Cliques to Colors

On Using Clique Statistics to Analyze Heuristic Solvers for the Graph Coloring Problem

Pawan Paleja
University of California, San Diego
A15586236

ABSTRACT

In the realm of solving NP-Hard problems, heuristics are often used as a black-box solution. However analysis of these heuristics proves almost as difficult as the problem instances themselves. Indeed, though certain heuristics are known or conjectured to perform better in certain situations, there is much to learn as to why. In this paper, we look at one avenue of gaining insight into the behavior of state-of-the-art solvers for the Graph Coloring Problem: clique. We reveal findings involving the variation, entropy, frequency, and size of cliques in a given graph influencing the performance and solution quality of specific heuristics and provide potential insight into the differentiating factors between the given solvers.

KEYWORDS

Graph Coloring, Algorithm Selection, Metaheuristics, Clique

1 INTRODUCTION

1.1 The Graph Coloring Problem

The Graph Coloring problem is one of the oldest, classic NP-Hard problem. The problem setup is deceptively simple: Given a graph, G, with vertex and edge set V, E respectively, a *coloring* of the graph is labeling $c : V \longrightarrow \mathbb{N}$ of all the vertices of the graph such that $c(u) \neq c(v)$ if $(u, v) \in E$ (i.e. if u, v are adjacent in G). While finding any such coloring is trivial (just label each vertex a new color), the hardness comes in trying to minimize the number of distinct labels needed to color the graph accordingly. In his seminal paper, Karp showed that the Graph Coloring Problem (GCP) was an NP-Complete problem [5], meaning that there is currently no known algorithm that can guarantee an optimal solution in an feasible amount of time for all or most instances. However, this is not the end of the story, as over the years, high-performance heuristic algorithms have been developed to tackle the GCP, offering strong empirical results on a variety of graphs [7][8]. However, this same empirical analysis also shows that there are large pockets across the given instance space in which there is high amounts of variance in the performance of any given heuristic [13]. This, coupled with ramifications of the 'No Free-Lunch' Theorems [14], specifically that any areas of high performance of a heuristic algorithm necessarily implies the existence of areas of low performance, it seems that there is no one heuristic algorithm that is the 'best' for all instances. Rather, some Heuristic A might fare better than another Heuristic B depending on the given instance.

1.2 Algorithm Selection

In noticing this trend, researchers approached developing a solution through the lens of Algorithm Selection. This problem, initially posed by Rice in [10] is described like so: Given an instance I for some problem and a space of algorithms A to solve that problem, predict the algorithm, $a \in A$ which will perform the best (using some metric of performance) on a given $i \in I$ using some features mined from the given instance. Using this paradigm, a team developed a classifier to do just that for the SATISFIABILITY problem, [15], another classic NP-Hard problem, and reported massive speed and solution quality gains. Perhaps inspired by this success, this method was co-opted for the graph coloring algorithm, in [9], where 6 state-of-the-art algorithms were selected to compose the set of algorithms to predict from. While they were able to report being able to create a predictor from these features which was able to generate the best known solution with significantly higher percentage than any one heuristic in their portfolio, they did not consider the predictive capacity of the specific features classes. In the first portion of this paper, I will be looking at the predictive capacity of one of the classes of graph metrics used by Musliu et al, clique, to understand the importance of their presence and distribution on the runtime and optimality of state-of-the-art GCP heuristics.

1.3 Related Work

Given the age of the GCP and Algorithm Selection, there has been an extensive amount of research into the analysis of graph features that are associated with instances hard for heuristics to solve. In particular, the variation in degrees as well as the 'equi-partiteness' of the graphs are especially associated with the hardest instances of GCP [11]. Additionally, in a review of heuristic algorithms, [2] found that the greedy algorithm DSATUR[6][1] performs well when the clique number of the graph is relatively close to the chromatic number. DSATUR has also been found to perform highly well on geometric graphs which is conjectured to be due to the presence of large cliques in these s (which can drive the chromatic number) and DSATUR's tendency to color denser regions first[3]. Analysis has also been done on Tabu Search, which was been found to outperform DSATUR on IID graphs, while DSATUR performed better on cycle-driven graphs [12].

