MODERN HYBRID COLORING ALGORITHM

João Fabrício Filho
Federal University of Technology - Parana, Campus Campo Mourão
Via Rosalina M. Santos, 1233 - Campo Mourão, PR - Brazil
joaof@utfpr.edu.br

Luis Gustavo Araujo Rodriguez
Department of Informatics - State University of Maringá
Colombo Avenue, 5790 - Block C56 - 87.020-900 - Maringá - PR - Brazil
luisgar1990@gmail.com

Anderson Faustino da Silva
Department of Informatics - State University of Maringá
Colombo Avenue, 5790 - Block C56 - 87.020-900 - Maringá - PR - Brazil
anderson@din.uem.br

ABSTRACT

In 1999, Philippe Galinier and Jin-Kao Hao developed the widely praised Hybrid Coloring Algorithm (HCA) to solve the Graph Coloring Problem (GCP), which is a complex computational problem that focuses in coloring all vertices of a given graph with a minimum number of colors, with the constraint that adjacent vertices can not receive the same color. Since then, powerful computer infrastructures have emerged, with considerable benefits to software developers and applications. Thus, the objective of this paper is to analyze in detail the current HCA and propose a modern version with improvements that take advantage of recent computer infrastructures. The expected contribution at the conclusion of this work is to encourage developers to take advantage of modern architectures to solve complex computational problems.

KEYWORDS. Hybrid Coloring Algorithm, Graph Coloring Problem, Modern Computer Architecture.

RESUMO

Em 1999, Philippe Galinier e Jin-Kao Hao desenvolveram o Algoritmo Híbrido de Coloração (HCA), o qual é amplamente elogiado para resolver o Problema de Coloração de Grafos (GCP). O GCP é um problema computacional complexo que concentra-se na coloração de todos os vértices de um determinado grafo com um número mínimo de cores, restringindo que os vértices adjacentes não recebam a mesma cor. Desde então, poderosas infra-estruturas informáticas têm surgido com benefícios consideráveis para desenvolvedores e aplicações. Assim, o objetivo deste trabalho é analisar em detalhe o HCA atual e propor uma versão moderna que aproveite todos os recursos computacionais disponíveis. A contribuição esperada na conclusão deste trabalho é promover as arquiteturas recentes de computadores para resolver problemas computacionais complexos.

PALAVRAS CHAVE. Algoritmo Híbrido de Coloração, Problema de Coloração de Grafo, Arquitetura Moderna de Computadores.

1. Introduction

Computer architectures have evolved tremendously over the last few years, with advancements such as multi-core processors, hyperthreading, or parallel programming emerging as huge benefits for software developers and applications. Recently, programmers have started to embed such technologies into their systems for the following reasons: (1) gain speed and efficiency, (2) take advantage of all hardware resources, (3) and reduce work-load.

Several types of algorithms have improved considerably due to utilizing modern computer infrastructures, such as *Metaheuristics*, which its goal is to search for good solutions in a given problem. One problem in particular, regarded important in Computer Science as well as in other areas, is *Graph Coloring*, which consists in coloring vertices with a minimum number of colors in a given graph, restricting adjacent vertices from receiving the same color.

Various projects have been proposed and implemented to solve the Graph Coloring Problem (GCP) (Abbasian e Mouhoub [2013]; Becirspahic et al. [2013]; Dabrowski e Kubale [2008]; Dawson e Stewart [2013]; Eblen et al. [2012]; Evstigneev e Tursunbay kyzy [2011]; Lukasik et al. [2008]). However, such works have not focused in improving or updating the Hybrid Coloring Algorithm (HCA), which is widely considered to be one of the most successful and finest algorithms to solve such problem, to modern standards.

Thus, the purpose of this paper is to analyze in detail the HCA and propose a modern version to explore architectural advances, thereby solving the GCP more efficiently and rapidly, which the former (efficiently) relates to the amount of utilized resources and the latter (rapidly) refers to the response time of the proposed algorithm.

The specific objectives of this work are the following: (1) analyze in detail the current HCA in order to determine areas of improvement, (2) propose and implement a modern version that takes advantage of all hardware resources, (3) and highlight the importance of updating classic algorithms to modern standards for solving combinatorial optimizations.

The model that was implemented to update the HCA is called the *Island Model*, which basically consists in sub-populations with capabilities to exchange individuals, and thus, improve convergence (Kokosiński et al. [2004]).

