A Memetic Algorithm with Two Distinct Solution Representations for the Partition Graph Coloring Problem

Petrica C. Pop[1], Bin Hu[2], and Günther R. Raidl[2]

[1] Tech. Univ. Cluj-Napoca, North Univ. Center Baia-Mare
76 Victoriei, 430122 Baia-Mare, Romania

[2] Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria

petrica.pop@ubm.ro, hu@ads.tuwien.ac.at, raidl@ads.tuwien.ac.at

Abstract. In this paper we propose a memetic algorithm (MA) for the partition graph coloring problem. Given a clustered graph $G = (V, E)$, the goal is to find a subset $V^* \subset V$ that contains exactly one node for each cluster and a coloring for V^* so that in the graph induced by V^*, two adjacent nodes have different colors and the total number of used colors is minimal. In our MA we use two distinct solution representations, one for the genetic operators and one for the local search procedure, which are tailored for the corresponding situations, respectively. The algorithm is evaluated on a common benchmark instances set and the computational results show that compared to a state-of-the-art branch and cut algorithm, our MA achieves solid results in very short run-times.

1 Introduction

The partition graph coloring problem (PGCP) belongs to the class of problems usually referred to as generalized network design problems (GNDPs). This class of problems is obtained by generalizing in a natural way many network design problems by considering a related problem on a clustered graph, where the original problem's feasibility constraints are expressed in terms of the clusters, i.e., node sets instead of individual nodes.

In the literature, several GNDPs have already been considered such as the generalized minimum spanning tree problem, the generalized traveling salesman problem, the generalized vehicle routing problem, the generalized (subset) assignment problem, the generalized fixed-charge network design problem, etc. All such problems belong to the class of $\mathcal{NP}$-complete problems, are typically harder to solve in practice than their original counterparts and nowadays are intensively studied due to their interesting properties and important real-world applications in telecommunication, network design, resource allocation, transportation problems, etc. Nevertheless, many practitioners are still reluctant to use these models

for modeling and solving practical problems because of the complexity of finding optimal or near-optimal solutions.

The PGCP was introduced by Li and Simha [10] motivated by the wavelength routing and assignment problem in wavelength division multiplexing optical network. The authors proved that the problem is $\mathcal{NP}$-complete. In this context several approaches for solving the problem have been proposed: heuristic algorithms [10], a branch-and-price algorithm [9], and a tabu search algorithm [15]. Demange *et al.* [3, 4] considered this type of graph coloring problem in the framework of GNDPs and named it selective graph coloring problem. They investigated some special classes of graphs and determined the complexity status of the PGCP in these classes. Extending our previous work [17], we propose a memetic algorithm (MA) which uses two different solution representations for the genetic operators and for the local search procedure.

2 Definition of the Partition Graph Coloring Problem

Formally, the partition graph coloring problem is defined on an undirected graph $G = (V, E)$ with the set of nodes V and the set of edges E. The set of nodes is partitioned into p mutually exclusive nonempty subsets, called clusters, $V_1, \ldots, V_p$ with $V_1 \cup \ldots \cup V_p = V$ and $V_i \cap V_j = \emptyset$ for all $i, j \in \{1, \ldots, p\}$ and $i \neq j$. The PGCP consists of finding a set $V^* \subset V$ such that $|V^* \cap V_i| = 1$, i.e., V^* contains exactly one node from each cluster V_i for all $i \in \{1, \ldots, p\}$, and the graph induced by V^* is k-colorable where k is minimal. The PGCP reduces to the classical graph coloring problem when all the clusters are singletons.

An illustration of the PGCP and an optimal solution with two colors is shown in Figure 1. In this example the graph $G = (V, E)$ has 10 nodes partitioned into 5 clusters. The optimal solution makes use of two colors: the first is used to color the nodes 3, 5 and 10 and the second for the nodes 2 and 8.