2 OUR ALGORITHM SELECTION TASK

2.1 Instance Space I

2.1.1 For Predictor. We will be reusing the dataset composed by [9]. In it are selections from the DIMACS challenge for the GCP[1] as well as 447 order-500 and 664 order-1000 graphs [2] generated

[1]https://mat.tepper.cmu.edu/COLOR/instances.html
[2]www.imada.sdu.dk/ marco/gcp-study/

CSE 291, Fall 2020, Final Project

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/

using Culberson's GCP instance generator representing 3 classes of graph types by edge distribution (IID, Geometric, and Weight biased). These were chosen to provide a representative comparison with the results of Musliu et al. A quarter of these graphs were hidden for evaluation of the classifier, while the rest were used to train it.

2.1.2 For Empirical Analysis. We will also be analyzing separately graphs generated ourselves using the same generator. Specifically, we experiment with the presence of large cliques in the graph, increasing their number while keeping the chromatic number fixed from an upper bound. We use two clique sizes - 5 and 20 - the former representing a baseline as it is close to the size of cliques that should be common in the graph ($log(|V|)$) and the latter representing a much larger clique. Culberson's generator places a parameterized number of cliques between nodes that are randomly chosen but guaranteed to not disrupt the chromatic number, so we range from inputting 1 of these cliques of the given size to 300 (though placing this many might not be possible). We chose a chromatic number of 20, to match the size of the largest clique (as $\chi(G) \geq \omega(G)$), however note that Culberson does not guarantee that the chromatic number be exactly 20, only that it is at most 20.

2.2 Algorithm Space A

The algorithms being used are from the source code[3] developed for Lewis et al's review of high-performing graph algorithms [7]. For more detailed descriptions, please refer to that paper. All algorithms were used with default parameters.

2.2.1 TabuCol. Tabu search is a local search metaheurisitic, specifically adapted to the GCP. Like other local search methods, it is characterized by first finding a random coloring and then improving upon it through changing the color of any one vertex that minimizes the some cost function related to the quality of the coloring (like the number of conflicts a particular coloring gives). However, to allow escape from local optima, the heuristic allows for backtracking to occur. However it keeps a list of solution called a 'tabu' list of solution states that are banned for a specific amount of iterations.

2.2.2 Hybrid Evolutionary Algorithm (HybridEA). Evolutionary Algorithms start with a population of initial solutions and then iteratively improving that population by combining two 'parent' solutions to get a (hopefully better) child solution. [4] improved upon this philosophy by inserting a step in which local search is performed after the recombination step (in this case a tabu search is the local search method).

2.2.3 AntCol. Similar to HybridEA, the AntCol algorithm uses a population of 'ants' representing different solutions based both on some heuristic, but is also influenced towards similar colorings to previous generations of ants by a pre-determined 'pheromone' parameter.

2.2.4 HillClimbing. HillClimbing starts with an initial solution given by the DSATUR heuristic, and iteratively moving vertices from one given color class to another. Then, to allow for more subsequent iterations of this (again to escape local minima), vertices are randomly selected to be recolored using Kemp chains.

2.2.5 BacktrackingDSATUR. DSATUR is a heuristic for coloring vertices according to the order that colors the most constrained (i.e. 'saturated') vertices first. Introducing a backtracking steps allows for other colorings for a given vertex to be considered as well, to see if they give better solutions that strict DSATUR. The search tree for backtracking is reduced from its actual size for an exact algorithm to allow for efficiency.

2.3 Feature Space F

As stated above, we will primarily be looking at one class of features - clique - pulled directly from [9], but which also reflect the information seen in the above related works. For clique statistics, a greedy method was implemented (add a vertex to the clique if it is connected to all the others in the current clique) which attempted to generate as many distinct cliques as possible in a 10-second period by starting at a different vertex each time, while for degrees, the degree sequence was simply parsed from the adjacency list. For the clique sets parsed, the minimum, maximum, mean, median, variance, entropy and computation time were calculated and stored as features. Additionally, the min and max were normalized by the largest clique, though the raw maximum were also stored.