The graph coloring instances utilized to evaluate both versions of the HCA belong to the DIMACS benchmark, which is widely used for the GCP (Alahmadi et al. [2014]; Consoli et al. [2013]; Djelloul et al. [2014]; Lintzmayer et al. [2011]; Myszkowski [2005]; Salari e Eshghi [2005]; Yesil et al. [2011]).

The results obtained in the evaluations prove that the proposed version of the HCA is an improvement over the original, due to achieving values more proximate to the best outcomes in the literature, as well as reducing work-load.

2. Related Works

Metaheuristics have proven to be successful in terms of finding optimal or sub-optimal solutions, however, there are problems that require a considerable amount of time and effort. As discussed before, a technique to overcome this obstacle is to take advantage of all hardware resources in order to gain speed and reduce work-load.

Lintzmayer et al. [2011], Tomar et al. [2013], and Markid et al. [2015] proposed bio-inspired algorithms to solve the GCP. The enhancements consisted in either: (1) utilizing a specific scheme for local search, (2) modifying the initialization phase, or (3) embedding a mechanism to gather data about high quality solutions.

Lukasik et al. [2008], Evstigneev e Tursunbay kyzy [2011], and Eblen et al. [2012] implemented modern algorithms to solve the GCP more efficiently. The developers incorporated techniques that consisted in dividing the problem in various tasks, utilizing neighboring vertices for local data transfers or the Master Slave-Model. These projects were developed in order to solve complex computational problems while taking advantage of modern infrastructures.

3. Modern Hybrid Coloring Algorithm

The Hybrid Evolutionary Algorithm (HEA) is a metaheuristic that combines parts of Genetic Algorithms (GAs) with a local search operator (Glass e Prügel-Bennett [2003]). In addition, the HEA applies a highly specialized *crossover* operation in order to create potentially interesting configurations and then improve them with local search (Galinier e Hao [1999]).

Basically, the HEA contains a set of configurations, called a *population*, and modifies them a fixed number of times. Such alterations are done with the crossover operator, which has to be carefully configured for each type of problem.

Each *individual* represents a possible solution for the GCP, and a *conflict* refers to adjacent vertices with the same color.

The HEA pertains the idea to specialize a metaheuristic for a given problem and thus, the Hybrid Coloring Algorithm (HCA) was developed (Galinier e Hao [1999]) by applying HEA to GCP.

The HCA has been tested extensively in benchmark graphs and ranks as one of the best methods to solve the GCP (Glass e Prügel-Bennett [2003]), achieving competitive results, such as better chromatic numbers, than previous outcomes in the literature.

Such algorithm consists of 5 main phases, which are the following:

1. Creates the population with $|P|$ individuals using the greedy saturation algorithm (Brélaz [1979]).

2. After the population is built, it randomly chooses two individuals, which are the parents for that specific iteration.

3. Applies the crossover operator to the parents in each iteration and consequently produces an offspring.

4. Applies the Tabu Search (TS). Such metaheuristic is employed to the offspring and consists in a set of practices in order to solve combinatorial optimization problems, thus functioning as an aid mechanism for local search. This procedure is characterized by the use of a *Tabu-List* (TL), which stores search history information. The HCA was proposed in 1999 utilizing a dynamic scheme for TS, which depends on the current solution and movement, as such, no prior information is known. The tabu tenure remains consistent throughout such process.

5. Replaces the worst parent for the offspring, consequently updating the population and leaving the best element untouched.

Such process is repeated until a certain stop criteria has been satisfied, which are the following (Galinier e Hao [1999]): (1) a specific number of iterations is reached, (2) population diversity is minimal, (3) the best solution is found, or (4) time elapsed without finding an optimal solution.

3.1. The Modern Model

Nowadays, multi-core processors are the standard and have allowed for significant hardware and software advancements. Modern infrastructures have permitted applications to utilize all machine resources and thus, improving time and memory consumption.

In this work, we propose a distributed model, which is able to adapt to the current hardware infrastructure. This means that such model automatically creates F execution flows, based on the number of CPUs provided by the hardware infrastructure. Thus, various instances of the HCA are executed, each one in a different execution flow. The Figure 1 displays a diagram of the distributed model.

