2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Swarm Intelligence Heuristics for Graph Coloring Problem

Piero Consoli, Alessio Collerà, Mario Pavone

Abstract— **In this research work we present two novel swarm heuristics based respectively on the ants and bees artificial colonies, called AS-GCP and ABC-GCP. The first is based mainly on the combination of Greedy Partitioning Crossover (GPX), and a local search approach that interact with the pheromone trails system; the last, instead, has as strengths three evolutionary operators, such as a mutation operator; an improved version of GPX and a Temperature mechanism. The aim of this work is to evaluate the efficiency and robustness of both developed swarm heuristics, in order to solve the classical Graph Coloring Problem (GCP). Many experiments have been performed in order to study what is the real contribution of variants and novelty designed both in AS-GCP and ABC-GCP. A first study has been conducted with the purpose for setting the best parameters tuning, and analyze the running time for both algorithms. Once done that, both swarm heuristics have been compared with 15 different algorithms using the classical DIMACS benchmark. Inspecting all the experiments done is possible to say that AS-GCP and ABC-GCP are very competitive with all compared algorithms, demonstrating thus the goodness of the variants and novelty designed. Moreover, focusing only on the comparison among AS-GCP and ABC-GCP is possible to claim that, albeit both seem to be suitable to solve the GCP, they show different features: AS-GCP presents a quickly convergence towards good solutions, reaching often the best coloring; ABC-GCP, instead, shows performances more robust, mainly in graphs with a more dense, and complex topology. Finally, ABC-GCP in the overall has showed to be more competitive with all compared algorithms than AS-GCP as average of the best colors found.**

I. Introduction

Swarm Intelligence represents the family of all algorithms that take directly inspiration by the collective intelligent behavior emerging from the aggregation of species colonies (flocks of birds; schools of fish; colonies of ants; and swarms of bees) in order to solve complex problems. The strength of these heuristics is given from the self-organization of each specie, and from a total deficiency of centralized supervision. Swarm Intelligence had a broad diffusion with successful applications in many research areas, thanks to the introduction of the *ant-based algorithms* – Ant System (AS), Ant Colony System (ACS), and Ant Colony Optimization (ACO)) [1] – and *Particle Swarm Optimization* (PSO) algorithms [2], inspired by the social behavior of bird flocks and fish schools. Recently two new swarm intelligence models have been successful applied in numerical and combinatorial optimization tasks, as the *bee-based algorithm* (Artificial Bee Colony - ABC) [3], and the algorithm based on the *calling behavior of Japanese tree frogs* [4]. The reason of this great diffusion is explained with the high reliability that Swarm Intelligence heuristics offer, and to their simplicity, being the agents quite basic units that do not need a deep knowledge of the problem they are trying to solve. Although the swarm-based algorithms have been successfully applied in many research fields, they seem to be more suitable primarily for optimization tasks. Moreover, inspecting the several works in literature is possible to claim that ACS seems to be the most efficient and robust of the family; whilst ABC, albeit is a younger model, shows competitive performances in multi-modal, and in multidimensional optimization tasks. For these two reasons we have developed two novel heuristics of ACS and ABC, and we have used the classical Graph Coloring Problem (GCP) for evaluate and compare the effectiveness and robustness of both designed algorithms.

The graph coloring is one of the most studied combinatorial optimization problems since it lies applicability in many real-world problems, particularly the ones that can be modeled by networks and graphs, such as wireless ad-hoc networks [5]; detection of mobile objects and reduction of signaling actuators [6], manufacturing [7], register allocation [9], frequency assignment [8], and many others. Given a graph $G = (V, E)$ with V the vertices set, and E the edges set, a coloring of G is formally defined as an assignment of one color for each vertex such that two adjacent vertices must have different colors. This is equivalent in partitioning the vertex set into groups in such way two vertices connected by an edge are not allowed to be in the same class. Let C the set of colors; the division in classes of the vertices is given by a coloring $\varphi : V \rightarrow C$, where $\varphi(x) \neq \varphi(y)$ for each $(x, y) \in E$. Every *color class* is an independent set of vertices, where no couple of vertices inside of it is connected by an edge. If C has cardinality k, G is said k-*colorable*; if, furthermore, k is own the minimum value then k is called *Chromatic Number* ($\chi(G)$), and G is said k-*chromatic*. Computing the chromatic number in a graph is a NP–complete problem [10].

The aim of this research work is of course to study, understand and evaluate the performances of both developed heuristics, in term of quality of the solution found; efficiency and robustness; search capabilities into the space of solutions; and comparison with the state-of-the-art for the bio-inspired algorithms in GCP. To this end, we have used the well–known DIMACS benchmark for our experiments being the most studied, where a great competitiveness exists.

II. Ant Colony Systems and Artificial Bee Colony

Swarm intelligence systems are today one of the most efficient and robust algorithmic methods for optimization

P. Consoli is with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK (email: p.a.consoli@cs.bham.ac.uk).
A. Collerà, and M. Pavone are with the Department of Mathematics and Computer Science, University of Catania, v.le A. Doria 6, 95125 Catania, Italy (email: alessio.collera@gmail.com; mpavone@dmi.unict.it).

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

tasks, where Ant Colony Systems and Artificial Bee Colony represent the most competitive heuristics of the family. We have developed two novel variants of them with the aim to tackle, and solve the Graph Coloring Problem that is one of the most classical combinatorial optimization problems. We will call them respectively AS-GCP and ABC-GCP. Both developed heuristics are based on two common strategies; (1) randomized RLF algorithm (*Recursive Largest First* [22]), for the assignment of the colors, and (2) Greedy Partitioning Crossover (GPX) [12] in order to generate better solutions. In the overall, the Ant Colony System proposed is an improved version of ANTCOL [11] that differs from it for the fitness function used, and for the combination of GPX with a local search, which interact with the pheromone trails system. The Artificial Bee Colony designed, instead, is based on three main operators: a mutation operator; an improved version of GPX; and a Temperature mechanism.