2.4 Performance Space Y

[9] measures success of a given algorithm by the closeness of the returned chromatic number to the true chromatic number of a given graph. However, there was no clear way described with which to achieve this, so borrowing and adapting from Xu et al's metrics for SAT[15], we look at 2 metrics: runtime complexity (measured in seconds and in data structure queries) and the number of colors found.

2.5 Classification Model

Musliu et al[9] and Lewis et al[12] both found that a Random Forest of decision trees best classified the data for the algorithm selection problem. However, they were classifying predicting non-categorical data, while both of the above performance metrics are continuous, so we use a Random Forest Regression model from the Sci-kit library[4].

3 RESULTS

Given that we are looking at the predictive capacity of certain metrics on predicting performance, we will look at each algorithm in our portfolio separately. Note that the Simple Greedy method was included for a baseline.

3.1 Predictor Score

As we can see from Figure 1, cliques clearly drive the chromatic number of the graph, with the classifier achieving an R^2 of over 90% for all algorithms. However, the values do not appear different enough from the baseline to be significant. For the runtime metrics, raw computation time seems 'easier' for the classifier to predict

[3]http://www.rhydlewis.eu/resources/gCol.zip

[4]https://scikit-learn.org/stable/modules/ensemble.htmlforest

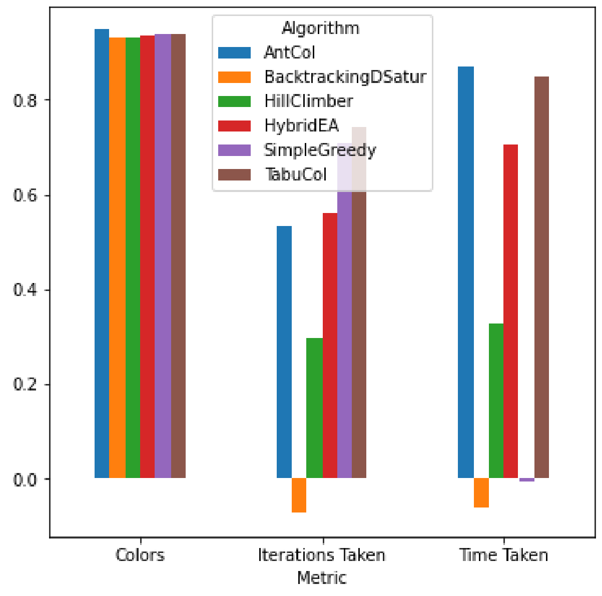


Figure 1: R^2 score for algorithms using clique statistics

across all algorithms except the baseline. Of these, only for HillClimber and BacktrackingDSATUR is the majority of the variance not explained. For those two, it is less than surprising, because DSATUR is a greedy algorithm that is only aided by the order in which it prioritizes vertices. For HillClimber, from [12] much of the runtime comes from the recoloring with Kemp chains step, which is unclear to be correleated to clique statistics.

3.2 Predictor Feature Importances

We measure feature importance by permuting a specific feature (turning it into noise) and taking the R^2 score of the samples with the permuted feature. We do this 10 times and average the scores to get $\hat{r}$. Then, the original score r minus $\hat{r}$ gives us the importance of the feature. Note that we only include results for which a given feature loses a majority of its importance for some algorithm and metric, meaning we disinclude the normalized min, max, and median features.

3.2.1 Variance. Clique variance specifically seems not to be an important feature for most of the algorithms when it comes to the number of colors found, however for predicting runtime complexity for TabuCol and AntCol it is much more important. It is possible that the large number of cliques overconstrains the problem instance meaning that reducing the number of conflicts of previous solutions (as TabuCol and AntCol both do) could lead to a long time spent per search step. This might explain why HillClimber also has a (relatively) high importance for this feature as well.

