Selecting Suitable Solution Strategies for Classes of Graph Coloring Instances Using Data Mining

Nur Insani
Mathematics Education Department
Yogyakarta State University
Indonesia
nurinsani2001@yahoo.com

Kate Smith-Miles
School of Mathematical Sciences
Monash University
Australia
kate.smith-miles@monash.edu

Davaatseren Baatar
School of Mathematical Sciences
Monash University
Australia
davaatseren.baatar@monash.edu

***Abstract*—The Maximal Independent Set (MIS) formulation tackles the graph coloring problem (GCP) as the partitioning of vertices of a graph into a minimum number of maximal independent sets as each MIS can be assigned a unique color. Mehrotra and Trick [5] solved the MIS formulation with an exact IP approach, but they were restricted to solving smaller or easier instances. For harder instances, it might be impossible to get the optimal solution within a reasonable computation time. We develop a heuristic algorithm, hoping that we can solve these problems in more reasonable time. However, though heuristics can find a near-optimal solution extremely fast compared to the exact approaches, there is still significant variations in performance that can only be explained by the fact that certain structures or properties in graphs may be better suited to some heuristics more than others. Selecting the best algorithm on average across all instances does not help us pick the best one for a particular instance. The need to understand how the best heuristic for a particular class of instance depends on these graph properties is an important issue. In this research, we use data mining to select the best solution strategies for classes of graph coloring instances.**

Keywords—graph coloring; integer linear programming; algorithm selection; data mining; algorithm footprint

I. INTRODUCTION

Many practical problems can be formulated as combinatorial optimization problems, which can then frequently be expressed naturally in terms of graphs and as integer linear programs. One of the most famous problems in graph theory is graph coloring. Graph coloring, or proper coloring, has been a popular research topic since its introduction to solve the map coloring problem more than 150 years ago. Consider an undirected graph $G = (V, E)$ where V is a set of vertices (nodes) and E is the set of edges, with $|V| = n$ and $|E| = m$. The Graph Coloring Problem (GCP) is defined as coloring the vertices $v \in V$ such that no two adjacent vertices $(u, v) \in E$ are assigned the same color. The most common type of graph coloring seeks to minimize the number of colors for G. The minimum number of colors with which the vertices of G can be colored is called the chromatic number of G, denoted by $\chi(G)$. The coloring problem is then to determine $\chi(G)$ and to find a coloring of G that uses $\chi(G)$ colors. GCP can be formulated in many different ways.

A formulation of GCP based on maximal independent sets was introduced by Mehrotra and Trick [5], and it is known as the Maximal Independent Set (MIS) formulation. They formulated the GCP as a problem of partitioning the vertices of a graph into a minimum number of maximal independent sets as each maximal independent set can be assigned a unique color. The MIS formulation does not have a symmetry problem like the standard model because it does not involve assigning a specific color to a vertex but it simply defines which vertices receive the same color. One of the advantages of the MIS formulation is the lower bound provided by the linear relaxation of the formulation is at least as good as the one provided by the standard formulation, and probably better. The computational results show that Mehrotra and Trick's approach can consistently solve instances with up to 70 vertices for random graphs and 250 vertices for random geometric graph. Thus, they have successfully solved two NP-hard problems, the Maximum Weight Independent Set (MWIS) problem and an Integer Programming (IP) pricing

sub-problem, and so far it gives the most efficient exact methods to solve GCP of small to medium sizes.

However, the fact that the method involves solving two NP-Hard problems means that it will not scale up to larger instances. Previous experiments showed that most of the time of the column generation technique is spent in solving the pricing sub-problem, while solving the master problem of LP relaxation takes a small fraction of the whole time. The MIS formulation introduced by Mehrotra and Trick has some interesting properties: it does not have a symmetry problem: and a vertex $v \in V$ is not dominated if and only if the corresponding constraint is facet defining such that the linear relaxation of the formulation provides good lower bound. These properties make it attractive and raise some interesting questions such as: Can a feasible (optimal) solution to LP relaxation lead us to a good feasible (or even optimal) solution to the GCP? What is the gap between the best integral solution obtained from an optimal solution of the LP relaxation and optimal solution of GCP? How does one construct a good integral solution efficiently from the optimal solution of the LP relaxation? Can the final pricing sub-problem used to get a good approximate solutionto the GCP? Hence to answer the above questions, we propose four different strategies to solve the GCP by exploiting the fractional solution of the LP relaxation. They are LP-IP which solves the LP relaxation exactly and constructs an integral solution using the exact IP method, LPapprox-IP which solves the LP relaxation approximately and constructs an integral solution using the exact IP method, LP-intHeuristic which solves the LP relaxation exactly and constructs an integral solution using a rounding heuristic, and LPapprox-intHeuristic which solves the LP relaxation approximately and construct an integral solution using a rounding heuristics.

