A Hybrid Artificial Bee Colony Algorithm
for Graph 3-Coloring

Iztok Fister Jr.,[*] Iztok Fister,[†] and Janez Brest[‡]

Abstract

The Artificial Bee Colony (ABC) is the name of an optimization algorithm that was inspired by the intelligent behavior of a honey bee swarm. It is widely recognized as a quick, reliable, and efficient methods for solving optimization problems. This paper proposes a hybrid ABC (HABC) algorithm for graph 3-coloring, which is a well-known discrete optimization problem. The results of HABC are compared with results of the well-known graph coloring algorithms of today, i.e. the Tabucol and Hybrid Evolutionary algorithm (HEA) and results of the traditional evolutionary algorithm with SAW method (EA-SAW). Extensive experimentations has shown that the HABC matched the competitive results of the best graph coloring algorithms, and did better than the traditional heuristics EA-SAW when solving equi-partite, flat, and random generated medium-sized graphs.

---

[*]University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: iztok.fister@guest.arnes.si

[†]University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: iztok.fister@uni-mb.si

[‡]University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: janez.brest@uni-mb.si

arXiv:1206.1012v1 [cs.NE] 6 May 2012

I. INTRODUCTION

Graph coloring represents a test bed for many newly developed algorithms because of its simple definition, which states: How to color a graph G with the k colors, so that none of the vertices connected with an edge have the same color. The coloring c is *proper* if no two connected vertices are assigned to the same color. A graph is *k-colorable* if it has a proper *k-coloring*. The minimum k for which a graph G is *k-colorable* is called its *chromatic number* $\chi(G)$.

Many approaches for solving the graph coloring problem (GCP) have been proposed over the time [12, 20]. The most natural way to solve this problem is, however, in a greedy fashion, where the vertices of the graph are ordered into a permutation, and colored sequential. Thus, the quality of coloring depends on the permutation of the vertices. For example, the DSatur algorithm [3], one of the best traditional heuristics for graph coloring today, orders the vertices v according to *saturation degrees* $\rho(v)$. The saturation degree represents the number of distinctly colored vertices adjacent to the vertex v. Furthermore, DSatur's ordering is calculated dynamically during the coloring process.

Many heuristic methods have been developed for larger instances [12] because exact algorithms can only color instances of up to 100 vertices. These methods can be divided into local search methods [1] and hybrid algorithms [19]. The most important representative of the former is Tabucol [15], which utilizes the tabu search, as proposed by Glover [14]. Later were combined local search methods with evolutionary algorithms and improved the results of pure Tabucol, as for example, the hybrid genetic algorithm by Fleurent and Ferland [10], and the hybrid evolutionary algorithm (HEA) by Galinier and Hao [11].

Swarm intelligence is the collective behavior of a self-organized system. Birds, insects, ants, and fish use collective behavior for foraging and defending. These individuals are looking for good food sources and help each other when a lack of food has arisen. This concept was introduced into the computer's world by Kennedy and Eberhart [18]. Moreover, it was successfully applied to several problem domains, for example, particle swarm optimization, which achieves good results during antenna optimization [24]. In addition, ant colony optimization reaches good results by solving the traveling-salesman person [6]. Finally, the artificial bee colony algorithm, proposed by Karaboga and Basturk [17], exhibited excellent results when solving combinatorial optimization problems [21, 25].

This paper focuses on the artificial bee colony (ABC) algorithm for graph 3-coloring (3-GCP), which belongs to a class of NP-complete [13] problems. There, the real-valued weights w are assigned to the vertices v. These weights determine how difficult the vertex is to color. The higher the weight, the earlier the vertex should be colored. Thus, weights define the order in which the vertices should be colored. This ordering is used by the DSatur traditional algorithm for constructing 3-coloring. The ABC algorithm incorporates DSatur as a decoder. In this manner, the ABC algorithm acts as a meta-heuristic concerned for a generation of new solutions (vector of weights), whilst the quality of the solution (its fitness) is evaluated by DSatur. This approach is not new: it was used by the evolutionary algorithm with SAW method (EA-SAW) of Eiben et al. [8], and by the hybrid self-adaptive differential evolution of Fister et al. [9]. In the former case, instead of Dsatur, a greedy heuristic was applied as a decoder. Finally, the proposed ABC algorithm was hybridized with a *random walk with direction exloitation* (RWDE) [23] local search heuristic. This local search heuristic was applied in place of the original sending scouts function and focuses itself on discovering new food sources in the vicinity of the current sources.

