International Journal of Research
Available at http://internationaljournalofresearch.org/
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 02 Issue 04
April 2015

Study About the Selection Algo for the Graph Coloring Problem Method

Abhinavsharma&Tamannakundra

Abstract

We present an automated algorithm selection method based on machine learning for the graph coloring problem (GCP). For this purpose, we identify 78 features for this problem and evaluate the per-formance of six state-of-the-art (meta)heuristics for the GCP. We use the obtained data to train several classication algorithms that are applied to predict on a new instance the algorithm with the highest expected performance. To achieve better performance for the machine learning al-gorithms, we investigate the impact of parameters, and evaluate di erent data discretization and feature selection methods. Finally, we evaluate our approach, which exploits the existing GCP techniques and the auto-mated algorithm selection, and compare it with existing heuristic algo-rithms. Experimental results show that the GCP solver based on machine learning outperforms previous methods on benchmark instances.

Keywords: Algorithm Selection, Graph Coloring, Machine Learning

Introduction

Many heuristic algorithms have been developed to solve combinatorial optimiza-tion problems. Usually, such techniques show di erent behaviorwhen solving particular instances. According to the no free lunch theorems [44], no algorithm can dominate all other techniques on each problem. In practice, this raises new issues, as selecting the best (or most appropriate) solver for a particular in-stance may be challenging. Often, the \winner-take-all" strategy is applied and the algorithm with the best average performance is chosen to solve all instances. However, this methodology has its drawbacks, because the distribution of tested instances e ects the average performance, and usually in practice only a special class of instances are solved.

One possible approach to obtain better solutions on average is to select for each particular instance the algorithm with the highest expectedperformance. This task is known as algorithm selection (AS) and one emerging and very promising approach that is used for AS is based on machine learning methods. These techniques are able to learn a model based on previous observations an then predict on a new and unseen instance the best algorithm.

In this paper, we address AS using classication algorithms for the well-known Graph Coloring Problem (GCP). The GCP is a classical NP-hard prob-lem in computer science. The task for this problem is to assign a color to each node of a given graph such that (a) no adjacent nodes received the same color and (b) the number of colors used is minimized. Various heuristic algorithms to solve GCP have been developed in the literature. However, recent studies [7,26] show that the performance of di erent heuristics highly depend on attributes of the graph like for example the density or the size. Therefore, the aim of this paper is to apply automated algorithm selection for this problem. We evaluate experimentally di erent heuristics and classication algorithms and show that our solver that includes algorithm selection is able to achieve much better per-formance than the underlying heuristics.

The rest of this paper is organized as follows: Section 2 gives a short introduc-tion into the GCP, AS and the related work. In Section 3, we present features of a GCP instance and describe our AS approach for the GCP. The experimental results are given in Section 4 while Section 5 concludes our work and describes the future work.

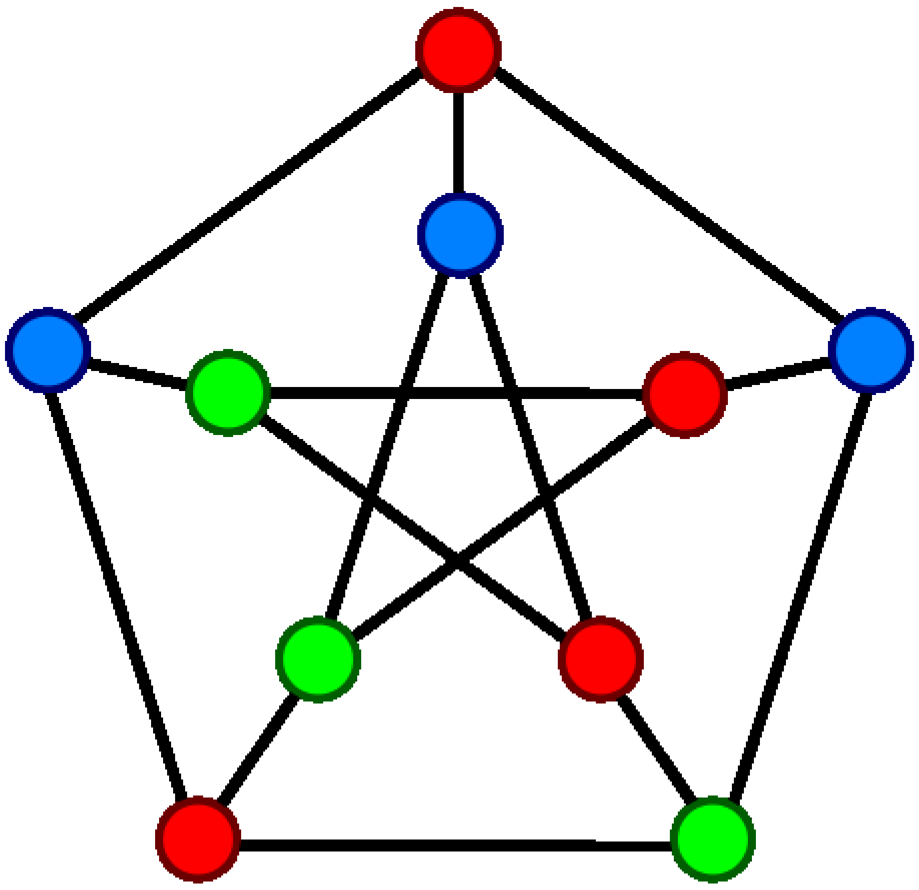
Graph Coloring Problem(Definition and Terminology)