Although heuristics might be able to find good feasible solutions extremely fast compared to exact approaches, there is still significant variation in performance that can only be explained by the fact that certain structure properties in graphs may be better suited to some heuristics more than others. The need to understand how the best heuristic for particular classes of instances depends on these graph properties is an important issue. Time and complexities can be minimized by knowing which algorithm to use for a particular instance rather than testing all algorithms by trial and error. Selecting the best algorithm on average across all instances does not help us pick the best one for a particular instance. Wolpert and Macready's [15] No-Free-Lunch (NFL) theorems tell us that if an algorithm does particularlywell on one class of problems then there are likely to be other classes of instances where it will perform poorly. Instances arising from some applications are actually not as hard as the theory predicts; while others can be just as bad as the worst case; so it is valuable to understand how an algorithm performs on various kinds of instances. The features of instances can help determine this, and some recent work [13] has begun to show how the features of graphs determine the performance of graph coloring heuristics like DSATUR [1] and tabu search [4]. In this research, we extend such analysis to understand how our four proposed approaches above performs on average across all test instances, as well as to perform data mining analysis to determine how the features of graphs affect algorithm performance. Specifically, we determine if there are particular classes of instances where one approach dominates the others, and aim to characterize the features of such instances. Thus, by using the powerful facet-defining MIS formulation for GCP, the goals of this research are firstly to find out if computational efficiencies can be obtained by using the LP relaxation to yield integer feasible solutions, and whether the LP needs to be solved exactly or if an approximation is sufficient. Likewise, we determine if the integer feasible solution needs to be solved exactly from the LP relaxation, or does a heuristic rounding procedure yield the same results? We will compare the proposed heuristic algorithms to general purpose heuristics commonly used for graph coloring, and downloaded from Joe Culberson's graph coloring web resources page [3]. We will also study the features of a collection of the graphs and determine if these features are predictive of algorithm performance for classes of instances. All of the computational experiments will use the DIMACS benchmark instances and some additional randomly generated instances (geometric and random graphs) from generators available from Culberson's web site [17].

II. Preliminary Concepts

A. Formulation of GCP

In this section, we introduce the classical Integer Linear Programming models for GCP. Let C be an upper bound on the number of colors. Then a straight forward ILP model for GCP can be defined using $(C + 1)|V|$ binary variables [2]. The standard formulation is

$$min \sum_{c=1}^{C} y_c$$

s.t.

$$\sum_{c=1}^{C} x_{vc} = 1 \qquad \forall v \in V \qquad (1)$$

$$x_{vc} + x_{wc} \leq y_c \qquad \forall (v,w) \in E, \forall c \in \{1,2,\dots C\} \qquad (2)$$

$$x_{vc}, y_c \in \{0,1\} \qquad \forall v \in V, \forall c \in \{1,2,\dots C\} \qquad (3)$$

where $x_{vc} = 1$ if color c is assigned to vertex v and $x_{vc} = 0$ otherwise. The binary variables y_c indicate whether color c is used in some vertices, i.e. $y_c = 1$ if $x_{vc} = 1$ for some vertex v. Constraint (1) ensures that each vertex receives exactly one color, while constraint (2) ensures that adjacent vertices have different colors. Finally constraint (3) imposes the variables to be binary.

In [6], Mehrotra and Trick proposed an alternative formulation for the GCP. The formulation relies on the fact that any set of vertices that have the same color is an independent set. In other words, independent sets can be used to represent the set of vertices that have the same color. Based on that the GCP can be formulated as (The Maximal Independent Set Formulation):

$$min \sum_{j \in J} x_j$$

s.t.

$$\sum_{j \in J; v \in S_j} x_j \geq 1 \quad \forall v \in V \tag{4}$$

$$x_j \in \{0,1\} \quad \forall j \in J \tag{5}$$

where J is the index set of all maximal independent sets of G. The binary variables x_j indicate whether the vertices in the maximal independent set S_j could be assigned to the same color or not. In other words, $x_j = 1$ implies that vertices that are in the maximal independent set S_j could have the same color, while $x_j = 0$ implies that the vertices are not required to have the same color. Thus the objective function is to find the minimum number of maximal independent sets that cover all vertices of the graph G. Constraint (4) ensures that every vertex v in the graph must belong to at least one maximal independent set (i.e. must receive at least one color). Constraint (5) states that variable x_j must be binary.