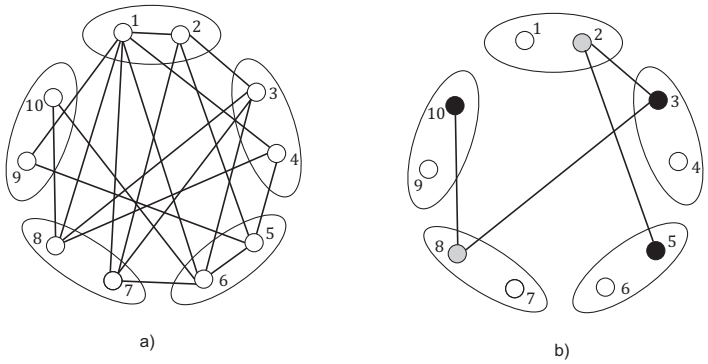

Fig. 1. a) An instance of PGCP and b) an optimal solution with two colors.

3 The Memetic Algorithm

Memetic algorithms have been introduced by Moscato [13] and denote a family of metaheuristic algorithms that emphasis on the used of a population-based approach with separate individual learning or local improvement procedures for problem search. It is frequently also interpreted as a genetic algorithm (GA) hybridized with a local search procedure to intensify the search in the solutions space. GAs typically are not well suited for fine-tuning structures which are close to optimal solutions. Therefore, incorporating local improvement as an additional operator to the GA can be beneficial to obtain a competitive algorithm. MAs have been recognized as a powerful algorithmic paradigm, being applied successfully to solve many combinatorial optimization problems such as the vehicle routing problem [14], the generalized traveling salesman problem [1], etc. We use a memetic algorithm for solving the PGCP. In the following we will describe the solution representations, genetic operators and the local search procedure.

Genetic representation: It is known that a good representation scheme is important for the performance of the GA and it should define noteworthy genetic operators to the problem in order to minimize the computational effort within these procedures. In order to meet this requirement, an individual is represented based on the color classes, i.e., $U = \{U_1, U_2, \ldots, U_k\}$ where $U_i,\ i \in \{1, \ldots, k\}$ consists of all nodes assigned to color i. Obviously, $U_1 \cup U_2 \cup \ldots \cup U_k = V^* \subseteq V$.

Initial population: In our MA we use a randomized version of the *onestepLF* algorithm introduced in [10]. The basic idea of onestepLF is to choose for each cluster the node with smallest degree to be in the solution. Then in each iteration among the uncolored nodes the one with largest degree is colored with the smallest possible color index so that no conflicts occur with its adjacent nodes. In our case we want to reach a certain level of diversity in the population, therefore we do not necessarily use for each cluster the node with the smallest degree, but choose it randomly.

Fitness evaluation: Every solution has a fitness value assigned to it, which measures its quality. In our case, the fitness value is given by $|U|$, the total number of used color classes, which corresponds to k, the number of different colors necessary for coloring the solution graph. The aim is to find to find a partition into color classes that minimizes k. The selection is purely based on the fitness values and ties (i.e. solutions with equal k) are broken randomly.

Crossover: Two parents are selected from the population by the binary tournament method. Consider two parent solutions represented based on the color classes

$$U^1 = \{U_1^1, U_2^1, \ldots, U_k^1\} \text{ and } U^2 = \{U_1^2, U_2^2, \ldots, U_{k'}^2\},$$

then the crossover operator is defined as follows:

1. start from parent U^1 and select a color class (partition) and copy it to the offspring;
2. delete from both parents the selected nodes and all other nodes that belong to the same clusters as the selected ones;
3. from parent U^2, select a color class, move it to the offspring and then remove from both parents the corresponding nodes analogously to step 1;
4. we continue this process until all nodes from the parents U^1 and U^2 are assigned to the offspring or removed.

An example of how the crossover operator works on the sample instance in Figure 1 is illustrated in Table 1.

Table 1. Example for the crossover operator.