The results of the proposed hybrid artificial bee colony algorithm for graph 3-coloring (HABC) was compared with the results obtained with Tabucol, HEA, and EA-SAW for solving an extensive set of random medium-scale graphs generated by the Culberson graph generator [5]. A comparison between these algorithms shows that the results of the proposed HABC algorithm are comparable with results of the other algorithms used in the experiments.

The structure of this paper is as follows: In Section 2, the 3-GCP is discussed, in detail. The HABC is described in Section 3, whilst the experiments and results are presented in Section 4. The paper is concluded with a discussion about the quality of the results, and directions for further work are outlined.

II. GRAPH 3-COLORING

3-*coloring* of a graph $G = (V, E)$ is a mapping $c : V \to C$, where $C = \{1, 2, 3\}$ is a set of three colors [2]. Note that V in the graph definition denotes a set of vertices $v \in V$ and E a set of edges that associates each edge $e \in E$ to an unordered pair of vertices (v_i, v_j) for $i = 1 \ldots n \wedge j = 1 \ldots n$.

3

3-GCP can be formally defined as a constraint satisfaction problem (CSP) that is represented as the pair $\langle S, \phi \rangle$, where $S = C^n$ with $C^n = \{1, 2, 3\}$ denotes the free search space, in which all solutions $c \in C^n$ are feasible and ϕ a Boolean function on S (also a feasibility condition) that divides search space into feasible and unfeasible regions. This function is composed of constraints belonging to edges. In fact, to each $e \in E$ the corresponding constraint b_e is assigned by $b_e(\langle c_1, \ldots, c_n \rangle) = true$ if and only if $e = (v_i, v_j)$ and $c_i \neq c_j$. Assume that $B^i = \{b_e | e = (v_i, v_j) \wedge j = 1 \ldots m\}$ defines the set of constraints belonging to variable v_i. Then, the feasibility condition ϕ is expressed as a conjunction of all the constraints $\phi(c) = \wedge_{v \in V} B^v(c)$.

Typically, constraints are handled indirectly in the sense of the penalty function that transforms the CSP into free optimization problem (FOP) [7] (also unconstrained problem). Thus, those infeasible solutions that are far away from a feasible region are punished by higher penalties. The penalty function that is also used as a fitness function here, is expressed as:

$$f(c) = min \sum_{i=0}^{n} \psi(c, B^i), \qquad (1)$$

where the function $\psi(c, B^i)$ is defined as:

$$\psi(c, B^i) = \begin{cases} 1 & \text{if } c \text{ violates at least one } b \in B^i, \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

In fact, Eq. (1) can be used as a feasibility condition in the sense that $\phi(c) = true$ if and only if $f(c) = 0$. Note that this equation evaluates the number of constraint violations and determines the quality of solution $c \in C^n$.

III. HABC FOR GRAPH 3-COLORING

In the ABC algorithm, the colony of artificial bees consists of three groups [27]: employed bees, onlookers, and scouts. The employed bees discover each food source, that is, only one employed bee exists for each food source. The employed bees share information about food sources with onlooker bees, in their hive. Then, the onlooker bees can choose a food sources to forage. Interestingly, those employed bees whose food source is exhausted either by

employed or onlooker bees, becomes scouts. The ABC algorithm is formally described in Algorithm 1, from which it can be seen that each cycle of the ABC search process (statements within a **while** loop) consists of three functions:

- SendEmployedBees(): sending the employed bees onto the food sources and evaluating their nectar amounts,

- SendOnlookerBees(): sharing the information about food sources with employed bees, selecting the proper food source and evaluating their nectar amounts,

- SendScouts(): determining the scout bees and then sending them onto possibly new food sources.