3.2.2 Entropy. Two things seem to be of interest. Again, it seems to have little influence on the quality of the coloring found, and for all but AntCol it seems to have little importance to the predictor. However, for AntCol specifically, entropy seems to be extremely important to the runtime complexity, especially given from Figure 1, our predictor was very effective at predicting the runtime of AntCol. Interestingly, permuting entropy actually increased the score of BacktrackingDSATUR, however from Figure 1 we see that this is a bit misleading given that the runtime predictions were already very poor to begin with.

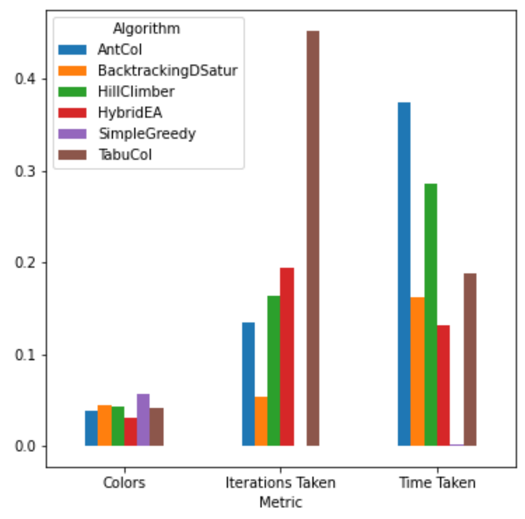


Figure 2: Mean score decrease from permuting Clique Var feature

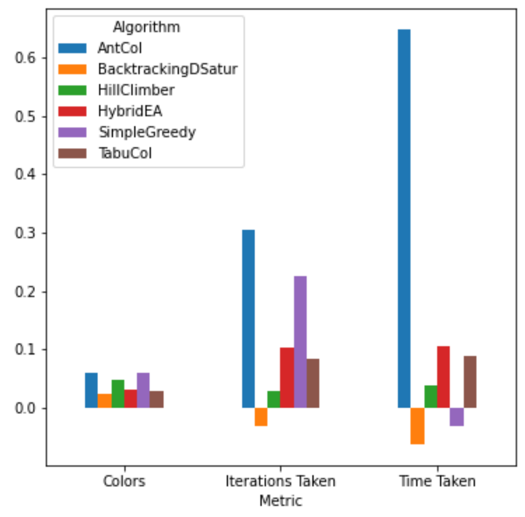


Figure 3: Mean score decrease from permuting Clique Entropy feature

3.2.3 Computation Time. We see from Figure **??** that again for most of the algorithms, this feature is not so important. But, specifically for the runtime of HybridEA, it is extremely important. This is quite significant, given both the discrepancy from the other algorithms, but also given that the classifier was quite accurate at predicting the time taken by HybridEA in particular. Additionally, HybridEA is considered the 'best' heuristic available for the average instance of the GCP [9, 12], meaning there is importance to understanding what sets it apart.