Parent 1	Parent 2	Offspring	
$U_1^1 = \{3, 5, 10\}$ $U_2^1 = \{1\}$ $U_3^1 = \{8\}$	$U_1^2 = \{2, 4, 8\}$ $U_2^2 = \{6\}$ $U_3^2 = \{9\}$	{} {} {}	Select U_1^1 as partition with most nodes and copy to the offspring. Delete copied nodes and corresponding nodes from the selected clusters from both parents.
$U_1^1 = \{\}$ $U_2^1 = \{1\}$ $U_3^1 = \{8\}$	$U_1^2 = \{2, 8\}$ $U_2^2 = \{\}$ $U_3^2 = \{\}$	$\{3, 5, 10\}$ {} {}	Select U_1^2 as partition with most nodes and copy to the offspring. Delete copied nodes and corresponding nodes from the selected clusters from both parents.
$U_1^1 = \{\}$ $U_2^1 = \{\}$ $U_3^1 = \{\}$	$U_1^2 = \{\}$ $U_2^2 = \{\}$ $U_3^2 = \{\}$	$\{3, 5, 10\}$ $\{2, 8\}$	A new solution to the PGCP making use of two colors.

During the steps 1 and 3, the choice of selecting a color class to be moved to the offspring can be done either randomly or according to their size and preferring the largest one. The latter strategy will likely generate offsprings with lower k, but the diversity will be lower. Since we use a local improvement procedure for intensification, we select the color classes randomly.

Mutation: Mutation is another important feature of genetic algorithms since it diversifies the search directions and avoids convergence to local optima. In our algorithm we randomly choose between these two procedures:

- Pick a random color class and try to move each of its nodes to another color class so that no conflicts occur. If this is not possible, the node stays in its original color class.
- Remove a randomly selected node from the color class where it is assigned to and insert it to a different color class so that no conflicts occur. If necessary, a new color class is created.

While the first method is generally able to modify the solution to a greater extent, it does not create new color classes, so the solution never gets worse.

Due to this restriction, in some situations this method is not able to cause any modifications. The second method changes the color assignment of one node only, but it is possible that a new color class is created. In our experiments these two mutation methods complement each other well.

Local improvement procedure: By solely relying on the genetic operators, experiments have shown that the algorithm converges too slowly and there is no guarantee that the final solution is even a local optimum. Therefore we apply a local improvement procedure on an offspring after it has been created by the genetic operators with certain probability.

For this purpose we consider the solution in a more compact representation than the genetic representation in order to reduce the search space. We only use the selection of nodes without specifying the coloring information here, i.e., solution $V^* = \{v_1, \ldots, v_p\},\ v_i \in V_i,\ i \in \{1, \ldots, p\}$. This is a common approach for representing solutions in GNDPs [16]. The problem here is that evaluating the solution, i.e., computing the necessary number of colors by solving the classical graph coloring problem, is $\mathcal{NP}$-hard. Therefore we use the DANGER construction heuristic proposed by Glover *et al.* [7]. The central idea is to color one node in each iteration that has the highest node-danger value (which indicates how danger it is to keep it for later) with a color that has the lowest color-danger value. In a previous version we used the DSATUR heuristic [2] which is faster since it uses simpler node and color evaluations, but the results of the DANGER heuristic are clearly better. There certainly are lots of options to choose from when it comes to solving the classical graph coloring problem. Using exact approaches such as mixed integer programming or constraint programming [8, 12] or even metaheuristics [11] would consume too much time since this process has to be done a multitude of times during local search when evaluating neighbor solutions. For this reason we chose the current evaluation algorithm that is focused on fast run-times.

A common drawback of an one-sided evaluation criterion is that there is always a large amount of solutions with equal k, i.e., they have the same number of colors. Therefore, we use an additional criterion that is commonly used in graph coloring problems: the number of conflicts. The idea is that if two solutions can be colored with k colors, we apply a modified version of DANGER heuristic that attempts to color these solutions with $k-1$ colors and minimizes the number of conflicts. Then the solution with less conflicts is considered the better one and will be used in the algorithm. However, we keep the original coloring information that uses k colors in order to avoid infeasible solutions being generated.