A. AS-GCP: an Ant Colony System for GCP

The backbone of AS-GCP is the ANT_RLF procedure, which has been directly derived from ANTCOL [11]. However, AS-GCP differentiates itself from the latter in two main steps: (1) it is based of a different fitness function; (2) after the ANT_RLF procedure the algorithm tries to improve his solutions via a combination of a crossover operator with a local search operator. This combination is able to interact with the pheromone trails system. The randomized ANT_RLF is a procedure that combines the good heuristic of RLF with the pheromone trails of the artificial ants. During the process, a randomized selection of the next vertex to be colored is performed, taking account of both local optimality and pheromone trails. Therefore the probability to select a node is proportional to τ_{ik}^{α} and η_{ik}^{β}, where τ_{ik} is the pheromone associated to the assignment of the vertex i to the color class k; η_{ik} is the local optimal value $deg_B(i)$ induced by a color class k; α and β are two control parameters used to modify the probability in favor of either the local optimality or the pheromone trail. The probability to select a node is then given by the formula:

$$P_{i,k} = \frac{\tau_{ik}^{\alpha}\eta_{ik}^{\beta}}{\sum_{j \in W} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}} \quad . \tag{1}$$

One of the novelty introduced in AS-GCP is the use of a fitness function which takes account of both the number of colors, and length of each color class, in order to slightly favor the classes with a higher cardinality during the pheromone deposit process. It's given by:

$$f(\vec{x}) = \frac{1}{||V| - |C_k|| \cdot \sqrt{c(\vec{x})}} \quad , \tag{2}$$

with C_k the class of color k; and $c(\vec{x})$ the number of colors of the solution $\vec{x}$. After the creation, and evaluation of each solution, the Trail matrix is updated. The algorithm increases the pheromone in $M_{r,s}$ for any solution where the vertices u and v, with $(u, v) \notin E$, are assigned to the same color class. Let k a valid color, v an uncolored vertex, and C_k a colorclass, then the pheromone trail is given by:

$$\tau_{v,k} = \begin{cases} 1, & \text{if } C_k \text{ is empty} \\ \frac{\sum_{u \in C_k} M_{u,v}}{|C_k|}, & \text{otherwise.} \end{cases} \tag{3}$$

An evaporation mechanism is subsequently applied to the whole matrix, decreasing its values according to an evaporation factor ρ, according to the formula: $M_{r,s} = (1-\rho)M_{r,s}$. Once performed the updating of the pheromone along the trails matrix, each ant that represents a candidate solution is undergoes to the crossover operator, and a local search in order to refine the solutions found so far. The developed crossover operator is inspired by the GPX [12], and its novelty is given by the integration of the local search operator in the process of partitioning. This feature is introduced in order to avoid an excessive fragmentation of the color classes during the operation, due to the constant removal of vertices from within the color classes. Therefore, whenever a color class is chosen to be copied to the new solution and its cardinality is below a threshold, the algorithm consolidates the partial solutions via a local search. However, to limit the computational time of this procedure the number of trials is set to very few units. Once the new solution is generated, if the number of its colors is less or equal to one of an its parent, then the pheromone matrix is updated. The proposed local search operator recursively tries to decrease the number of colors of each solution within a certain amount of trials. In case of success, then the new solution will replace the old one. In order to limit the impact of this operator, the higher bound to the number of recursive calls is ruled by the parameter *minRec*; if the operator exceeds this value, then only the swaps who generated a conflicts number ≤ 1 are accepted. If nevertheless the number of recursive calls grows beyond the parameter *maxRec*, then the operator aborts the process. The designed AS-GCP is summarized in the pseudocode below (Algorithm 1).

Algorithm 1 AS-GCP (p, a, b, minRic, maxRic, pLim)

```
Initialize trail matrix
repeat
   ANT_RLF
   Update Trail Matrix
   GPX Crossover
   LocalSearch
   Update Trail Matrix
until (termination criteria is satisfied)
```

AS-GCP represents an advanced variant of ANTCOL, since it is able to find better solutions in term of best coloring, and fitness function evaluations needed to reach the best coloring. Table I shows the compared results between the two algorithms on several DIMACS instances [16] in order to evaluate the goodness of the introduced novelties. Analyzing their comparison is easy to see the several improvements produced by AS-GCP with respect ANTCOL. In particular, AS-GCP outperforms ANTCOL in all instances, finding always the best coloring with a smaller number of fitness

TABLE I

COMPARISON BETWEEN ANTCOL AND AS-GCP ON DIMACS INSTANCES [16] IN ORDER TO POINT OUT THE IMPROVEMENTS PRODUCED BY NOVELTIES INTRODUCED IN AS-GCP.

		ANTCOL		AS-GCP	
instance	k	bf	$SR \pm AES$	bf	$SR \pm AES$
Queen6_6	7	7	$100\% \pm 956$	7	$\mathbf{100\% \pm 96}$
Queen7_7	7	7	$61\% \pm 1991$	7	$\mathbf{100\% \pm 700}$
Queen8_8	9	9	$48\% \pm 1294$	9	$\mathbf{100\% \pm 203}$
Queen8_12	12	12	$87\% \pm 6625$	12	$\mathbf{100\% \pm 162}$
Queen9_9	10	10	$60\% \pm 2614$	10	$\mathbf{100\% \pm 1022}$
School1.nsh	14	14	$100\% \pm 2966$	14	$\mathbf{100\% \pm 1417}$
School1	14	14	$100\% \pm 3120$	14	$\mathbf{100\% \pm 922}$
DSJC125.1	5	5	$40\% \pm 9310$	5	$\mathbf{100\% \pm 160}$
DSJC125.5	12	18	$30\% \pm 9751$	**17**	$100\% \pm 4590$
DSJC125.9	30	44	$40\% \pm 10112$	44	$\mathbf{100\% \pm 1128}$
DSJC250.1	8	9	$100\% \pm 2561$	**8**	$100\% \pm 21773$
DSJC250.5	13	31	$100\% \pm 19238$	**29**	$30\% \pm 24156$
DSJC250.9	35	75	$40\% \pm 9274$	**73**	$40\% \pm 7554$
le450_15a	15	16	$100\% \pm 7913$	16	$\mathbf{100\% \pm 1600}$
le450_15b	15	16	$100\% \pm 3872$	16	$\mathbf{100\% \pm 1052}$
le450_15c	15	15	$100\% \pm 12422$	15	$\mathbf{100\% \pm 4300}$
le450_15d	15	15	$100\% \pm 12755$	15	$\mathbf{100\% \pm 8604}$
flat300_20	20	20	$100\% \pm 8188$	20	$\mathbf{100\% \pm 6394}$
flat300_26	26	34	$60\% \pm 21548$	**32**	$60\% \pm 10780$
flat300_28	28	34	$70\% \pm 19120$	**32**	$60\% \pm 9186$

function evaluations. Moreover, AS-GCP is able to decrease the colors number in 6 instances, whilst in other 6 instances it increases the success rate (SR). All experiments were made on 100 independent runs for queen and school instances, and on 10 runs for the remaining ones.