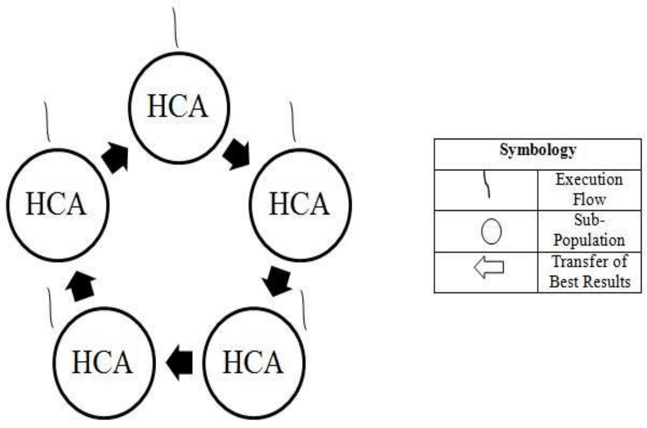

Figure 1: Distributed Model of the HCA.

The synchronization process between execution flows begins with a parameter I, which defines how many iterations will occur in an execution flow i, and the best individual of the respective i flow is sent to $(i + 1)\%F$. Afterwards, the flow i receives the individual sent by $|i - 1|\%F$ and the former substitutes its worst individual for the given element.

Each HCA creates its own population in different search points, and thus, providing a higher probability for the algorithm to direct its exploration to the best solution. The finest result of the entire execution is shared between all flows after $F * I$ iterations.

The stop criterion for the modern algorithm are: (1) number of iterations, (2) execution time, (3) population diversity, (4) maximum number of cycles without convergence and (5) finding the best solution. When the algorithm obtains the finest result (a graph colored without conflicting vertices), the execution halts because the solution can no longer be improved.

If a stop criteria is reached by a single flow, it sends a signal to the other execution flows, requesting to end the execution. However, the Modern HCA only halts if all flows have called for it. If all execution flows demand to stop the operation, every flow will terminate in the next iteration.

The structure of the Modern HCA (MHCA) can be viewed in Algorithm 1.

Algorithm 1: Modern HCA

```
Input: Graph $G$, Integer $k$, Flow_id $id$
Output: the best configuration found
$P = create\_population(|P|)$;
$counter \leftarrow 0$;
while true do
    $choose\_parents(P)$
        Randomly choose two parents $s_1$ and $s_2$ from the population;
        return $s_1$ and $s_2$;
    $crossover(s_1, s_2)$
        for $l$ ($1 \leq l \leq k$) do
            if $l$ is odd $A \leftarrow 1$ else $A \leftarrow 2$;
            choose $i$ such that $V_i^A$ has a maximum cardinality;
            $V_l \leftarrow V_i^A$;
            remove the vertices of $V_l$ from $s_1$ and $s_2$;
        Assign randomly the vertices of $V - (V_1 \cup ... \cup V_k)$;
        return offspring;
    $tabu\_search(offspring)$
        $s \leftarrow generate\_initial\_solution(P)$;
        $initialize\_tabu\_lists(TL)$;
        while Terminate conditions not met do
            $\mathcal{N}_a(s) \leftarrow \{s' \in \mathcal{N}(s) | s' \notin TL$ or $s'$ satisfies at least one aspiration criteria $\}$;
            $s' \leftarrow argmin\{f(s'') | s'' \in \mathcal{N}_a(s)\}$;
            $update\_tabu\_lists(TL, s, s')$;
            // update process depends on whether the dynamic or reactive scheme is
              activated.
            $s \leftarrow s'$;
        return offspring;
    $P = substitute\_worst(P, s_1, s_2, offspring)$;
    synchronization (P, counter, id)
        $c \leftarrow c + 1$;
        if $c < I$ then
            $c \leftarrow 0$;
            $b \leftarrow$ Best solution of $P$;
            send $b$ to Flow $(i + 1) \equiv T$;
            $b' \leftarrow$ Solution sent by Flow $|i - 1| \equiv T$;
            Replace the worst individual of $P$ by $b'$;
    $terminate \leftarrow terminate\_conditions()$;
    Flow_end_control (P, terminate)
        if $terminate$ then
            ask for execution end;
            if All flows asked for end then
                send the best element of $P$ to main flow;
                Terminate this flow;
            else
                Continue executing;
```

4. Evaluating the Modern Hybrid Coloring Algorithm

The experiments were executed in a machine with an Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz with 12 cores; hyperthreading; 32GB of RAM; and running Ubuntu 12.10.