For local search we use a standard node exchange neighborhood structure, i.e., the neighborhood of a solution $V^* = \{v_1, \ldots, v_p\}$ consists of all node vectors in which for precisely one cluster V_i the node v_i is replaced by a different node v_i' of the same cluster. In preliminary tests we also tried changing nodes of two clusters, but the size of the neighborhood becomes too large and the run-time increases too drastically. We follow a best improvement strategy since solutions are distinguishable accurately due to the finely granulated evaluation.

4 Experimental Results

Our experiments were run as single threads on a Intel Core i7 PC with 3.4 GHz and 16 GB memory. We use the `Rand`-set of instances [5] that was also used in [6]. It contains randomized instances with 20 to 120 nodes partitioned into 10 to 60 clusters, respectively. We performed 30 independent runs for each instance and determined the average and standard deviations of the final objective values. Each run was terminated after generating 2000 solutions without improvements. The probability for local improvement was set to 30%. We compare two MA variants: one that only uses the number of colors as evaluation criterion (MA1) and one that uses number of colors and the number of conflicts for evaluation (MA2). Table 2 contains experimental results on instances with 20 – 120 nodes and an edge density of 0.5 while Table 3 contains results on instances with 90 nodes and an edge density of 0.1 – 0.9. Each line corresponds to a set of 5 different instances. Both tables show the instance characteristics, the lower and upper bounds obtained by the branch and cut (B&C) approach [6] within two hours run-time, followed by the average objective values of the final best solutions, their standard deviations, and the run-time in seconds for the MA variants.

Table 2. Experimental results on instances with different size.

Instance set		B&C		MA1			MA2		
nodes	density	LB	UB	$\overline{obj}$	sd	$\overline{time}$	$\overline{obj}$	sd	$\overline{time}$
20	0.5	3	3	3.00	0.00	0.02s	3.00	0.00	0.14s
40	0.5	4	4	4.50	0.51	0.10s	4.00	0.00	0.60s
60	0.5	5	5	5.96	0.20	0.31s	5.63	0.49	2.00s
70	0.5	6	6	6.86	0.40	0.53s	6.06	0.24	3.33s
80	0.5	6	6	7.66	0.48	0.80s	6.94	0.29	4.90s
90	0.5	6	7	8.22	0.42	1.21s	7.55	0.50	7.49s
100	0.5	6	7	8.90	0.30	1.74s	7.93	0.30	11.04s
120	0.5	7	8	10.26	0.44	3.41s	9.22	0.43	21.05s

Table 3. Experimental results on instances with different density.

Instance set		B&C		MA1			MA2		
nodes	density	LB	UB	$\overline{obj}$	sd	$\overline{time}$	$\overline{obj}$	sd	$\overline{time}$
90	0.1	2	3	3.13	0.33	0.22s	3.09	0.29	1.37s
90	0.2	3	4	4.71	0.45	0.52s	4.41	0.49	3.24s
90	0.3	4	5	6.06	0.24	0.78s	5.52	0.56	4.90s
90	0.4	5	6	7.59	0.49	1.07s	6.79	0.83	6.54s
90	0.5	6	7	8.22	0.42	1.21s	7.55	0.50	7.49s
90	0.6	8	8	10.98	0.34	1.88s	10.50	0.87	11.95s
90	0.7	10	10	12.93	0.38	2.37s	12.39	1.12	14.83s
90	0.8	12	12	15.55	0.51	3.38s	15.18	0.80	20.98s
90	0.9	16	16	17.69	0.86	7.38s	17.27	0.98	45.75s

We observe that while MA2 consumes more time than MA1, the results are significantly better, particularly on larger instances the difference is approximately one color. This underlines that using the number of colors alone as evaluation criterion is insufficient because the search process circles around plateau regions that contain equally good solutions. When we additionally aim at minimizing the number of conflicts, the MA gets valuable information of what it should focus on. Compared to B&C, MA2 is worse in terms of solution quality. However, we have to take into account that B&C has a time limit of two hours while MA2 finishes in less than one minute. Therefore, MA2 is a practical approach when it comes to time-critical applications and/or large instances due to its excellent scalability.