Algorithm 1 Pseudo code of the ABC algorithm

1: Init();
2: **while** !TerminationConditionMeet() **do**
3: SendEmployedBees();
4: SendOnlookerBees();
5: SendScouts();
6: **end while**

However, before this search process can take place, initialization is performed (function Init()). A termination condition (function TerminationConditionMeet()) is responsible for stoping the search cycle. Typically, the maximum number of function evaluations MAX_FES is used as the termination condition.

The ABC algorithm belongs to population-based algorithms, where the solution of an optimization problem is represented by a food source. The solution of 3-GCP is represented as a real-valued vector $Y_i = \{w_{ij}\}$ for $i = 1...NP \wedge j = 1...n$, where w_{ij} denotes the weight associated with the j-th vertex of the i-th solution; NP is the number of solutions within the population, and n the number of vertices. The values of the weights are taken from the interval $w_{ij} \in [lb, ub]$, where lb indicates the lower, and ub the upper bounds. The initial values of the food sources are generated randomly, according to the equation:

$$w_{ij} = \Phi_{ij} \cdot (ub - lb) + lb, \tag{3}$$

where the function Φ_{ij} denotes the random value from the interval $[-1, 1]$.

The employed and onlooker bees change their food positions within the search space, according to the equation:

$$w'_{ij} = w_{ij} + \Phi_{ij}(w_{ij} - w_{kj}),\qquad(4)$$

where Φ_{ij} is a random number from interval $[-1, 1]$. The onlooker bee selects a food source with regard to the probability value associated with that food source p_i calculated by the equation:

$$p_i = \frac{f(\Gamma(Y_i))}{\sum_{j=0}^{NP} f(\Gamma(Y_j))},\qquad(5)$$

where Γ indicates a mapping from the real-valued search space to the problem search space, as explained in the next subsection, and f the fitness function according to Eq. (1).

A. Fitness calculation

The ABC for 3-GCP explores continuous real-valued search space, where the solution is represented as $Y_i = \{w_{ij}\}$ for $i = 1...NP \wedge j = 1...n$. Firstly, this solution needs to be transformed into a permutation of vertices $X_i = \{v_{ij}\}$. Such a permutation can be decoded into 3-coloring $C_i = \{c_{ij}\}$ by the DSatur heuristic. The 3-coloring C_i represents the solution of 3-GCP in its original problem space. Whilst a new position regarding a food source is performed within the real-valued search space, its quality is evaluated within the original problem space, according to the equation Eq.(1). This relation can be expressed mathematically as follows:

$$X_i = \Gamma(Y_i), \qquad \text{for } i = 1...NP.\qquad(6)$$

Note that the function Γ is not injective, i.e. more than one food source can be mapped into the same value of the fitness function. On the other hand, a weakness of this function is that a small move in the real-valued search space can cause a significant increase or decrease in the fitness function.

B. Hybridization with local search

In the classical ABC algorithm, scouts act as a random selection process. That is, if the position of a food source cannot be improved further within a predetermined number of cycles called *limit*, then that source is replaced by the randomly generated position. In HABC, instead of randomly generating the new position in the search space (exploration), a deterministic exploitation in the vicinity of the current solution was used [16]. Thus, in place of the original SendScouts() function, the RWDE local search heuristic was implemented, which generates the new food sources according to the following equation [23]:

$$Y_i' = Y_i + \lambda \cdot U_i, \tag{7}$$

where λ is the prescribed scalar step size length and U_i is a unit random vector generated for the i-th solution.

IV.   EXPERIMENTS AND RESULTS

The goal of the experimental work was to show that HABC can be successfully applied to 3-GCP. In line with this, the proposed HABC was compared with: EA-SAW, Tabucol, and HEA, whose implementations were downloaded from the Internet.