B. ABC-GCP: an Artificial Bee Colony for GCP

The ABC-GCP algorithm is structured as follows. During an initialization phase, a population of solutions is generated using two different operators, namely a randomized RLF and a simple random order operator. During the *Employed Bees Phase*, the algorithm tries to improve each solution using the mutation, and crossover operators. If the newly generated solution has a higher fitness value than the original one, it takes its place, resetting the counter of the improvement trials for this solution. Otherwise, this counter is increased. In the next phase, the *Onlooker Bees* choose the solution to exploit with a roulette wheel selection. The same operations performed in the previous phase are now performed on the onlooker bees. In the *Scout Bees Phase*, all the solutions whose improvement trials counter is greater than a certain parameter are replaced by a new solution, by means of the same operators used during the initialization phase. Moreover, a temperature mechanism that controls some of the parameters of the algorithm is updated. During the initialization phase, the solutions are generated making use of two different operators: Recursively Largest First (RLF) [22], and the classical random generation. In the first one, the coloring is created assigning each color to a *stable set* until all nodes are colored. Given W the set of the uncolored nodes available for insertion in the stable set, and B the uncolored nodes that cannot be included, the next node to color v is the one in W with the greatest deg_B, where deg_B is the degree of v in the graph induced by B. A solution generated with the second operator, basically is a random permutation of all vertices of the graph, which indicates the visit order of the graph for the assignment of colors. For each vertex in this order, the minimum available color is then assigned among all the available ones. The choice of a good initialization operator is fundamental for the outcome of the algorithm, for its convergence speed and for maintains a good diversity inside the population. An algorithm with a simple naive heuristic, as the random order operator, is most likely to explore a much broader section of the search space, producing equally bad solutions. Vice versa, a smart operator like RLF will produce some good solutions from a too narrow portion of the search space. A study not included in this paper proves how this dramatically affects the final results of the algorithm in terms of best coloring found, success rate (SR), and average number of evaluations to solution (AES). A mixed population is therefore a good trade-off between the exploitation and exploration of the search space, inheriting the good solutions from RLF and the variety of the population from the random order operator. In addition, it allows avoiding the use of a control mechanism over diversity. In Employed Bees phase, the algorithm makes use of a mutation operator to try to escape local optima, together with a crossover operator in order to transmit good structures among solutions. The aim of the mutation operator is to handle the most troublesome nodes. Therefore, the operator swaps the nodes of the smallest color class, where the troublesome nodes are most likely going to be, with the ones of the biggest classes. In order to allow such a transformation, we admit the presence of a certain number of conflicts in the smallest color class, provided that their number is lower than the value of the parameter *partLimit*. After the mutation operator, the well-known Greedy Partitioning Crossover – GPX [12] is performed. It works as follows: each parent solution is selected with a round robin criterion and its biggest color class is copied in the new solution. All copied nodes are therefore removed from the existing color classes of the parent solutions. The algorithm proceeds in this way until only color classes with a cardinality of 1 are left. In such case, the operator tries to insert these nodes in the existing color classes. In ABC-GCP we have designed a variant of GPX, which differs basically in two aspects. Firstly, it is a multi-parent operator. Secondly, the cardinality of the color classes that can be copied in the new solution has a lower bound corresponding to a parameter of the algorithm. In this way, the operator is forced to transmit only the best color classes to the child solutions. Moreover, the previously generated conflicts lying in the smallest color class are automatically eliminated. An analysis conducted comparing several different versions of the algorithm confirmed us how these improvements greatly contributed on its performance. The number of onlooker bees recruited by each solution is proportionally determined according to its fitness, which is proportional to the number of used color classes. Thus, a good solution will likely have a higher number of onlooker bees, which means a more complete exploration of its neighborhood. For this reason, this is a pure exploitation phase where the highest number of improvements takes place.

Nevertheless, one must pay particular attention to avoid the premature exhaustion of the best solutions, before their good structures could be transmitted to the next generations, due to a too high number of unsuccessful trials. The probability of recruiting a onlooker bee for the solution s is:

$$p(s) = \frac{f(s)}{\sum_{i \in P} f(i)} \quad (4)$$

where P is the set of the solutions associated to the employed bees population. During the last phase (Scout Bees phase) of the algorithm, all the solutions whose improvement trials number has exceeded a set limit are replaced with new ones. This is basically a phase of pure exploration, where the new solutions will hopefully introduce new information leading to better solutions. Some of the parameters of the algorithm are controlled by a *temperature mechanism*. This allows binding certain values of these parameters to different steps of the research. One will prefer a much narrower search when the algorithm has just discovered a new best solution, while a wider one would be preferable after many unsuccessful generations. The parameters connected to the variation of the temperature are the number of parents involved in the crossover operator ($partSol$), the number of the improvement trails needed to replace a solution ($evLimit$), the number of scout bees ($nScouts$) and the percentage of the solutions generated by the RLF operator during the scout bees phase ($percSol$). The temperature is set to its maximum value whenever a new best solution is found. Similarly, the parameters $partSol$, $evLimit$, $nScouts$ and $percSol$ assume their highest values, which are $[100, 20, 5, 100\%]$. If no other new best solutions is found, the temperature linearly decreases after each generation, and after about 1000 generations it will reach its minimum which corresponds to the values $[10, 5, 2, 10\%]$. Several experiments have proved how besides simplifying its use through the reduction of the number of control parameters, this simple mechanism also successfully guides the search contributing to its overall efficiency. Below (Algorithm 2), we summarize the main structure of ABC-GCP designed via a classical pseudo-code.