To evaluate the original HCA and its modern counterpart, 47 GCP instances were utilized for the experiment and they are the following:

- *dsjc125.1, dsjc125.5, dsjc125.9, dsjc250.1, dsjc250.5, dsjc250.9, dsjc500.1, dsjc500.5, dsjc500.9, dsjc1000.1, dsjc1000.5, dsjc1000.9*: random graphs dsjc$n.d$, where n is the number of vertices and d the probability that two vertices are adjacent.
- *dsjr500.1, dsjr500.1c, dsjr500.5*: geometric graphs dsjr$n.d$ that are generated by selecting randomly and consistently n vertices in a square and defining the edges between them with a distance less than d. The c at the end of the name means that the instance is a complement graph.
- *flat300_20_0, flat300_26_0, flat300_28_0, flat1000_50_0, flat1000_60_0, flat1000_76_0*: flat$n$_χ_0 are generated by partitioning n vertices in χ classes of almost the same size and later selecting edges between different class vertices. Finding the best coloring is equivalent to restoring the initial partition and thus, the chromatic number of the graph is χ.
- *le450_5a, le450_5b, le450_5c, le450_5d, le450_15a, le450_15b, le450_15c, le450_15d, le450_25a, le450_25b, le450_25c, le450_25d*: le450_χ graphs with 450 vertices and a chromatic number of χ.
- *latin_square, qg.order30, qg.order40, qg.order60, qg.order100*: graphs based on the *Latin Square* problem, where there exists a n by n matrix with n different symbols that appear only once in each row or column. The graph has 900 vertices, with an independent set no greater than 10 vertices. χ is the chromatic number in the name qg.orderχ.
- *r125.1, r125.1c, r125.5, r250.1, r250.1c, r250.5, r1000.1, r1000.1c, r1000.5*: similar to the dsjr$n.d$, they are geometric graphs r$n.d$ generated by selecting randomly and consistently n vertices in a square and defining the edges between them with a distance less than d. The c at the end of the name means that the instance is a complement graph.

For each instance, a parameter k was established, which is the best result found in the literature to color the graph. In case the algorithm does not find an optimal solution with a value of k , $k = k+1$ will be executed until the algorithm solves the problem with zero conflicts. Furthermore, for each instance and k value, the algorithm was executed 50 times. The stop criterion for the classic and Modern HCA (MHCA) in this experiment were the following: (1) 4,000 iterations for the dynamic local search scheme, (2) population diversity is minimal, (3) the algorithm reached the total number of cycles, (4) the algorithm solves the GCP with zero conflicts. It terms of the MHCA, the number of iterations before the synchronization procedure is 100 cycles.

4.1. Results

The results can be classified in the following categories: (1) distance to best k, (2) number of conflicts in the resulting graph and (3) cycles per execution flow during the search process. Each category has its own measurements and represents a unique perspective of the results. Distance to best k is related to the quality of the successful results, number of conflicts represents the quality of unsuccessful results and cycles per execution flow displays the effort performed by the algorithm during the search process.

Figure 2 compares the distance to best k, per instance class, between the classic HCA and the MHCA. Based on the experiments, the MHCA achieved more proximate values to the best coloring result in the literature, therefore, confirming the efficiency of the algorithm in searching for optimal solutions. Nearly all the distances reached by the MHCA are lower or at least equal to its classic counterpart. In just one instance, *dsjc1000.1*, the MHCA achieved a larger distance than the HCA, resulting in a difference of only 1.

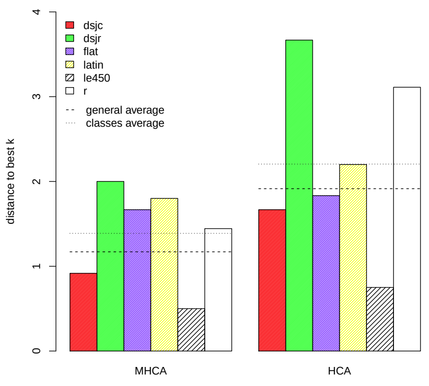

Figure 2: HCA's and MHCA's distance to the best k

The general average indicated in Figure 2 belongs to the HCA and MHCA, for all 47 instances in 50 executions, and the classes average is a straightforward average between the 6 instance classes utilized. In terms of the general average, the distance achieved by the MHCA was 38,8% better than the HCA and 37,04% considering the classes average.

The improvement in the *r* instance classes is the best when compared to the HCA, reaching 53,6%, *dsjc* and *dsjr* classes obtained 45% and 45,5%, *le450* acquired 33,3% of improvement, and *latin* and *flat* got lower values of enhancement, but still significant, 18,2% and 9,9%, respectively.

Figure 3 exhibits, for each instance class, the average number of conflicts in each distance to the best k from the 50 executions. In these graphs, when a line hits zero, there are no conflicts, and the algorithm was successful in coloring every instance for the respective class. These graphs can be analysed by viewing unfavorable results until the number of conflicts is zero, resulting in a successful coloring process .