B. Heuristics Approach to the GCP

Good heuristic algorithms are essential to tackle hard optimization problems. Using heuristics one might be able to find good feasible solution quickly. Moreover, heuristics are useful to tighten the bounds and consequently to reduce the search space, especially in enumerative approaches such as branch and bound/cut/price. This motivates the large amount of literature concerning the heuristic and metaheuristic approaches for GCP. Some of the most popular heuristics and metaheuristics found in the literature are DSATUR [1] and tabu search [4].

DSATUR is one of the most well-known exact algorithms and is exact for bipartite graphs [1]. It is a sequential vertex coloring that successively colors the vertices sorted in predetermined order based on the saturation degree of a vertex. Saturation degree of a vertex is the number of different colors connected to the vertex. DSATUR works by subdividing the problem into sub-problems. TABU Search (TS) is a higher level heuristic algorithm for solving combinatorial optimization problems. It is an iterative improvement procedure that starts from any initial feasible solution and attempts to determine a better solution. The version of this algorithm proposed by Hertz [4] for graph coloring is referred to as TABUCOL.

C. Algorithm Selection

Research into algorithm performance has led to the question of which algorithm is the most suitable for particular domains and certain instances. In her research, Smith-Miles [12] has formulated the questions: For a particular problem domain, what are the features or characteristics of particular instances of the problem that are likely to correlate with good or bad algorithm performance? Can we model the relationship between these characteristics and algorithm performance? To answer those questions, we can use the algorithm selection framework proposed by John Rice in 1976 [7] as follows: For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ minimizes the performance mapping $y(\alpha(x)) \in Y$ [8]. By using this model, there are four essential components that can be derived for graph coloring problems:

1. The problem space P: the input graphs defined by a set of vertices and edges.
2. The feature space F: the features or characteristics of the graph.
3. The algorithm space A: all algorithms that we consider to solve the graph coloring problems.
4. The performance space Y: the minimum number of colors found by an algorithm.

The collection of data describing $\{P, F, A, Y\}$ is known as meta-data [12]. The role of data mining is to learn the relationship between the feature space and the performance space so that performance can be predicted based on the features of an instance alone. In addition, data visualization techniques can be employed to gain insights into the relationships in the high dimensional feature space. Data mining methods will be utilized in this research to provide insights into the strengths and weaknesses of the solution strategies used in the research.

III. Solution Strategies Based on LP Relaxation

We present our approaches and strategies developed to give answers to the research questions. It includes the preprocessing process before we begin the main computation, the pricing problem, and our four different approaches to solve the final solution of LP relaxation to obtain a feasible integral solution. From now on, without loss of generality, we may assume that all vertices in a graph are not dominated. If the graph has dominated vertices then using some preprocessing we can always reduce the CG problem into a smaller problem where no vertices are dominated.

A. LP Relaxation and Column Generation

We developed four greedy heuristics that we use to solve the pricing problem. We use the best solution from these heuristics to generate a new column for the restricted master problem (RMP) if the reduced cost is negative. If the heuristics fail, we solve the MWIS exactly. However, we terminate the solver whenever it finds a column with negative reduced cost. In this way, MWIS problem is solved exactly only to prove the optimality. The four greedy heuristics are Heuristic Based on the Maximum Weight, Heuristic Based on the Net Marginal Weight, Heuristic Based on the Weight Ratio, and Heuristic Based on the Weight of Neighborhoods.

B. Generating Feasible Integer Solution for GCP

We introduce the column generation approach that we use to solve the LPrelaxation of the MIS formulation of the GCP in the previous section. From any feasible solution of the LP relaxation we can construct a feasible solution to GCP. In this section, we describe the exact approach that we use to find the best solution for GCP that could be obtained guided by the optimal (most likely fractional) solution of the LP relaxation. Then we introduce some heuristic algorithms that could be used to find a good integral solution using the optimal solution of the relaxed problem.

Exploiting the fractional solution, one might construct different integral feasible solutions to the GCP. The most straight forward approach is by using integer programming. Therefore, after solving the LP relaxation, we remove all columns (independent sets) that are not in the optimal solution from the final restricted master problem. Then we solve it as an integer program. From a computational point of view, this approach might not be a very efficient way to construct an integral solution to the GCP using the fractional solution. However, it will provide us the best solution to the GCP that could be constructed using the fractional solution.

However, for large problems it might be impossible to get the optimal solution within a reasonable computation time. Therefore, we have developed a heuristic algorithm; we refer to it as greedy rounding. The heuristic is computationally efficient and could be used for large problems to get a reasonably good integral solution to the GCP. This greedy rounding constructs integral solutions, guided by the fractional solution from the LP relaxation, using three different heuristic algorithms and provides the best solution as the final feasible integral solution of the GCP. Our three different heuristic algorithms are Heuristic based on Weight, Heuristic based on Number of Cover, Heuristic based on Least Cover.