Algorithm 2 ABC-GCP (popSize, partLimit, percEmp)

```
Initialize population
repeat
   Employed Bees Phase
        Mutation
        GPX Crossover
   Onlooker Bees Phase
        Mutation
        GPX Crossover
   Scout Bees Phase
        UpdateTemp
until (termination criteria is satisfied)
```

III. Results

In this section we present all experiments done in order to: (1) evaluate the performances of the both heuristics; (2) impact of the variants, and novelties introduced; (3) efficiency and robustness; (4) search capabilities; finally (5) attempts to understand which of the two swarm heuristics seems to be more suitable for coloring a graph. We present, first, a study on the dynamic performances for both swarm heuristics, from which we have determined the best tuning of the parameters; a further analysis on the running time has been performed using the *time-to-target* plots [23]. As last, we show the experimental comparisons conducted between the two developed heuristics, and with many other algorithms, both deterministic and nature-inspired.

As a first step, many experiments have been performed in order to find the best tuning of the parameters for both algorithms, from which is possible also to identify the real contribution of each parameter. Six are the parameters that influence the AS-GCP performances (α, β, ρ, $minRec$, $maxRec$, and $partLim$), whilst only three for ABC-GCP ($popSize$, $percEmp$ and $partLimit$). The six control parameters of AS-GCP have been studied on the instance *le450_15c* of the DIMACS benchmark [16], and the relative dynamic behaviors are showed in figure 1. Each experiment was averaged over 10 independent runs, and termination criteria was set to $T_{max} = 10^5$ fitness function evaluations allowed. If, however, AS-GCP finds the chromatic number, or the best coloring known, before than T_{max}, then it ends the current run, and goes to the next one. The α parameter rules the influence of the pheromone trails during the decision of the next vertex to color. It is used in combination with β parameter in order to balance the influence of pheromone trail, and local optimality. When this value is set to 0 the algorithm will ignore the pheromone trails. This is clearly a bad choice, since the algorithm is not able to find a better coloring than 24 (see figure 1, left plot in $1st$ line). Similarly, a value of this parameter equal to 1 is not enough to balance the influence of β parameter set to 4. Thus, the best results are reached when both parameters assume the value of 4. The β parameter has an analogous behavior (figure 1 right plot in $1st$ line). When we choose to ignore the local optimality the performances of the algorithm are poor. It is clear, indeed, how higher values for this parameter lead to better convergences. The ρ parameter regulates the evaporation speed of the pheromone trails (figure 1, left plot in $2nd$ line). The results clearly show how a slow evaporation rate negatively affects the convergence speed of the algorithm. Therefore, a faster evaporation rate ($\rho = 0.99$) will force the algorithm to consider much more the best solutions. The parameters $minRec$ and $maxRec$ are computational bounds to the local search (right plot in $2nd$ line, and left plot in $3rd$ line of figure 1). A deeper local search will clearly lead to a faster convergence. However this does not seem to be the case of this instance, where different values of these parameters did not diversify the results. The last parameter, $partLim$, had an analogous behavior for this instance (right plot in $3rd$ line).

Regarding ABC-GCP, the control parameters are *popSize*, *percEmp* and *partLimit*, which are respectively the size of the population, its percentage of employed bees and

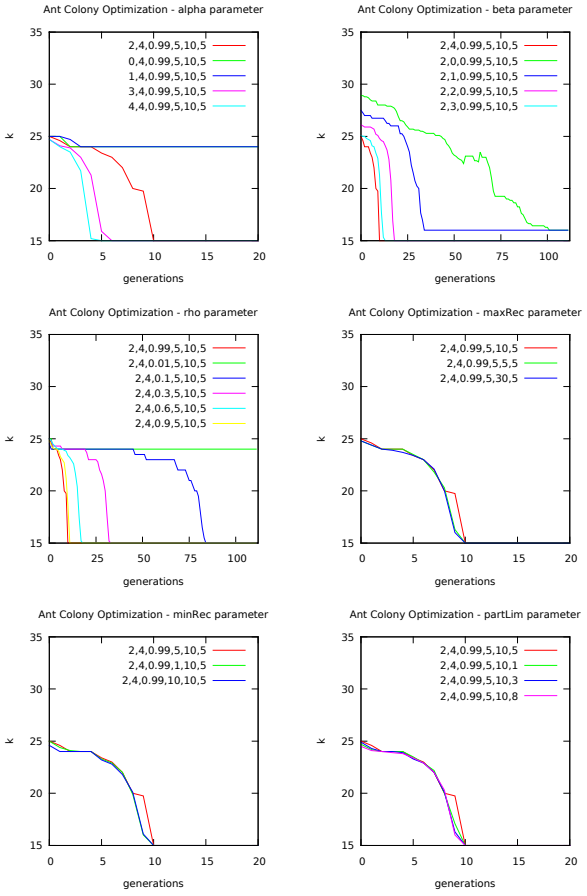

Fig. 1. An analysis of the convergence speed of different combinations of the parameters of the AS-GCP algorithm. Each result is averaged over 10 runs. The range of the values for the six parameters ρ, α, β, *recMin*, *recMax* and *partLim* are respectively $\{0.01, 0.1, 0.3, 0.6, 0.9, 0.99\}$, $\{0, 1, 2, 3, 4\}$, $\{0, 1, 2, 3, 4\}$, $\{1, 5, 10, 30\}$, $\{5, 10, 30\}$ and $\{1, 3, 5, 8\}$