Relatively, the best scenario for the MHCA occurs in the *r* instances, where the difference begins with 43,4%, and grows rapidly in each increment, going to 61,1%, 81,8%, 92% and 98,5%. Furthermore, the average number of conflicts obtained by the MHCA is zero when best k + 5, notably better than HCA, which obtained zero only when best k + 11.

Additionally, satisfying results belonging to the MHCA were acquired for the *latin* instances, achieving 32,8% better results than the HCA using the best k colors. The difference is accentuated with more colors, with the HCA obtaining zero in best k + 11, while the MHCA reached zero in best k + 6.

In regards to the *dsjr* instances, there is a significant difference (starting with 34,5%) in the average number of conflicts between the HCA and MHCA executions, and such element is maintained using larger color values. The MHCA hits zero in best k + 4, while the HCA achieves such number only in best k + 7.

In the *dsjc* instance classes, the number of conflicts of the MHCA is nearly identical to

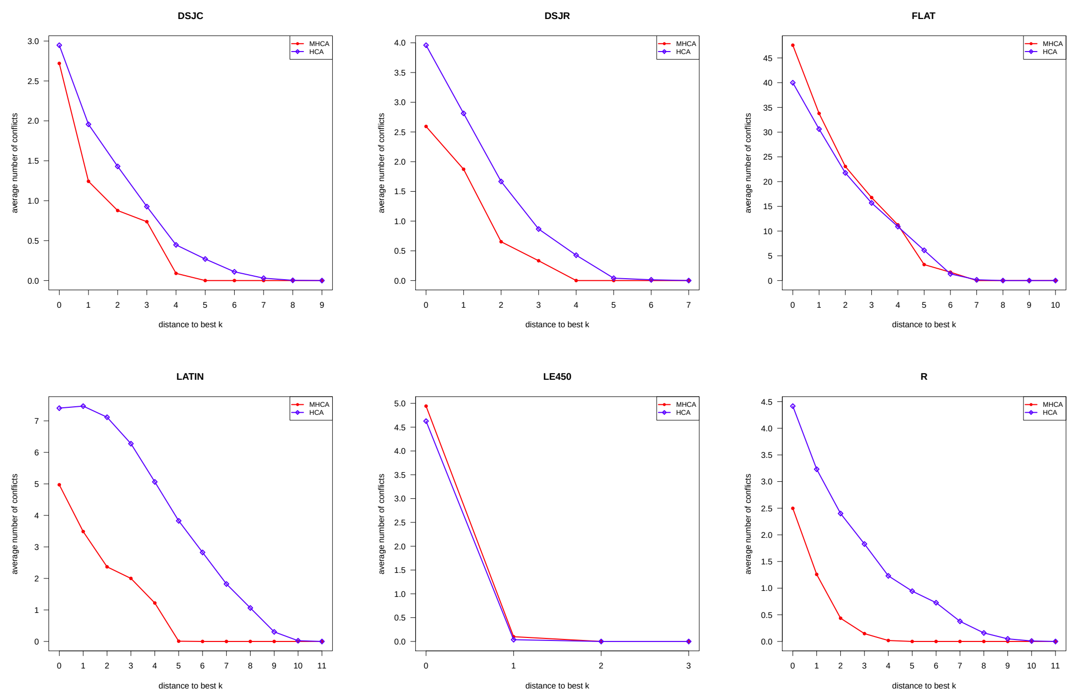

Figure 3: Number of conflicts of the HCA and MHCA for each class and k provided.

its counterpart when trying to color with the best k, resulting in a difference of 7,7%. However, the dissimilarity expands when incrementing the parameter, reaching 79,9% of difference in best $k + 4$. The MHCA hits zero in best $k + 5$, while the classic HCA achieves such value in best $k + 9$.

There are two classes in which the MHCA underperformed when compared to the HCA, they are the *flat* and *le450*. Nonetheless, the difference is not notable, the *flat* instances begins with 15,9% and decreases to 3,1% in best $k + 4$. In regards to *le450*, the dissimilarity is not significant, starting with approximately 6% and hitting zero in best $k + 2$ for both the MHCA and HCA.

The third category of results, shown in Figure 4, is the total number of cycles for each algorithm, which means the effort to color a respective instance, with k colors. The purpose of the MHCA is to distribute the effort between CPUs in order to expand the search and avoid local maximum.