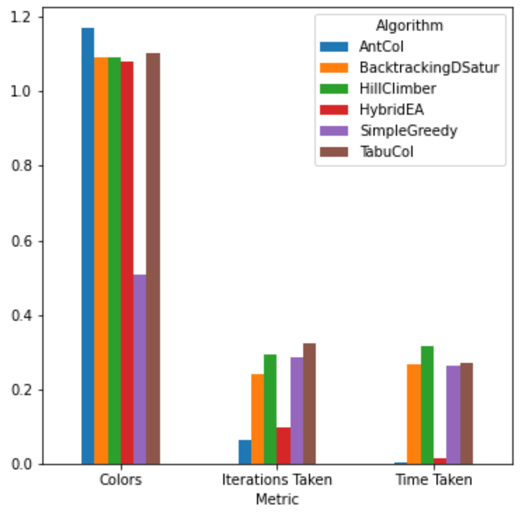


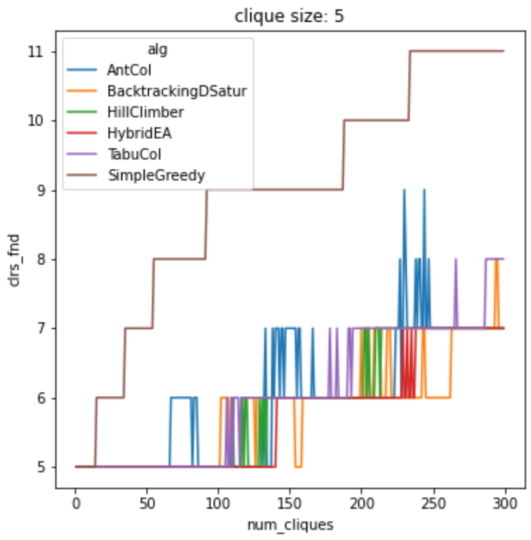
Figure 4: Mean score decrease from permuting Clique Raw Max feature

3.2.4 Raw Max. Finally, as seen in Figure 4 the cardinality of the largest clique found was the most important factor by far for all the algorithms in predicting the number of colors. This is unsurprising and supports the notion that the clique number of a graph drives its chromatic number. However, interestingly, it seems to not be important at all for the runtime complexity of the heuristics. This is likely because other features are much more descriptive for that facet than just the max clique.

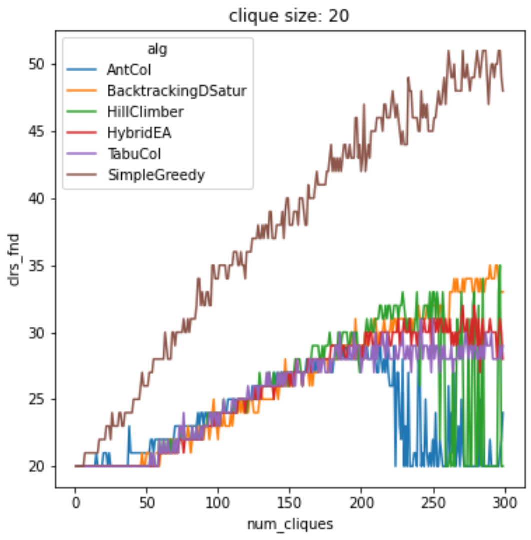
3.3 Analysis on Clique-Driven Graphs

3.3.1 Colors Found. For graphs with the baseline clique size (essentially random graphs), we see in Figure 5 that the heuristics (other than the baseline Simple Greedy) find similar colorings to each other (plus/minus some small discrepancy). This is almost true again, for when the clique size is 20, however, there is an interesting phase transition at around 200 cliques being added in which we see that AntCol and HillClimber both start finding much better solutions than their counterparts, and find solutions very close to optimal.

3.3.2 Runtime. From Figures 7 and 6, most of the algorithms seem to follow the same trend of an increasing runtime with the number of cliques. However interestingly again, there seems to be the same phase transitions seen in 3.3.1 with AntCol and HillClimber. Thus, we could infer that the reason why both these heuristics are able to find better colorings than their counterparts is because they were able to search in spaces where they could better improve their current solution state, which is markedly significant.

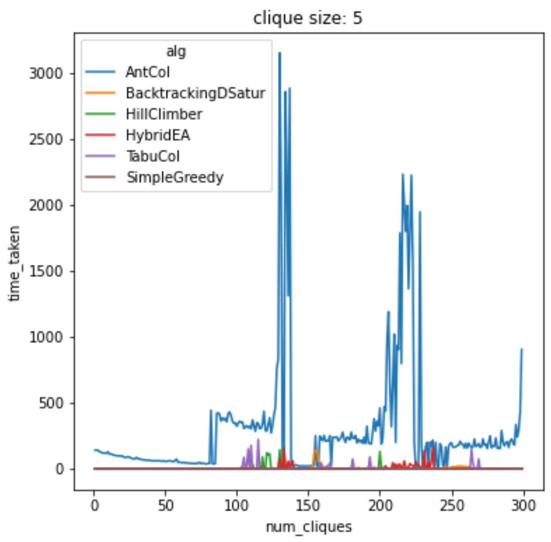


(a)