C. *Solution Strategies for GCP*

Based on our proposed algorithms above, we use a number of different ways to find an integral feasible solution to the GCP:

1. LP-IP: Solve the LP relaxation exactly and construct an integral solution using the exact method.
2. LPapprox-IP: Solve the LP relaxation approximately and construct an integral solution using the exact method.
3. LP-IntHeuristic: Solve the LP relaxation exactly and construct an integral solution using the rounding heuristic.
4. LPapprox-IntHeuristic: Solve the LP relaxation approximately and construct an integral solution using the rounding heuristic.

We do expect that the quality of the integral solution will be better for the approaches listed higher in the list. Those alternative approaches are naturally derived from the fact that we have two challenging problems: MWIS and final IP problems. Approximation of the LP relaxation can be obtained by terminating the column generation approach early.

IV. NUMERICAL RESULTS

We test our approaches on the benchmark data sets which are widely used by many researchers and on some randomly generated data set. All algorithms are implemented and tested using CPLEX12.01 embedded in C++ on a machine with i5 processor, 3.33GHz, and 4 GBRAM. Throughout this paper, the CPU times are reported in seconds. The instances used for the experiments are DIMACS benchmarks instances taken from ftp://dimacs.rutgers.edu/pub/challenge/graph/ and randomly generated instances. The DIMACS benchmark set includes: artificial graphs, register allocation graphs, random graphs, Leighton graphs and flat graph. All of geometric and random graphs are randomly generated using built-in generator from an open source Python package called Networkx. These generators are available at http://networkx.lanl.gov/reference/. The data set can be divided into two groups: easy and hard. We call it as easy instances as these instances are solved exactly by some approaches proposed in literature in reasonable amount of CPU time. There are 171 instances in this group and most of them can be solved within few seconds. We also have 28 hard instances. We call them hard instances as they are hard to solve exactly in a reasonable amount of time and most of the proposed heuristic approaches in the literature use these instances as a benchmark data set.

Applying each solution strategy to each graph, we can draw some conclusion about the average performance of each strategy. Generally, LP-IP strategy produces the best feasible integral solutions among other heuristics, while the LPapprox-intHeuristic solves the problems fastest compared to the other heuristics. The LP-IP and the LPapprox-IPsolve more problems with the same solution quality compared to DSATUR, while the LP-intHeuristic and the LPapprox-intHeuristic strategies solve more problems with the same solution quality, compared to tabu search.

In terms of trade-off between the solution quality and the computational time, though the LP-IP provides feasible solutions with the smallest gaps to the optimal solution, compared to the other heuristics, with a maximum gap of 20%, this heuristic requires much more time to solve the problems on easy instances. The longest time to solve the easy instances is 1396 seconds and the longest computational time to solve the harder instances is almost 7000 seconds. The LPapprox-intHeuristic approach provides feasible solutions with the maximum gap the same as the solution obtainedby the LP-IP for the easy instances, but it produces larger maximum gap for the harder instances, which is 160%, where the LP-IP only has 150% as the maximum gap. However, the computational time spent to solve the GCP on both types of instances are faster compare to the other heuristics, up to 1407 seconds and 4034 seconds, respectively. Although the third approach, the LP-intHeuristic strategy, solves the easy and harder instances much faster compared to the two previous strategies, where the maximum time spent for easy instances is 1019 seconds and for the harder instances is 3847 seconds (half time from LP-IP time), their feasible integral solutions obtained are bit worse, up to 33% for easy instances and 177% for the harder ones. However, the solutions produced by this strategy are much better compared to the solutions obtained by DSATUR which has the maximum gap of 43% and 128%for easy and harder instances, respectively. The LPapprox-intHeuristic solves all instances with the fastest time, where the maximum time spent for easy instances is 73 seconds and for harder instances is 1005 seconds; they are 95% and 86% faster compared to the first strategy, LP-IP strategy. Unfortunately, its solution qualities are much worse among

other proposed approaches and DSATUR, but they are better than the solution obtained by tabu search and Greedy approaches. Our fourth proposed approach, LPapprox-intHeuristic strategy, gives feasible integral solutions with the maximum gap to the optimal solution up to 45% for easy instances and 190% for harder ones. Therefore, DSATUR is better to solve the harder instances compare to the LPapprox-intHeuristic. The two general heuristics, tabu search and Greedy, give similar solution quality but they have the largest maximum gaps to the optimal solution compared to the others, which are about 71% for easy instances and 260% for harder ones.