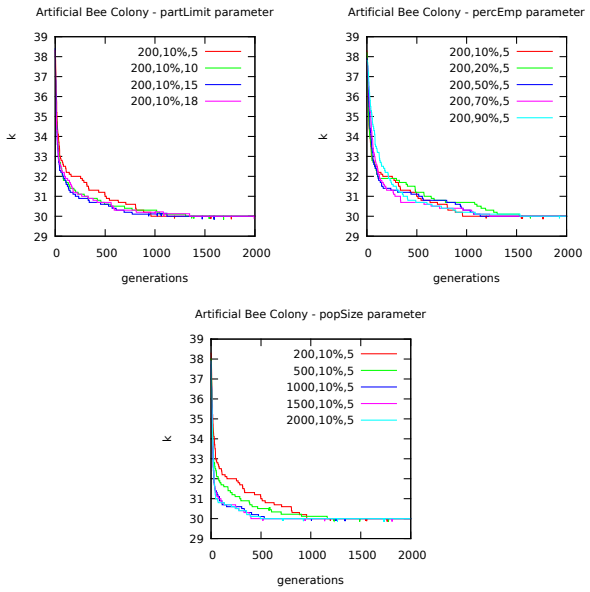

Fig. 2. An analysis of the convergence speed of different combinations of the parameters of the ABC-GCP algorithm. Each result is averaged over 10 runs. The range of the values for the three parameters $popSize$, $percEmp$ and $partLimit$ are respectively $\{200, 500, 1000, 1500, 2000\}$, $\{10\%, 20\%, 50\%, 70\%, 90\%\}$ and $\{5, 10, 15, 18\}$

the lowest cardinality of the color classes allowed to be transmitted during the partitioning phase. For this study the experiments were conducted on the instance *DSJC250.5*, and for each experiment 10 independent runs were made. The range of the values taken into account for the three parameters are respectively: $popSize \in \{200,\ 500,\ 1000,\ 1500,\ 2000\}$, $percEmp \in \{10\%,\ 20\%,\ 50\%,\ 70\%,\ 90\%\}$ and $partLimit \in \{5,\ 10,\ 15,\ 18\}$. In order to identifying the best combination of three parameters, we analyzed the performance of the algorithm when each of the control parameters varies within a certain range (figure 2). A low number of solutions apparently slow the convergence speed of the algorithm. However, the strong competition among the solutions guarantees that only the best ones will transmit their colorclass. Thus, after 1000 generations, the initial gap is filled. On all graphs tested, the best result has been obtained when these parameters assumed values in the range $(200, 2000)$. A higher number of onlooker bees clearly boost the convergence speed of the algorithm, yet reducing its variety. This percentage ranged within the values $(10\%, 90\%)$. This result is by all means interesting especially if we compare how common ABC algorithms use an even number of employed and onlooker bees. Finally, the lower bound to the color classes transmitted during the partitioning phase, which usually assumes values within the range $(2, \frac{|V|}{\chi})$ is the one which contributes most to the convergence speed of the algorithm. In conclusion, after several experiments made also on other instances the best values for the three parameters seems to be $\{200, 10\%, 5\}$.

Once the best tuning of parameters was determined for both swarm heuristics, we also studied the running time of AS-GCP and ABC-GCP. To do that, we have considered the *Time-To-Target* plots [23], which are a standard graphical methodology for data analysis [25], comparing the empirical and theoretical distributions. They represent a tool for characterizing the running time of stochastic algorithms in order to solve a specific optimization problem. In particular, for this study we have used a Perl program [24], which display the probability that an algorithm will find a solution as good as a target within a given running time. In order to perform such task, we make use of the cpu-time of 200 independent runs for both ABC-GCP and AS-GCP. For the two algorithms, the analysis was conducted respectively on the instances *DSJC250.1* and *le450_15b*. This tool produces two different plots: a quantile-quantile plot with execution times and probabilities, and the comparison of the derived theoretical distribution and the empirical distribution. The plots are shown in figure 3. The figures show how for the ABC-GCP algorithm the empirical curve perfectly fits the theoretical one except for very few worst cases (left plots). However, in the quantile-quantile plot, the algorithm shows how its results are in most of the cases equal or better than

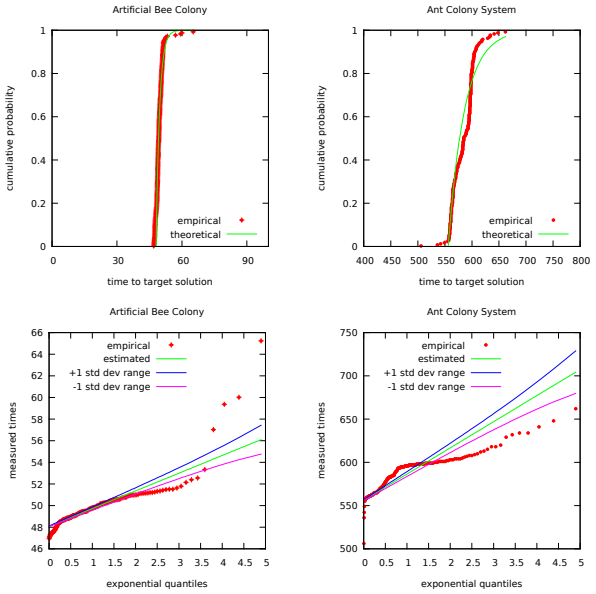

Fig. 3. Time to target plots for the presented algorithms. The values have been obtained over 200 hundred executions of the algorithms, respectively on the instance *le450_15b* for AS-GCP and *School1* for ABC-GCP

the theoretical ones. The algorithm AS-GCP, instead, has a different behavior (right plots). In most of the cases the empirical curve grows equally or worse than the theoretical one. However, in the worst cases, the empirical curve is much faster than the theoretical one. Comparing these ttt-plots is possible to see how ABC-GCP, despite its young age and poor experience, presents a behavior very closer, almost corresponding, with the estimated ones, and thus producing good approximations of the theoretical distributions than AS-GCP.

AS-GCP and ABC-GCP were compared using as test beds the classical DIMACS challenge benchmark [16] in order to assess their efficiency and robustness in finding the best coloring. The relative results are showed in table II where we report the number of vertices (V) and edges (E) for each instance tested; the chromatic number or the best coloring known (k); and for each algorithm we report also the best coloring found (bf), the success rate (SR) and average number of evaluations (AES) to reach the best coloring. For *queen* and *school* instances the results were averaged over 100 independent runs, whilst for the remaining ones the results were averaged over 10 runs. The termination criteria was set or in finding the optimal coloring (or the best known), or a maximum number of fitness function evaluations. Inspecting the preliminary experiments (not included) is already possible to claim that AS-GCP has a fast learning with respect to ABC-GCP that, instead, needs more generations. Several preliminary experiments performed on AS-GCP proved us how after a quick convergence, it has a really slow learning rate. The size of the ant colony was set to 100 for all the instances where $|V| < 100$, and to $|V|$ otherwise. Costa and Hertz in [11] have proved how the number of ants has a poor influence on the behavior of the

TABLE II
COMPARISON OF THE RESULTS OF THE TWO ALGORITHMS ON SEVERAL DIMACS INSTANCES.