5 Conclusions and Future Work

We proposed a memetic algorithm (MA) for the partition graph coloring problem that uses two distinct solution representations. For maintaining a diverse population and to keep the computational effort for genetic operators low, we use a full solution representation for crossover and mutation. In contrast, we use a more compact and incomplete solution representation during local search. Both representations work well in combination in the MA. During local search, we observed that minimizing the number of colors results in many solutions of equal quality. Therefore, we use a second evaluation criterion based on the number of conflicts when using one color less. Computational experiments on common benchmark instances sets show that although the MA is not always able to find the optimal solutions, it produces solid results with very low run-times and therefore has excellent scalability when it comes to large instances.

For future work, we want to consider a further incomplete solution representation which is based on characterizing the colors of the clusters. The challenge will be to develop efficient algorithms for choosing the nodes in the clusters that are compatible with the color assignments. We also want to consider further evaluation criteria besides color and conflicts so that more fine-tuned measurements depending on specific situations are possible.

Acknowledgments: This work was supported by grant PHC BOSPHORE 2012 N 26284RB which is gratefully acknowledged.

References

1. Bontoux, B., Artigues, C., Feillet, D.: A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. Computers and Operations Research 37(11), 1844–1852 (2010)
2. Brélaz, D.: New methods to color the vertices of a graph. Communication of ACM 22(4), 251–256 (1979)
3. Demange, M., Monnot, J., Pop, P., Ries, B.: Selective graph coloring in some special classes of graphs. In: Combinatorial Optimization, Lecture Notes in Computer Science, vol. 7422, pp. 320–331 (2012)

4. Demange, M., Monnot, J., Pop, P., Ries, B.: On the complexity of the selective graph coloring problem in some special classes of graphs. Theoretical Computer Science (2013), in press
5. Frota, Y., Maculan, N., T.F., N., Ribeiro, C.: Instances for the partition coloring problem. www.ic.uff.br/ celso/grupo/pcp.htm
6. Frota, Y., Maculan, N., Noronha, T.F., Ribeiro, C.C.: A branch-and-cut algorithm for the partition coloring problem. Networks 55(3), 194–204 (2010)
7. Glover, F., Parker, M., Ryan, J.: Coloring by tabu branch and bound. DIMACS Series on Discrete Mathematics and Theoretical Computer Science 26, 285–308 (1996)
8. Gualandi, S., Malucelli, F.: Exact solution of graph coloring problems via constraint programming and column generation. INFORMS Journal on Computing 24(1), 81–100 (2012)
9. Hoshino, E.A., Frota, Y.A., de Souza, C.C.: A branch-and-price approach for the partition coloring problem. Operations Research Letters 39(2), 132–137 (2011)
10. Li, G., Simha, R.: The partition coloring problem and its application to wavelength routing and assignment. In: 1st Workshop on Optical Networks (2000)
11. Lü, , Hao, J.: A memetic algorithm for graph coloring. European Journal of Operational Research 203, 241–250 (2010)
12. Mehrotra, A., Trick, M.A.: A column generation approach for graph coloring. INFORMS Journal on Computing 8, 344–354 (1996)
13. Moscato, P.: Memetic algorithms: A short introduction. In: Corne, D., et al. (eds.) New Ideas in Optimization, pp. 219–234. McGraw Hill (1999)
14. Ngueveu, S.U., Prins, C., Calvo, R.W.: An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Computers and Operations Research 37(11), 1877–1885 (2010)
15. Noronha, T.F., Ribeiro, C.C.: Routing and wavelength assignment by partition colouring. European Journal of Operational Research 171(3), 797–810 (2006)
16. Pop, P.C.: Generalized network design problems. Modeling and Optimization. De Gruyter Series in Discrete Mathematics and Applications, Germany (2012)
17. Pop, P.C., Hu, B., Raidl, G.R.: A memetic algorithm for the partition graph coloring problem. In: Extended Abstracts of the 14th International Conference on Computer Aided Systems Theory. pp. 167–169. Gran Canaria, Spain (2013)