To summarize our observations on the seven heuristics above, we formulate the following observations. If the solution quality is the only criterion to be taken into account and an efficient exact IP approach is not available, the use of LP-IP strategy or LPapprox-IP strategy for easy instances is advised, while for the harder instances, the use of DSATUR is advised. However, if both the solution quality and computational time are relevant, we strongly recommend the use of LP-intHeuristic, LPapprox-intHeuristic or DSATUR for easy instances and only LP-intHeuristic strategy or LPapprox-intHeuristic strategy for harder ones. Finally we advise not to use tabu search or Greedy for both types of instances unless no other choice is available.

V. Predicting Algorithm Performance Using Data Mining

There are two types of solution approaches proposed and used in this paper: exact and heuristic. Although heuristics can often find a near-optimal solution extremely fast compared to exact approaches, the performance of any heuristic is verydependent on the properties of the instance. Time and complexities can be minimized by knowing which approach to use for a particular instance rather than testing all approaches by trial and error. However, selecting the best algorithm on average across all instances does not help us pick the best one for a particular instance, as supported by Wolpert and Macready's [15] No-Free-Lunch (NFL) theorems. Even though we have the experimental results for each proposed approaches as well the other heuristics mentioned above, we need to understand more about how our proposed approaches perform on different subsets of test instances. We would like to know if there are particular features of instances that make some approaches perform well or fail. Based on the recent research of Smith-Miles and Lopes [9], exploring the featuresof graph coloring instances to determine if there is a particular class of instances where one approach dominates the others is the final contribution of the research. Part of this work, focused on the general purpose heuristics, has already been published [13], and in this paper we extend the work further to learn about the suitability of the exact and heuristics strategies based upon the MIS formulation of graph coloring. We explore different features of instances that have been described before, to determine which of our proposed approaches, LP-IP, LPapprox-IP, LP-intHeuristic and LPapprox-intHeuristic, as well as other general purpose heuristics, DSATUR, tabu search and Greedy, gives a near-optimal solution for particular classes of instances. Here, data mining can help us to gain a good understanding of the relationship between the features of instances and the performance of algorithms; in this case they are our proposed strategies and three general heuristics: DSATUR, tabu search and Greedy.

Graph features are the properties of a graph that depends on its abstract structures. In this research, graph features play important roles as they have correlation with the performance of graph coloring algorithms. The analysis of different features described below has been performed in our recent work [13] by using a Python interface to the open source software called igraph and an open source library of scientific tools called SciPy.

The 16 graph features we used are: the number of vertices in a graph (denoted by n); The number of edges in a graph (denoted by m); The diameter of a graph; The density of a graph; The average path length; The girth; Mean and standard deviation of node degree; The clustering coefficient; The betweenness centrality; The standard deviation of betweenness centrality; The mean of eigenvector centrality; The standard deviation of eigenvector; The algebraic connectivity; and the graph spectrum. Details can be found in [13].

A. Experimental Meta-Data

We present our experimental meta-data using the framework proposedby Rice in 1976 [16], and adapted to the study of optimization algorithm performance by Smith-Miles [12]. There are four essential components in our meta data that are derived from the GCP:

1. The problem space P: a set of 63 easy instances and 28 harder instances, of which 64 instances are DIMACS benchmark instances and 27 instances are geometric and random instances that are randomly generated using Networkx generator.
2. The feature space F: a set of 16 features or characteristics of graph as described above [13].
3. The algorithm space A: consists of four proposed heuristic approaches: LP-IP, LPapprox-IP, LP-intHeuristic and LPapprox-intHeuristic, and three other general purpose heuristics: DSATUR, tabu search and Greedy,
4. The performance space Y: the minimum number of colors found by an approach after a single run.

The Algorithm Selection Problem, defined by Rice [58], requires us to learn the relationship between the feature space F and the algorithm performance space Y for each algorithm in the algorithm space A, by using the instances in the problem space P as training data. We use a combination of supervised and unsupervised data mining techniques to learn these relationships and gain insights in the next section.

B. Visualizing Instance Space

We begin with a self-organizing feature map (SOFM) to visualize the high dimensional feature space as a two dimensional map of the instance space. Here, our instance space comprises a set of 91 instances, where each of these

instances is defined by 16 features that are related to GCP. We apply a preprocessing procedure which involves two transformations. Firstly, the 16 features are transformed using a logarithmic transformation to improve the normality of the distributions. Secondly, by using a scaling transformation, all features are normalized to [0,1]. We used the software package Viscovery SOMine [14] to generate the SOFM using a rectangular map of approximate ratio 100:77, trained with 42 cycles (complete presentations of all 91 instances). Figure 1 displays the distribution of five classes of instances where the 16 dimensional feature space is projected onto a two dimensional map. The size of each cluster is indicative of how many instances belong to each cluster. Two instances are close to each other in the two dimensional map if their 16 dimensional feature vectors is similar according to the Euclidean distance metric. In practice, we label each class shown in Figure 1 as Class 1, Class 2, Class 3, Class 4, and Class 5 for the blue region, pink region, yellow region, dark green region, and light green region, respectively. The distribution of the 91 instances can be seen in Figure 2.