				ABC-GCP			AS-GCP		
instance	$\|V\|$	$\|E\|$	k	bf	SR	AES	bf	SR	AES
Queen6_6	36	580	7	7	100%	1.7K	7	100%	**96**
Queen7_7	49	952	7	7	100%	6.6K	7	100%	**700**
Queen8_8	64	1,456	9	9	100%	22.1K	9	100%	**203**
Queen8_12	96	2,736	12	12	100%	1.2M	12	100%	**162**
Queen9_9	81	1,056	10	10	100%	31.2K	10	100%	**1K**
School1.nsh	352	14,612	14	14	100%	1.7K	14	100%	**1.4K**
School1	385	19,095	14	14	100%	**821,5**	14	100%	922
DSJC125.1	125	736	5	5	50%	528K	5	**100%**	160
DSJC125.5	125	3,891	12	17	10%	464K	17	**100%**	4.5K
DSJC125.9	125	6,961	30	44	100%	29K	44	100%	**1.1K**
DSJC250.1	250	3,218	8	9	100%	252K	**8**	100%	21.7K
DSJC250.5	250	15,668	13	29	**100%**	471K	29	30%	24.5K
DSJC250.9	250	27,897	35	73	**90%**	24.4M	73	40%	7.5K
le450_15a	450	8,168	15	16	100%	17.6M	16	100%	**1.6K**
le450_15b	450	8,169	15	16	100%	6.1M	16	100%	**1K**
le450_15c	450	16,680	15	15	100%	5.9M	15	100%	**4.3K**
le450_15d	450	16,750	15	15	80%	18.6M	15	**100%**	8.6K
flat300_20	300	21,375	20	20	100%	**4800**	20	100%	6.3K
flat300_26	300	21,633	26	**26**	100%	72.9K	32	60%	10.7K
flat300_28	300	21,695	28	**31**	20%	5.6M	32	60%	9.1K

algorithm. Analyzing table II, the two algorithms perform equally well for all *queen* and *school* instances; in such instances they find the optimal coloring with a SR of 100%. As expected AS-GCP finds the best coloring with a smaller fitness function evaluation, except for *school1*. In the second group of instances, the performances are instead slightly different. We highlighted in bold style the best results. In the instance *dsjc125.1*, ABC-GCP manages to find the optimal solution in only 5 cases out of 10, while AS-GCP has a SR of 100%. The difference is higher for the instance *dsjc125.5*, where only in one case out of 10 the ABC-GCP algorithm is able to find a 17-coloring. For the instance *dsjc250.1*, ABC-GCP fails to find an 8-coloring, while AS-GCP is able to find it in all the runs. For the instance *dsjc250.5*, AS-GCP finds a coloring of 29 in the 30% of the cases, while ABC-GCP succeeds in each run. The SR of ABC-GCP is again higher than the one of AS-GCP for the instance *dsjc250.9*. The same performances are achieved also in the *le450_15* family, where AS-GCP reaches more quickly the best coloring; in addition, AS-GCP produces slightly higher SR than ABC-GCP for *le450_15d* instance. ABC-GCP and AS-GCP both achieve the chromatic number in *flat300_20*. In *flat300_26*, AS-GCP is not able to reach a lower coloring than 32, while ABC-GCP achieves the chromatic number of 26. Finally, in *flat300_28*, ABC-GCP reaches a 31-coloring in 2 cases out of 10, while AS-GCP does not manage to do better than 32. Inspecting in overall these results, is possible to conclude that: (1) AS-GCP and ABC-GCP show very closer performances in term of best coloring found, and SR values; (2) confirm us how AS-GCP shows a quickly learning with respect to ABC-GCP; (3) if we tackle graphs with a more dense, and complex topology, ABC-GCP produces better performances in term of SR values, and, sometime, of colors found; showing also a more robust behavior. A comparison with many other algorithms for GCP was carried out to test the performance of both swarm intelligence heuristics. The

TABLE III

AS-GCP AND ABC-GCP VERSUS 15 DIFFERENT ALGORITHMS AMONG DETERMINISTIC AND NATURE-INSPIRED.

instance	I_Greedy [17]	T_B&B [18]	DIST [19]	PAR [20]	T_GEN_1 [21]	IMMALG [14]	ANTCOL [11]	EVOLVE_AO [13]	HCA [12]	MACOL [15]	HPSO [30]	GPB [27]	VNS [28]	VSS [29]	HANTCOL [31]	ABC-GCP	AS-GCP
DSJC125.5	18.9	20.0	**17.0**	**17.0**	**17.0**	18.0	18.7	17.2	-	17.0	-	-	-	-	-	17.9	**17.0**
DSJC250.5	32.8	35.0	**28.0**	29.2	29.0	**28.0**	31.0	29.1	28.1	**28.0**	**28.0**	**28.0**	-	-	**28.0**	29.0	29.7
flat300_20	**20.0**	39.0	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	26.0	-	**20.0**	-	-	-	-	-	**20.0**	**20.0**
flat300_26	37.1	41.0	**26.0**	32.4	**26.0**	27.0	34.4	31.0	-	**26.0**	**26.0**	-	31	-	-	**26.0**	32.6
flat300_28	37.0	41.0	31.0	33.0	33.0	32.0	34.3	33.0	31.4	**29.0**	31	31	31	**29**	31	31.8	32.6
le450_15a	17.9	16.0	**15.0**	**15.0**	**15.0**	**15.0**	16.0	**15.0**	-	**15.0**	-	-	-	-	-	16.0	16.0
le450_15b	17.9	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	16.0	**15.0**	-	**15.0**	-	-	-	-	-	16.0	16.0
le450_15c	25.6	23.0	**15**	16.6	16.0	**15.0**	**15.0**	16.0	15.6	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**
le450_15d	25.8	23.0	**15.0**	16.8	16.0	16.0	**15.0**	19.0	-	**15.0**	**15.0**	-	**15.0**	**15.0**	-	15.2	**15.0**
mulsol.i.1	**49.0**	**49.0**	**49.0**	**49.0**	**49.0**	**49.0**	-	-	-	-	-	-	-	-	-	**49.0**	**49.0**
school1.nsh	14.1	26.0	20.0	**14.0**	**14.0**	15.0	-	-	-	**14.0**	-	-	-	-	-	**14.0**	**14.0**