The characteristics of the HABC used during the experiments were as follows: The population size was set at 100 because this value represents a good selection, as was indicated during the experimental work. The value of *limit* was set at 1,000, whilst the MAX_FES was limited to 300,000. The former value was obtained through experimental work, whilst the later was selected in order to draw a fair comparison with the other algorithms, i.e. the other algorithms also obey the same limitation. In the end, 25 independent runs were observed, because of the stochastic nature of observed algorithms.

The algorithms were compared according to two measures: *success rate* (SR) and *average number of objective function evaluations to solution* (AES). The first measure expresses the ratio of the number of successful runs from among all runs, whilst the second reflects the efficiency of a particular algorithm.

A. Test suite

All graphs in the test suite were generated using the Culberson random graph generator [5], which allows to generate graphs of different: size n, type t, edge probability p, and seed s. This paper focuses on medium-sized graphs, i.e. graphs with 500 vertices. Three types of graphs were used as follows: uniform (a random graph with variability set at zero), equi-partite, and flat graphs. The edge probability was varied from 0.008 to 0.028 with steps of 0.001. Thus, 21 instances of randomly generated graphs were obtained. Ten different seeds were employed, i.e. from 1 to 10. In summary, each algorithm was solved $3 \times 21 \times 10 \times 25 = 15,750$ different instances of graphs.

An interval of edge probabilities was selected such that the region of *phase transition* was included. Phase transition is a phenomenon that is connected with most combinatorial optimization problems and indicates those regions, where the problem passes over the state of "solvable" to the state of "unsolvable", and vice versa [26]. The 3-GCP determination of this phenomenon is connected with parameter edge probability. Interestingly, this region is identified differently by many authors. For example, Petford and Welsh [22] stated that this phenomenon occurs when $2pn/3 \approx 16/3$, Cheeseman et al. [4] when $2m/n \approx 5.4$, and Eiben et al. [8] when $7/n \leq p \leq 8/n$. In our case, the phase transition needed to be by $p = 0.016$ over Petford and Welsh, by $p \approx 0.016$ over Cheeseman, and between $0.014 \leq p \leq 0.016$ over Eiben et al..
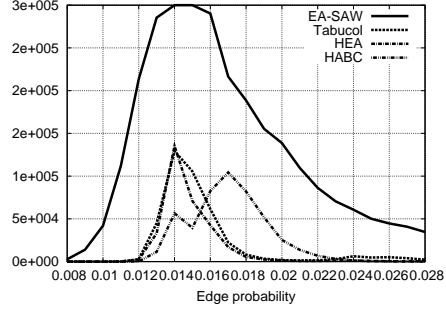
B. Influence of edge probability