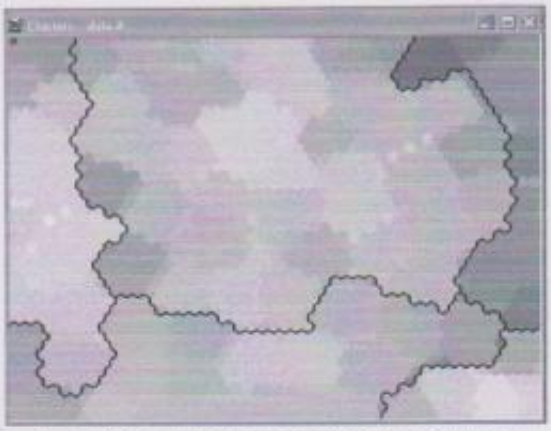

Fig. 1. The distribution of five clusters of instances across instance spaces.

Fig. 2. The distribution of all labelled instances across instance space.

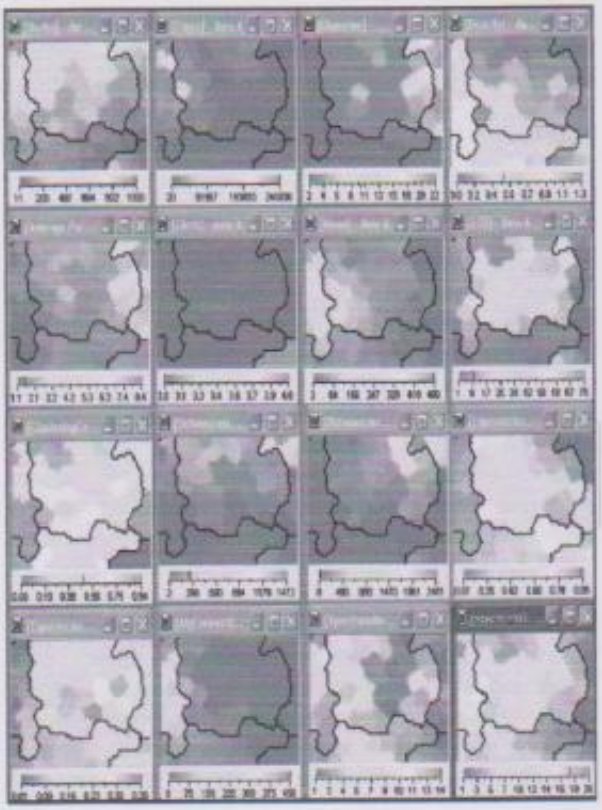

Fig. 3. The distribution of instances features across instace space.

We can determine specific features that are dominant in a class by looking at the distribution of features as seen in Figure 3. The values according to one variable are displayed by the color of the pixel; where warm colors (red, orange, and yellow) represent high values and cool colors (blue) represent low values.

We can see that Cluster 1 contains instances that have a high standard deviation of node degree, a low mean and high standard deviation of eigenvector centrality. Instances with a very low standard deviation of node degree, a low betweenness centrality mean and standard deviation are found in Cluster 2. Cluster 3 contains instances with a medium to high value of mean node degree, a low betweenness centrality standard deviation, high eigenvector centrality mean, a medium to high value of algebraic connectivity, and a medium to high value of spectrum standard deviation. Instances with a low density, a high average path length, and a low spectrum mean are clustered in Cluster 4. We can conclude that the instances that are in Cluster 4 are sparse and close to bipartite since the spectrum mean for bipartite graphs is zero due tothe symmetry of the eigenvalues $\pm\lambda_i$. Cluster 5 contains instances with a high girth and a very low clustering coefficient.

C. Footprints of Algorithm Performance

After the clusters or instance classes are identified, the next step is to examine the algorithm performances across

those instances. Here, the algorithm performances are the performances of our four proposed strategies as well DSATUR, tabu search and Greedy in producing the feasible integral solution, ie. number of colors. The solution gaps to the optimal solution or best known solution (as a percentage) for each algorithm is shown for all instances across the map in Figure 4.