aim of this comparison is to prove how the results of AS-GCP and ABC-GCP are competitive with them, and in some cases among the best ones. In table III we present these comparisons, showing for each algorithm the average of the best coloring found over 10 independent runs, except for the two algorithms, HCA [12] and MACOL [15], where the results were averaged only on 5 independent runs. We highlighted in boldface the best results. The algorithms considered for the comparisons are: I_GREEDY, an iterated version of the greedy procedure [17]; T_B&B, a branch-and-bound algorithm using tabu search [18]; DIST [19]; PAR, a different version of DIST using a parallel branch-and-bound approach [20]; T_GEN_1, an hybrid genetic algorithm with tabu search [21]; IMMALG [14] a clonal selection algorithm; ANTCOL [11]; EVOLVE_AO [13], HCA [12] and GPB [27] three evolutionary algorithms; MACOL [15] a memetic algorithm; HPSO [30] an hybrid discrete PSO; VNS a variable neighborhood search algorithm [28]; VSS a variable space search algorithm [29]; and finally HANTCOL an hybrid ant colony optimization [31]. If we compare ABC-GCP and AS-GCP with the first 5 algorithms shown in table III, is possible to see how ABC-GCP finds in 5 instances out of 11 the best coloring, while however in the other ones it can be classified as second best. AS-GCP, instead, obtains the best result in 6 cases out of 11. In two more instances, the algorithm finds the second best coloring. Anyway in none of the remaining instances AS-GCP obtains the worst result. Comparing the two proposed algorithms with IMMALG, one of the best nature-inspired algorithms for optimization tasks [26], is possible to claim that ABC-GCP is the most efficient among the three algorithms, since it outperforms IMMALG in 8 instances over 11, and obtaining worst solutions in only 3 instances. AS-GCP, instead, is able to obtain only in 3 instances a better coloring than IMMALG, producing in 5 instances a worst coloring. The comparison with ANTCOL and EVOLVE_AO can be done only on 9 instances. Both ABC-GCP and AS-GCP outperform on all 9 instances the solutions obtained by ANTCOL, except for ABC-GCP on *le450_15d*, where the difference is very close (-0.2). Inspecting the coloring produced with respect to EVOLVE_AO, which exploits the relationship between direct acyclic graphs and the chromatic number, is possible to see that ABC-GCP shows again better performances in the overall. It outperforms EVOLVE_AO in 6 instances over 9, unlike AS-GCP that wins the comparisons only in 5 instances. In the overall, ABC-GCP produces a good assignment of colors in 5 out of 9 instances than both AS-GCP and EVOLVE_AO. HCA and MACOL are two hybrid algorithms that integrate a local search, specifically a tabu search, in a canonical evolutionary algorithm, whilst GPB is a similar algorithm to HCA but without the using of tabu search. Nowadays, HCA and MACOL are two of the best algorithms for GCP, therefore during the comparisons we don't expect to perform as well as or better than them. Anyway, is possible to see how far our algorithms are from them by computing the difference of results. Inspecting these last comparisons is important to keep in mind that only one of our algorithms, namely AS-GCP, makes use of a local search operator, while ABC-GCP only uses perturbation operators. The results show how clearly HCA, MACOL and GPB outperform the proposed algorithms; however, both ABC-GCP and AS-GCP manage to do better than HCA in the instance *le450_15c*, having a higher SR. About the last four algorithms is important to point out that the results have been taken in [30] where only the best coloring found is showed. Therefore, we analyze the comparisons done taking into account only the best coloring found by AS-GCP and ABC-GCP. Thus, inspecting the table with respect the best coloring found is possible to see how ABC-GCP is comparable with all algorithms. In fact, the coloring produced by ABC-GCP is similar in all instances, except in DSJC250.5 where it finds a coloring with a color more (28 vs. 29). AS-GCP, instead, fails to get the same colors found by compared algorithms, showing then slightly worst performances. Analyzing in overall all comparisons performed is evident the good outcome of both swarm heuristics, showing efficient and robust performances. Albeit AS-GCP presents a quickly convergence in reaching the best coloring, inspecting these last results in table III, instead, emerges how ABC-GCP shows more robust performances than AS-GCP, resulting more comparable with the many tested algorithms.

IV. CONCLUSIONS

A comparative study is presented between two of the well known swarm intelligence methodologies, as Ant Colony Systems and Artificial Bee Colony, called respectively AS-GCP and ABC-GCP. The AS-GCP designed is based on

the combination of GPX (Greedy Partitioning Crossover) and a local search that interact with the pheromone trails system; while ABC-GCP has in the mutation operator; in the improved version of GPX and in a Temperature mechanism its strengths. The aim of this study is to compare the two algorithms in term of efficiency and robustness in solving one of the most popular combinatorial optimization problems that is the Graph Coloring Problem (GCP). As first step, for both algorithms was studied the best tuning of the parameters from which depend the algorithms. About that, in ABC-GCP a smaller number of parameters are involved (3) than AS-GCP (6). Once obtained the best tuning for each swarm heuristic an analysis on their running time was also made using the *Time-To-Target* plots, an useful tool able to display the probability that a generic stochastic algorithm finds the optimal solution on n independent runs within a given running time. Inspecting such plots, one can see that the bees-based algorithm produces good approximations of the theoretical distributions than AS-GCP. Hence an experimental comparison was performed both between the two algorithms, and with 15 different algorithms for GCP. For these experiments, we used the classical DIMACS benchmark. Analyzing the experiments performed, and inspecting all comparisons done is possible to claim that both swarm heuristics are competitive with all tested algorithms in term of coloring found, ranking them among the best ones in many instances. This proves us that the variants, and novelties designed are very helpful in order to improve the performances in both algorithms. Moreover, reducing the analysis only on the comparison between AS-GCP and ABC-GCP is possible to conclude that AS-GCP shows a quickly learning, which helps the ants-based algorithm in reaching the best coloring in a very few generations; whilst ABC-GCP presents a behavior more robust than AS-GCP, and in overall more competitive with the compared algorithms. Finally, if we tackle graphs with a more dense, and complex topology, then bees-based heuristic is able to find best colors.