In this experiment, the phenomenon of phase transition was investigated, as illustrated by Fig. 1. The figure is divided into six graphs according to type, and two different measures SR and AES. The graphs capture the results of 21 instances that were obtained by varying the edge probability through a region, including phase transition. Due to space limitation of this paper's length, a more detailed analysis of the results is left to the reader.
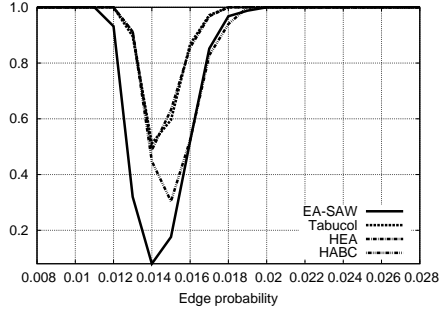
In summary, the best results on medium-sized graphs were reported by HEA and Tabucol. The results of HABC were slightly worse but comparable to both of the mentioned algorithms, whilst the EA-SAW saw the worst results.
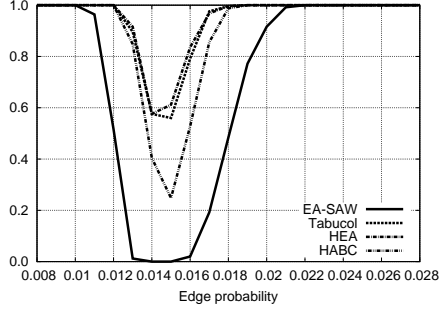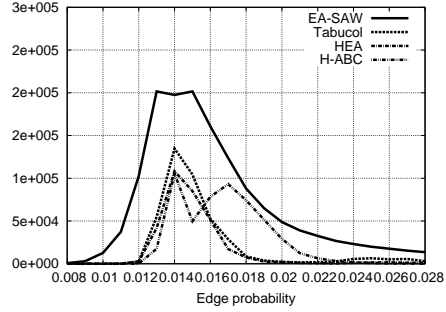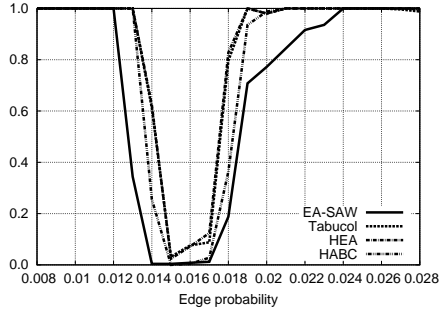
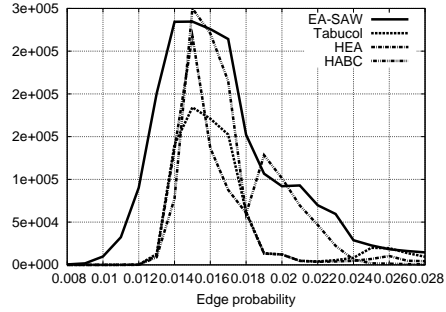(a)SR by uniform graphs   (b)AES by uniform graphs

(c)SR by equi-partite graphs   (d)AES by equi-partite graphs

(e)SR by flat graphs   (f)AES by flat graphs

FIG. 1: Results of algorithms for 3-GCP solving different types of random graphs

C. Influence of the local search

During this experiment, the influence of hybridizing the ABC algorithm with a RWDE local search heuristic was observed. Therefore, a especial focus was placed on the instances during phase transition, i.e. $p \in [0.013, 0.017]$. The two versions of ABC were compared: the original and the hybridized version. In the former, the scouts were generated randomly, whilst in the later the RWDE local search heuristic was used.

The results of this experiment are shown in Table I, where the row *Graphs* indicates different graph types, whilst the columns *Random* and *Hybrid* indicate the original and the

hybrid ABC algorithms. Note that the average results of the mentioned instances have a varying seed $s \in [1, 10]$ and are presented in the table.

TABLE I: Influence of the local search by HABC on different graph types.

Graphs	Uniform		Equi-partite		Flat	
p	Random	Hybrid	Random	Hybrid	Random	Hybrid
0.013	0.816	0.848	0.872	0.912	1.000	1.000
0.014	0.112	0.404	0.200	0.448	0.012	0.256
0.015	0.060	0.248	0.036	0.304	0.000	0.000
0.016	0.180	0.528	0.104	0.524	0.000	0.004
0.017	0.328	0.856	0.340	0.828	0.000	0.028
avg	0.299	0.577	0.310	0.603	0.202	0.258

The results showed that using the RWDE local search, substantially improved the results of the original ABC. For example, this improvement amounted to 92.98% for uniform, 94.52% for equi-partite, and 27.72% for flat graphs. On average, hybridization improved the results of the original ABC for 71.74%.

V.   CONCLUSION

The results of the proposed HABC for 3-GCP convinced us that the original ABC algorithm is a powerful tool for solving combinatorial optimization problems. HABC gained results that are comparable with the results of the best algorithm for k-GCP today (Tabucol and HEA), and improved results obtained with EA-SAW when solving the medium-sized extensive suite of random generated graphs. Note that these graphs are not the hardest to color but are difficult enough that the suitability of the ABC technology for solving the 3-GCP could be successfully proven.