In each execution, the number of cycles refers to the difficulty to color the given graph with an assigned number of colors. During the process, if the algorithm does not color a respective graph, more cycles are required to find a solution. The maximum number of cycles in each algorithm is given from a parameter: 100.000 cycles in the HCA and 4.166 cycles in each execution flow of the MHCA, both following the same stop conditions described in the setup section, which means that when a MHCA flow achieves 4.166 cycles, it continues until all execution flows reach the maximum number of cycles. Thus, the total effort of the MHCA surpasses 9.9984 cycles (24 flows x 4.166), and in most cases, the effort of the MHCA is larger than the HCA when both algorithms were unsuccessful in the coloring process.

The algorithm utilizes more effort and cycles if it obtains an unfavorable solution, on the other hand, an uncomplicated result can be achieved with few iterations and minimal work load. A straight line in a graph belonging to the HCA, such as the one shown in Figure 4, signifies that there is no progress to color an instance of the respective class, such as the *latin_square* instance using between best k and best $k + 7$ colors, in which no execution obtained a successful solution to color the instance. This also occurs with *flat300_28_0* until best $k + 3$, *flat1000_76_0* to best $k + 6$ and *r1000.5* until best $k + 6$.

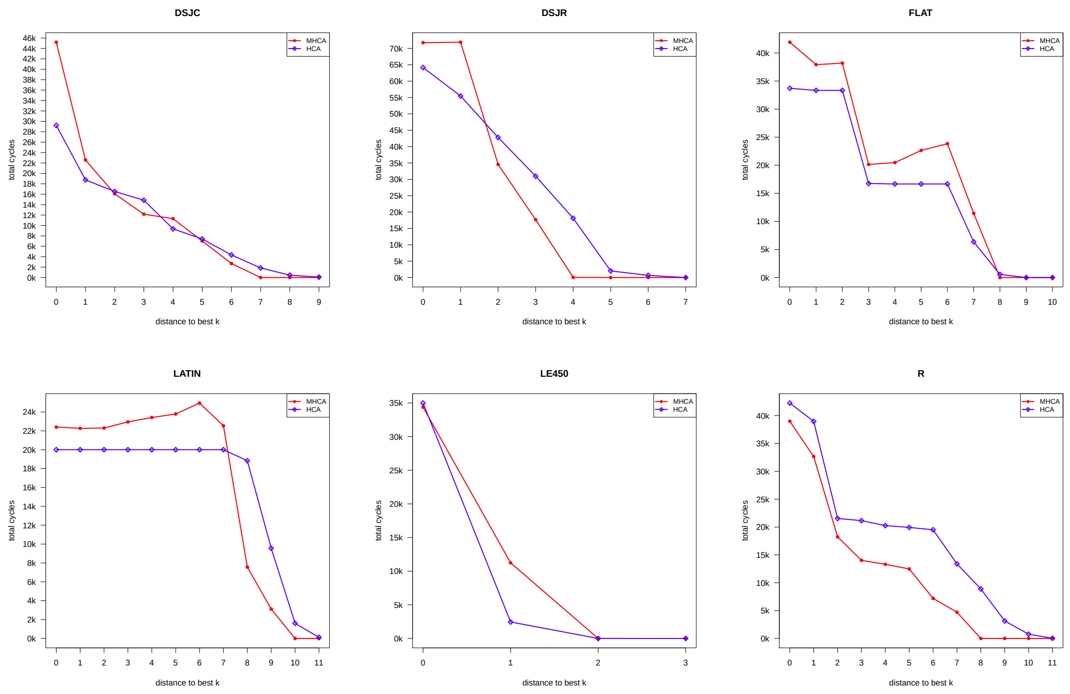

Figure 4: Total number of cycles on the HCA and MHCA for each class and k provided

The MHCA uses more cycles than the HCA when obtaining an unsuccessful solution, however, the higher number of iterations does not justify the lower distances achieved by the MHCA when compared to its counterpart. The main reason why the MHCA achieves better results is due to the efficient exploration of the search space, as shown in nearly every graph, thus, easily coloring with a minimum number of colors.

The results acquired in the evaluation of the MHCA proves that the updated algorithm is in fact an improvement over the original, achieving better results than its predecessor in all three categories. Additionally, it is worth mentioning that the latter is already considered to be one of the finest algorithms for graph coloring, which proves how beneficial the modern version is for the GCP.

4.2. The Instances

The table 1 displays the k-value achieved by the algorithms and the best result, for each instance, known so far in the literature. The column labeled *k-HCA* signifies the k-value of colors that the HCA obtained in its executions, while *k-MHCA* refers to the k-values for the MHCA.