REFERENCES

[1] Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 26(1): 29–41, (1996)

[2] Kennedy J., Eberhart, R.: Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, 1942–1948, (1995).

[3] Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3): 459–471, (2007).

[4] Hernandez, H., Blum, C.: Distributed Graph Coloring: An Approach Based on the Calling Behavior of Japanese Tree Frogs. Swarm Intelligence, 6(2): 117–150, (2012).

[5] Hernandez, H., Blum, C.: FrogSim: Distributed Graph Coloring in Wireless Ad Hoc Networks – An Algorithm Inspired by the Calling Behavior of Japanese Tree Frogs. Telecommunication Systems. In press (2013).

[6] Zhang, W., Wang, G., Xing, Z., Wittenburg, L.: Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. Artificial Intelligence, 161(1-2): 55– 87, (2005).

[7] Glass, C.: Bag rationalization for a food manufacturer. Journal of the Operational Research Society, (53): 544–551, (2002).

[8] Gamst, A.: Some Lower Bounds for a Class of Frequency Assignment Problems. IEEE Transaction on Vehicular Technology, (35): 8–14, (1986).

[9] Chow, F. C., Hennessy, J. L.: The priority-based Coloring Approach to Register Allocation. ACM Transaction on Programming Languages and Systems, (12): 501–536, (1990).

[10] Garey, M. R., Johnson, D. S.: Computers and Intractability: a Guide to the Theory of NP-completeness. Freeman, New York, (1979).

[11] Costa, D., Hertz, A.: Ants can colour graphs. Journal of the Operational Research Society, 48: 295–305, (1997).

[12] Galinier, P., Hao, J.: Hybrid Evolutionary Algorithm for Graph Coloring. Journal of Combinatorial Optimization, 3(4): 379–397, (1999)

[13] Barbosa, V., Assis, C., do Nascimento, J.: Two Novel Evolutionary Formulations of the Graph Coloring Problem. Journal of Combinatorial Optimization, 8(1): 41–63, (2004).

[14] Cutello, V., Nicosia, G., Pavone, M.: An immune algorithm with stochastic aging and kullback entropy for the chromatic number problem. Journal of Combinatorial Optimization, 14(1): 9–33 (2007)

[15] Lü, Z., Hao, J.: A memetic algorithm for graph coloring. European Journal of Operational Research, 203(1): 241–250, (2010)

[16] Johnson, D., Trick, M.: Cliques, coloring and satisfability: second DIMACS implementation challenge. Providence, R.I.: American Mathematical Society, (1996)

[17] Culberson, J., Luo, F.: Exploring the k-colorable Landscape with Iterated Greedy. Cliques, coloring and satisfiability: second DIMACS implementation challenge, 14(1): 254–284, (1996).

[18] Glover, F., Parker, M., Ryan, J.: Coloring by Tabu Branch and Bound. Cliques, coloring and satisfiability: second DIMACS implementation challenge,14(1): 285–307, (1996)

[19] Morgenstern, C.: Distributed coloration neighborhood search. Cliques, coloring and satisfiability: second DIMACS implementation challenge, 14(1): 335–357, (1996)

[20] Lewandowski, G., Condon A.: Experiments with Parallel Graph Coloring Heuristics and Applications of Graph Coloring. Cliques, coloring and satisfiability: second DIMACS implementation challenge, 14(1): 309–334, (1996)

[21] Fleurent, C., Ferland, J.: Object-oriented implementation of heuristic search methods for graph coloring, maximum clique and satisfability. Cliques, coloring and satisfiability: second DIMACS implementation challenge, 14(1): 619–652, (1996)

[22] Leighton, F.: A graph coloring algorithm for large scheduling problems. Journal of research of the National Bureau of Standards, 84(6), (1979)

[23] Aiex, R. M., Resende, M. G. C., Ribeiro, C. C.: Probability distribution of solution time in GRASP: an experimental investigation. Journal of Heuristics, 8: 343–373 (2002)

[24] Aiex, R. M., Resende, M. G. C., Ribeiro, C. C.: TTTPLOTS: A perl program to create time-to-target plots. Optimization Letters, 1: 355–366 (2007)

[25] Chambers, J. M., Cleveland, W. S., Kleiner, B., Tukey, P. A.: Graphical Models for Data Analysis. Chapman & Hall, London (1983)

[26] Pavone, M., Narzisi, G., Nicosia, G.: Clonal Selection – An Immunological Algorithm for Global Optimization over Continuous Spaces. Journal of Global Optimization, 53(4): 769–808 (2012).

[27] Glass, C. A., Prügel-Bennet, A.: Genetic Algorithm for Graph Coloring: Exploration of Galinier and Hao's Algorithm. Journal of Combinatorial Optimization, 7(3): 229–236 (2003).

[28] Avanthay, C., Hertz, A., Zufferey, N.: A Variable Neighborhood Search for Graph Coloring. European Journal of Operational Research, 151(2): 379–388 (2003).

[29] Hertz,A., Plumettaz, M., Zufferey, N.: Variable Space Search for Graph Coloring. Discrete Applied Mathematics, 156: 2551–2560 (2008).

[30] Qin, J., Yin, Y., Ban, X-J.: Hybrid Discrete Particle Swarm Algorithm for Graph Coloring Problem. Journal of Computers, 6(6): 1175–1182 (2011).

[31] Dowsland, K. A., Thompson, J. M.: An Improved Ant Colony Optimisation Heuristic for Graph Colouring. Discrete Applied Mathematics, 156(3): 313–324 (2008).