In the future, the HABC for 3-GCP could be additionally improved. In particular, the problem-specific knowledge via local search heuristics could be conducted into the algorithm. The greatest challenge for further work remains the solving of large-scale graph suite (graphs with 1,000 vertices). We are convinced that these graphs could also be successfully solved using the proposed HABC algorithm.

[1] Aarts E. and Lenstra J.K.: Local Search in Combinatorial Optimization. Princeton University Press. Princeton and Oxford. (1997).

[2] Bondy J.A. and Murty U.S.R.: Graph Theory. Springer-Verlag. Berlin. (2008).

[3] Brelaz D.: New methods to color vertices of a graph. Communications of the ACM. 22(4): 251–256 (1979).

[4] Cheeseman P. and Kanefsky B. and Taylor W.M.: Where the really hard problems are. Proceedings of the International Joint Conference on Artificial Intelligence. Morgan Kaufmann. 1: 331–337 (1991).

[5] Culberson J.: Graph Coloring Page. `http://www.ncbi.nlm.nih.gov`.

[6] Dorigo M. and Stützle T.: Ant Colony Optimization. MIT Press. Cambridge. (2004).

[7] Eiben A.E. and Smith J.E.: Introduction to Evolutionary Computing. Springer-Verlag. Berlin. (2003).

[8] Eiben A.E. and Hauw K. and Hemert J.I.: Graph coloring with adaptive evolutionary algorithms. Journal of Heuristics. 4(1): 25–46 (1998).

[9] Fister I. and Brest J.: Using differential evolution for the graph coloring. Proceedings of IEEE SSCI2011 symposium series on computational intelligence. Piscataway. 150–156 (2011).

[10] Fleurent C. and Ferland J.: Genetic and Hybrid Algorithms for Graph Coloring. Annals of Operations Research. 63: 437–464 (1996).

[11] Galinier P. and Hao J.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization. 3(4): 379–397 (1999).

[12] Galinier P. and Hertz A.: A survey of local search methods for graph coloring. Computers & Operations Research. 33: 2547–2562 (2006).

[13] Garey M.R. and Johnson D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co. New York, NY, USA. (1979).

[14] Glover F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research. 13(5): 533-549 (1986).

[15] Hertz A. and de Werra D.: Using tabu search techniques for graph coloring. Computing. Springer-Verlag New York. 39(4): 345–351 (1987).

[16] Iacca G. and Neri F. and Mininno E. and Ong Y.-S. and Lim M.-H.: Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration. In press (2011).

[17] Karaboga D. and Basturk A.: A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review. 31(1–4): 61–85 (2009).

[18] Kennedy J. and Eberhart R.C.: Particle swarm optimization. In: Proceedings of the 1995

IEEE International Conference on Neural Networks. IEEE Service Center. Piscataway. 4: 1942–1948 (1995).

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

[20] Malaguti E. and Toth P.: A survey on vertex coloring problems. International Transactions in Operational Research. 1–34 (2009).

[21] Pan Q.K. and Tasgetiren M.F. and Suganthan P.N. and Chua T.J.: A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Information Sciences. 181(12): 2455–2468 (2011).

[22] Petford A.D. and Welsh D.J.A.: A randomized 3-coloring algorithms. Discrete Mathematic. 74(1–2): 253–261 (1989).

[23] Rao S.S.: Engineering Optimization: Theory and Practice. John Willey & Sons. New Jersey. (2009).

[24] Robinson J. and Rahmat-Samii Y.: Particle Swarm Optimization in Electro-Magnetics. IEEE on Antennas and Propagation. 52(2): 397–407 (2004).

[25] Tasgetiren M.F. and Pan Q.K. and Suganthan P.N. and Chen A.H.-L.: A discrete artificial bee colony algorithm for the total flowtime minimization and permutation flow shops. Information Sciences. 181(16): 3459–3475 (2011).

[26] Turner J.S.: Almost all k-colorable graphs are easy to color. Journal of Algorithms. 9(1): 63–82 (1988).

[27] Yang X.-S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press. Cambridge. (2008).

Updated 5 May 2012.