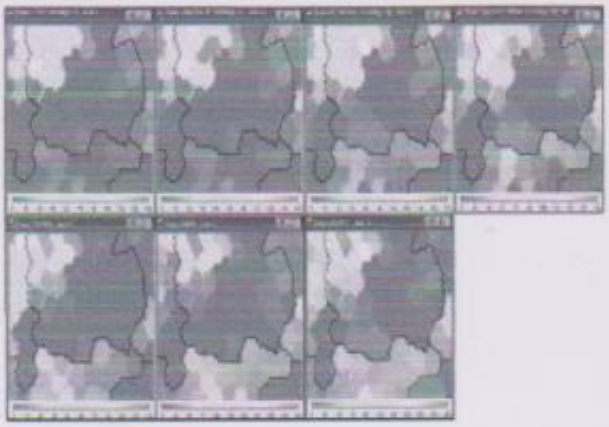

Fig. 4. The performance of each algorithm based on the gap to the optimal solution across instance space. Blue value represent a small optimality gap, while red values represent a poor solution with a high optimality gap.

In Figure 4, the performance of LP-IP, LPapprox-IP, LP-intHeuristic and LPapprox-intHeuristic, are shown in the top map, while the performance of DSATUR, tabu search and Greedy are shown in the bottom map. The warm colors (red, orange, and yellow) represent a high gap between the optimal solution and the solution obtained by each algorithm. Cool colors (blue) represent a low gap between the optimal solution and the solution obtained by each algorithm, thus a very good near-optimal solution. There are many interesting results from this SOFM from which we can draw some conclusions. From the first two maps at the top part, we can see that LP-IP and LPapprox-IP achieve similar performances across most of instances. Both algorithms perform poorly in the half subset of Cluster 3 and small subset of Cluster 1, which are on the top left corner of the maps, while the use of these algorithms are effective for rest of the instances in the map.

There is a small subset of instances from Cluster 1 close to the top left corner where LP-IP and LP-intHeuristic produce a better feasible integral solution rather than those obtained by the LPapprox-IP or the LPapprox-intHeuristic strategies. Among the four proposed heuristics, LP-intHeuristic gives the best feasible solution to those instances, even though the gap is still high. It means that for those instances, it is necessary to solve the LP relaxation exactly. However, DSATUR or tabu search will give the best feasible integral solution among all heuristics for these instances. LP-IP and LPapprox-IP are needed for all instances in Cluster 2 to solve the problem, since the use of LP-intHeuristic and LPapprox-intHeuristic gives slightly worse feasible integral solutions. It means that the final solution of the LP relaxation of these instances needs to be solved as an IP. Likewise DSATUR, tabu search and Greedy will give the worst performances in this instance class. All heuristics give a good performance on the instances in Cluster 5, except for a small pocket in this class where Greedy produced slightly worse integral solutions. If we compare to the Figure 3 in the previous section, we can conclude that all heuristics perform well on instances having small number of nodes, edges, standard deviation, clustering coefficient, betweenness mean, betweenness standard deviation, algebraic connectivity; and medium to high value of eigenvector centrality mean and high girth. DSATUR performs particularly well for the instances in Cluster 4 in the top right, which are close to bipartite, since it is known to be exact for bipartite graphs [1].

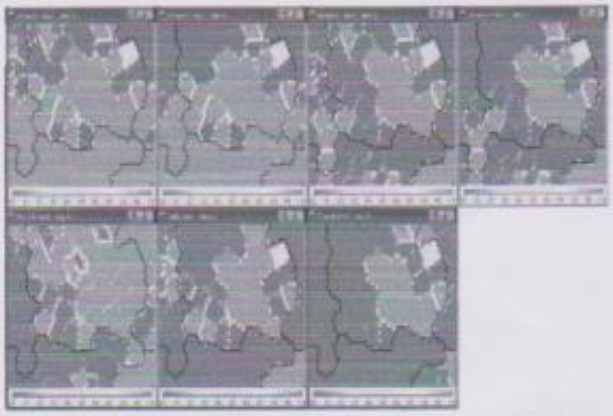

Fig. 5. The distribution of algorithm performance based on their ability to obtain the best solution among other heuristics.

If we convert the optimality gap results into a binary flag to indicate if a given algorithm achieved the best result from the set of 7 considered approaches, then we can show the distribution in Figure 5 of each algorithm's ability to achieve the best performance across the instance space. The warm region shows the algorithm produces the best feasible solution among other 6 algorithms, but it does not mean that this heuristic gave an optimal solution for the problem. Here, the best feasible solution among others is the minimum number of colors that all algorithms can obtain. Blue regions shows the algorithm does not obtain the best solution. The red regions of the instance space show the footprint of each algorithm, a term defined by Corne and Reynolds [16], to show where an algorithm can be expected to perform well. The footprints for LP-IP, LPapprox-IP, LP-intHeuristic and LPapprox-intHeuristic, are shown in the top map, while the footprints for DSATUR, tabu search and Greedy are shown in the bottom map. As seen in Figure 5, the LP-IP and the LPapprox-IP strategies, give the best solutions for most of the instances, where tabu search gives the least number of best solutions among other algorithms and has the smallest footprint. Most of instances in Class 2, also some instances in Class 1 and 3 obtain the best solution when we construct the integral solution exactly while the greedy heuristic is failed to give the best solution. If we take a look closer to Figure 3, those