A total of 47 instances were evaluated, and the MHCA achieved best k-values than its counterpart in 16 cases, 34,04% respectively. In just one case the number of colors in the MHCA is worse than the HCA, 2,1% of the instances.

In general terms, the HCA obtained 3,23% of an average difference when compared to the best k known in the literature, while the MHCA obtained more proximate values to best k, reaching 1,85% of an average distance. In 30 instances, the MHCA obtained values equal to best k, while the HCA obtained 25, getting 20% more successful coloring cases with best k.

Based on these results, we can confirm that the MHCA is an interesting and beneficial alternative to the original HCA, achieving k values that are equivalent or proximate to the best re-

Table 1: Colors achieved per instance in the literature (best k column), HCA and MHCA.

instance	best k	k-HCA	k-MHCA	instance	best k	k-HCA	k-MHCA
dsjc125.1	5	5	5	qg.order100	100	100	100
dsjc125.5	17	17	17	latin_square	97	108	106
dsjc125.9	44	45	44	le450_5a	5	6	5
dsjc250.1	8	8	8	le450_5b	5	5	5
dsjc250.5	28	28	28	le450_5c	5	5	5
dsjc250.9	72	75	73	le450_5d	5	5	5
dsjc500.1	12	12	12	le450_15a	15	16	16
dsjc500.5	48	49	48	le450_15b	15	16	16
dsjc500.9	126	132	129	le450_15c	15	16	15
dsjc1000.1	20	20	21	le450_15d	15	16	15
dsjc1000.5	85	85	85	le450_25a	25	25	25
dsjc1000.9	223	232	229	le450_25b	25	25	25
dsjr500.1	12	12	12	le450_25c	25	27	27
dsjr500.1c	84	91	88	le450_25d	25	27	27
dsjr500.5	122	126	124	r125.1	5	5	5
flat300_20_0	20	20	20	r125.1c	46	46	46
flat300_26_0	26	26	26	r125.5	36	38	36
flat300_28_0	28	31	31	r250.1	8	8	8
flat1000_50_0	50	50	50	r250.1c	64	66	65
flat1000_60_0	60	60	60	r250.5	65	67	67
flat1000_76_0	76	84	83	r1000.1	20	20	20
qg.order30	30	30	30	r1000.1c	98	109	101
qg.order40	40	40	40	r1000.5	234	245	241
qg.order60	60	60	60				

sults known in the literature. The improvements offered by the MHCA are significant due to taking advantage of modern hardware resources and thus, developers should be aware of such technology in order to solve complex computational problems more efficiently.

5. Conclusion

The Graph Coloring Problem is considered to be one of the most important concepts and areas in Computer Science. The GCP can be applied in many real-world applications such as *Physical Layout Segmentation*, *Aircraft Scheduling*, *Student Time Table* and *Bi-processor Tasks*. Because of this, many methods have been proposed to solve this problem including metaheuristic algorithms such as *Genetic Algorithms*, *Simulated Annealing*, *Tabu Search*, *Ant Colony Optimization* or *GRASP*. Metaheuristics have proven to be successful in terms of finding optimal or sub-optimal solutions efficiently and rapidly. In fact, algorithms that take advantage of modern computer architectures have become popular in recent years to solve the GCP due to their positive characteristics and overall achievements in the combinatorial optimization area.

The Hybrid Coloring Algorithm (HCA) of Galinier and Hao proposed in 1999 is currently understood to be one the best performing algorithms for graph coloring. This is due to having specific attributes such as operating in a space of infeasible solutions and making use of robust global and local search operators.

The objective of this paper was to analyze in detail the HCA for graph coloring and propose a modern version that benefits from recent computer infrastructures. The model that was implemented in order to improve the HCA is called the Island Model, which basically consists in having sub-populations and exchanging individuals between them.

Based on the results, we can conclude that the modern HCA is a competitive alternative and improvement to the original version developed over 15 years ago. In all three evaluated categories (Distance to best k, average number of conflicts and cycles per execution flow during the search process), the MHCA outperformed its counterpart. Therefore, we can confirm that recent computer architectures can significantly aid in solving complex computational problems, and as such, developers should start transitioning their algorithms to modern standards.

References

Abbasian, R. e Mouhoub, M. (2013). A hierarchical parallel genetic approach for the graph coloring problem. *Applied Intelligence*, 39(3):510–528. ISSN 0924-669X. URL `http://dx.doi.org/10.1007/s10489-013-0429-5`.

Alahmadi, A., Alamri, T., e Hosny, M. (2014). Time efficient demon algorithm for graph coloring with search cut-off property. In *Proceedings of the Science and Information Conference*, p. 254–259.