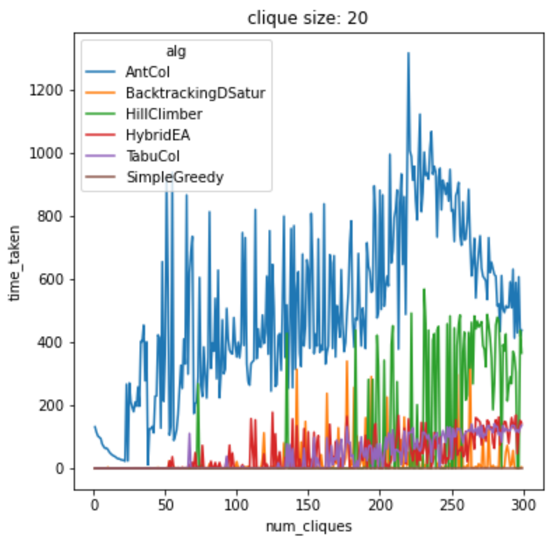


(b)

Figure 5: Colors Found vs Number of Cliques

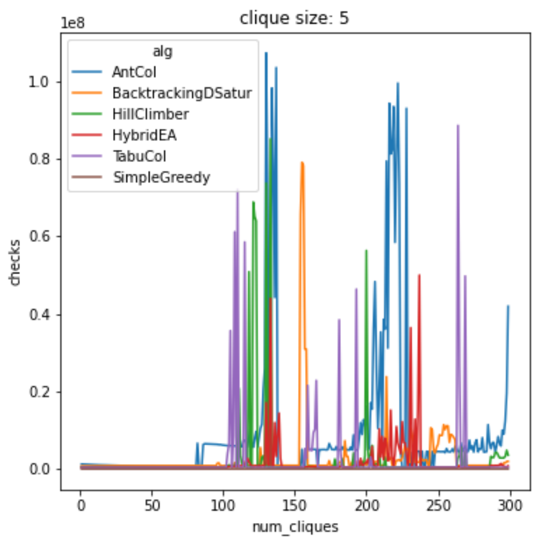


(a)

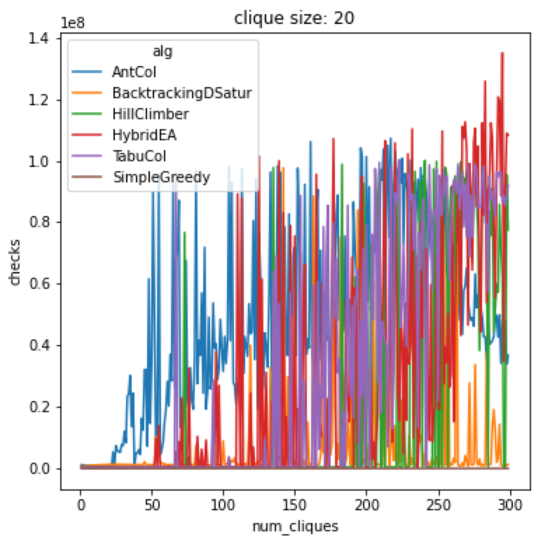


(b)

Figure 6: Time Taken vs Number of Cliques



(a)



(b)

Figure 7: Number of Checks vs Number of Cliques

4 CONCLUSION

In this paper, we analyzed the capacity of certain clique statistics to predict the performance and solution quality of state-of-the-art heuristics for the graph coloring problem. We saw that there are certain features that directly influence the performance of only one or two specific heuristics while leaving the rest unchanged (e.g. variance, entropy, or frequency of cliques in the graph). This is provides avenues of exploration in which the optimal conditions of heuristics can be more rigorously analyzed to improve upon them and better understand the GCP. For example, AntCol and HybridEA are both population-based heuristics, however AntCol experiences a phase transition in which it can improve on solutions for longer than HybridEA and thus return a better coloring. I believe it is likely due to the difference in recombination, because while HybridEA's children takes characteristics from its parents, a given solution in an AntCol iteration is influenced by the entire colony. It would be interesting to explore this more, both to see if more can be proven about the relationship between cliques and search space for the GCP and for insight into the functioning of the heuristics we take for granted.