instances have medium value of density and medium to high value of clustering coefficient. Thus, constructing the integral solution using an exact method is necessary when the graph's density is high as well as if it is well-clustered. We can also see the superiority of DSATUR in the top right corner where DSATUR can produce the best integral solution, while the others cannot. From Figure 3, we can conclude that DSATUR gives the best solution for instances with a high number of vertices, a high number of edges, a high mean, a high algebraic connectivity and a high spectrum mean. In other words, a large non-bipartite graph which is very well connected will obtain the best solution using DSATUR. There is also a small pocket at the bottom of Cluster 1 (in the middle of the map), where LP-IP and LPapprox-IP, tabu search obtains the best solution, while the other algorithms fail. Unfortunately, we cannot observe the specific instance features that make it different from others. For all instances in Cluster 5, any algorithms will give the best solution, except Greedy. Similar to with our previous work [13] focused on the general purpose heuristics; here SOFM is also able to show clear regions where the performance of algorithms is similar. It should be reiterated that we do not use any algorithm performance data as inputs to the clustering process of SOFM, but only superimpose performance data after the clusters have formed. The fact that two instances that are close to each other in feature space and also have similar algorithm performance results (shown by the continuity of the footprints) means that the features that we chose are well suited to characterize the similarities between the instances, and the properties of these features clearly affect the performance of algorithms.

ACKNOWLEDGMENT

The authors would like to thank to Dr. Leonardo Lopes and Brendan Wreford who implemented the codes to generate the instances features.

REFERENCES

[1] D. Brelaz, New methods to color the vertices of a graph, Communications of the ACM, vol. 22, no. 4, p. 251, 1979.

[2] E. Burke, J. Marecek, A. Parkes, and H. Rudova, A supernodal formulation of vertex colouring with its applications in course timetabling, Annals of operations research, vol. 179, no. 1, pp. 105-130, 2007.

[3] J. Culberson, Graph coloring page, URL: http://www. cs. ualberta.ca/joe/Coloring, 2006.

[4] A. Hertz and D. Werra, Using tabu search techniques for graph coloring, Computing, vol. 39, no. 4, pp. 345-351, 1987.

[5] A. Mehrotra, A column generation approach for graph coloring, INFORMS Journal on Computing, vol. 8, no. 4, p. 344, 1996.

[6] A. Mehrotra and M. A. Trick, A branch and price approach for graph multicoloring, in Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, ser. Operations Research/Computer Science Interfaces Series, E. K. Baker, A. Joseph, A. Mehrotra, M. A.Trick, R. Sharda, and S. Voh, Eds. Springer US, 2007, vol. 37, pp.15-29.

[7] J. R. Rice, The algorithm selection problem, ser. Advances in Computers, M. Rubino and M. C. Yovits, Eds. Elsevier, 1976, vol. 15, pp. 65 - 118. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0065245808605203

[8] K. Smith-Miles, R. James, J. Giffin, and Y. Tu, Understanding the relationship between scheduling problem structure and heuristic performance using knowledge discovery, lncs, Learning and Intelligent OptimizatioN, LION, vol. 3, 2009.

[9] K. Smith-Miles and L. Lopes, Measuring instance difficulty for combinatorial optimization problems, Computers & Operations Research, vol. 39, no. 5, pp. 875-889, 2012.

[10] K. Smith-Miles, J. van Hemert, and X. Lim, Understanding tsp difficulty by learning from evolved instances, in Learning and Intelligent Optimization: 4th International Conference, Lion 4, Venice, Italy, January 2010.Selected Papers, vol. 6073. Springer-Verlag New York Inc, 2010, p. 266.

[11] K. Smith-Miles, Towards insightful algorithm selection for optimisation using meta-learning concepts, in Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. IEEE, 2008, pp. 4118-4124.

[12] K. A. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv., vol. 41, no. 1, pp. 1-25, 2008.

[13] Smith-Miles, K., B. Wreford, L. Lopes and N. Insani, Predicting metaheuristic performance on a graph coloring problems using data mining, in Hybrid Metaheuristics, E.G. Talbi, Ed. Springer, 2013.

[14] V. SOMine, Eudaptics software gmbh.

[15] D. H.Wolpert and W. G. Macready, No free lunch theorems for optimization, Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, pp. 67-82, 1997.