multiple Learning using Immune Algorithms. Proceedings of the 4th International Conference on Recent Advances in Soft Computing, RASC 2002, Nottingham, UK, 12–13 December (2002)
18. Johnson, D.R., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. Operations Research 39 (1991) 378–406
19. Barbosa, V.C., Assis, C.A.G., do Nascimento, J.O.: Two Novel Evolutionary Formulations of the Graph Coloring Problem. Journal of Combinatorial Optimization (to appear)