REFERENCES

[1] Daniel Brélaz. "New methods to color the vertices of a graph". In: *Communications of the ACM* 22.4 (Apr. 1979), pp. 251–256. ISSN: 0001-0782. DOI: 10.1145/359094.359101. URL: https://doi.org/10.1145/359094.359101 (visited on 11/02/2020).

[2] Marco Chiarandini and Thomas Stützle. "An Analysis of Heuristics for Vertex Colouring". In: Jan. 1970, pp. 326–337. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_28.

[3] Joseph Culberson and Adam Beacham. "Hiding our Colors". In: *In CP'95 Workshop on Studying and Solving Really Hard Problems*. 1995, pp. 31–42.

[4] Philippe Galinier and Jin-Kao Hao. "Hybrid Evolutionary Algorithms for Graph Coloring". In: *Journal of Combinatorial Optimization* 3.4 (Dec. 1999), pp. 379–397. ISSN: 1573-2886. DOI: 10.1023/A:1009823419804. URL: https://doi.org/10.1023/A:1009823419804.

[5] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[6] Samuel M Korman. "The graph-colouring problem". In: *Combinatorial optimization* (1979). Publisher: Wiley, New York, pp. 211–235.

[7] R. Lewis et al. "A wide-ranging computational comparison of high-performance graph colouring algorithms". en. In: *Computers & Operations Research* 39.9 (Sept. 2012), pp. 1933–1950. ISSN: 0305-0548. DOI: 10.1016/j.cor.2011.08.010. URL: http://www.sciencedirect.com/science/article/pii/S0305054811002425 (visited on 11/02/2020).

[8] Rhyd Lewis. "A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing". en. In: *Computers & Operations Research* 36.7 (July 2009), pp. 2295–2310. ISSN: 0305-0548. DOI: 10.1016/j.cor.2008.09.004. URL: http://www.sciencedirect.com/science/article/pii/S030505480800169X (visited on 11/02/2020).

[9] Nysret Musliu and Martin Schwengerer. "Algorithm Selection for the Graph Coloring Problem". In: *Learning and Intelligent Optimization*. Ed. by Giuseppe Nicosia and Panos Pardalos. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 389–403. ISBN: 978-3-642-44973-4.

[10] John R. Rice. "The Algorithm Selection Problem**This work was partially supported by the National Science Foundation through Grant GP-32940X. This chapter was presented as the George E. Forsythe Memorial Lecture at the Computer Science Conference, February 19, 1975, Washington, D. C." In: ed. by Morris Rubinoff and Marshall C. Yovits. Vol. 15. Advances in Computers. Elsevier, 1976, pp. 65–118. DOI: https://doi.org/10.1016/S0065-2458(08)60520-3. URL: http://www.sciencedirect.com/science/article/pii/S0065245808605203.

[11] Kate Smith-Miles and Leo Lopes. "Measuring instance difficulty for combinatorial optimization problems". In: *Computers & Operations Research* 39.5 (2012), pp. 875–889. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2011.07.006. URL: http://www.sciencedirect.com/science/article/pii/S0305054811001997.

[12] Kate Smith-Miles et al. "Predicting Metaheuristic Performance on Graph Coloring Problems Using Data Mining". In: *Hybrid Metaheuristics*. Ed. by El-Ghazali Talbi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 417–432. ISBN: 978-3-642-30671-6. DOI: 10.1007/978-3-642-30671-6_16. URL: https://doi.org/10.1007/978-3-642-30671-6_16.

[13] Kate Smith-Miles et al. "Towards objective measures of algorithm performance across instance space". In: *Computers Operations Research* 45 (May 2014), pp. 12–24. DOI: 10.1016/j.cor.2013.11.015.

[14] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893.

[15] Lin Xu et al. "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *Journal of Artificial Intelligence Research* 32 (2008), pp. 565–606. DOI: 10.1613/jair.2490.