Becirspahic, L., Dulovic, A., e Nosovic, N. (2013). Parallelization and performance analysis of the simulated annealing algorithm for graph coloring problem. In *International Convention on Information and Communication Technology Electronics and Microelectronics*, p. 1306–1309.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256. ISSN 0001-0782. URL `http://doi.acm.org/10.1145/359094.359101`.

Consoli, P., Collera, A., e Pavone, M. (2013). Swarm intelligence heuristics for graph coloring problem. In *Proceedings of the IEEE Congress on Evolutionary Computation*, p. 1909–1916.

Dabrowski, J. e Kubale, M. (2008). Computer experiments with a parallel clonal selection algorithm for the graph coloring problem. In *Proceedings of International Symposium on Parallel and Distributed Processing*, p. 1–6.

Dawson, L. e Stewart, I. (2013). Improving ant colony optimization performance on the GPU using CUDA. In *IEEE Congress on Evolutionary Computation*, p. 1901–1908.

Djelloul, H., Sabba, S., e Chikhi, S. (2014). Binary bat algorithm for graph coloring problem. In *Proceedings of the World Conference on Complex Systems*, p. 481–486.

Eblen, J. D., Rogers, G. L., Jr., Phillips, C. A., e Langston, M. A. (2012). The use of fast approximate graph coloring to enhance exact parallel algorithm performance. In *Proceedings of the Australasian Symposium on Parallel and Distributed Computing*, p. 31–32. Australian Computer Society, Inc. ISBN 978-1-921770-08-1. URL `http://dl.acm.org/citation.cfm?id=2523685.2523689`.

Evstigneev, V. e Tursunbay kyzy, Y. (2011). Graph coloring in a class of parallel local algorithms. *Numerical Analysis and Applications*, 4(3):189–198. ISSN 1995-4239. URL `http://dx.doi.org/10.1134/S1995423911030013`.

Galinier, P. e Hao, J.-K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397. ISSN 1382-6905. URL `http://dx.doi.org/10.1023/A%3A1009823419804`.

Glass, C. A. e Prügel-Bennett, A. (2003). Genetic algorithm for graph coloring: Exploration of Galinier and Hao's algorithm. *Journal of Combinatorial Optimization*, 7(3):229–236. ISSN 1382-6905. URL `http://dx.doi.org/10.1023/A%3A1027312403532`.

Kokosiński, Z., Kołodziej, M., e Kwarciany, K. (2004). *Proceedings of the International Conference of Computational Science*, chapter Parallel Genetic Algorithm for Graph Coloring Problem, p. 215–222. Springer Berlin Heidelberg. ISBN 978-3-540-24685-5. URL `http://dx.doi.org/10.1007/978-3-540-24685-5_27`.

Lintzmayer, C. N., Mulati, M. H., e d. Silva, A. F. (2011). Toward better performance of colorant aco algorithm. In *Proceedings of the Chilean Computer Science Society*, p. 256–264.

Lukasik, S., Kokosiński, Z., e Świketoń, G. (2008). Parallel simulated annealing algorithm for graph coloring problem. In *Proceedings of the International Conference on Parallel Processing and Applied Mathematics*, p. 229–238. Springer-Verlag. ISBN 3-540-68105-1, 978-3-540-68105-2. URL `http://dl.acm.org/citation.cfm?id=1786194.1786222`.

Markid, H. Y., Dadaneh, B. Z., e Moghaddam, M. E. (2015). A new tabucol embedded artificial bee colony based algorithm for graph coloring. In *Proceedings of the International Conference on Computer and Knowledge Engineering*, p. 112–117.

Myszkowski, P. (2005). New evolutionary approach to the GCP: a premature convergence and an evolution process character. In *Proceedings of the International Conference on Intelligent Systems Design and Applications*, p. 338–343.

Salari, E. e Eshghi, K. (2005). An ACO algorithm for graph coloring problem. In *Proceedings of the Congress on Computational Intelligence Methods and Applications*, p. 5 pp.–.

Tomar, R. S., Singh, S., Verma, S., e Tomar, G. S. (2013). A novel ABC optimization algorithm for graph coloring problem. In *Proceedings of the Computational Intelligence and Communication Networks*, p. 257–261.

Yesil, C., Yılmaz, B., e Korkmaz, E. E. (2011). Hybrid local search algorithms on graph coloring problem. In *Proceedings of the International Conference on Hybrid Intelligent Systems*, p. 468–473.