In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges share the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a plane graph is just a vertex coloring of its dual. However, non-vertex coloring problems are often stated and studied as is. That is partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a graph embedded in the plane. By planar duality it became coloring the vertices, and in this form it generalizes to all graphs. In mathematical and computer representations, it is typical to use the first few positive or nonnegative integers as the "colors". In general, one can use any finite set as the "color set". The nature of the coloring problem depends on the number of colors but not on what they are.

Graph coloring enjoys many practical applications as well as theoretical challenges. Beside the classical types of problems, different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself. It has even reached popularity with the general public in the form of the popular number puzzle Sudoku. Graph coloring is still a very active field of research.



A proper vertex coloring of the Petersen graph with 3 colors, the minimum number possible.

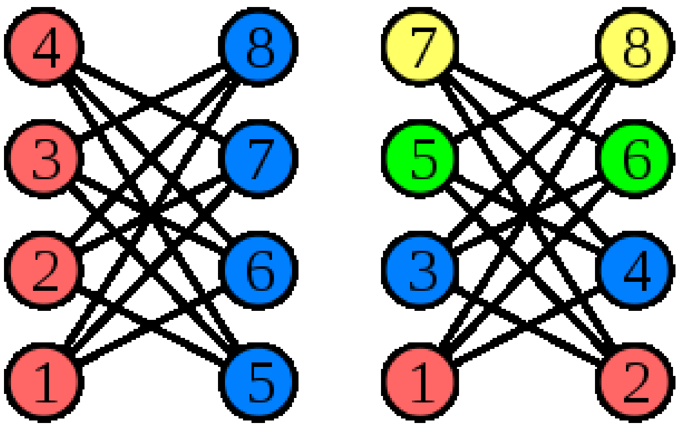
Greedy coloring

Two greedy colorings of the same graph using different vertex orders. The right example generalises to 2-colorable graphs with *n* vertices, where the greedy algorithm expends colors.

The greedy algorithm considers the vertices in a specific order $v_1,\ldots,v_n$ and assigns to v_i the smallest available color not used by v_i's neighbours among $v_1,\ldots,v_{i-1}$, adding a fresh color if needed. The quality of the resulting coloring depends on the chosen ordering. There exists an ordering that leads to a greedy coloring with the optimal number of $\chi(G)$ colors. On the other hand, greedy colorings can be arbitrarily bad; for example, the crown graph on n vertices can be 2-colored, but has an ordering that leads to a greedy coloring with $n/2$ colors.

For chordal graphs, and for special cases of chordal graphs such as interval graphs and indifference graphs, the greedy coloring algorithm can be used to find optimal colorings in polynomial time, by choosing the vertex ordering to be the reverse of a perfect elimination ordering for the graph. The perfectly orderable graphs generalize this property, but it is NP-hard to find a perfect ordering of these graphs.

If the vertices are ordered according to their degrees, the resulting greedy coloring uses at most $\max_i \min\{d(x_i) + 1, i\}$ colors, at most one more than the graph's maximum degree. This heuristic is sometimes called the Welsh–Powell algorithm.[11] Another heuristic due to Brélaz establishes the ordering dynamically while the algorithm proceeds, choosing next the vertex adjacent to the largest number of different colors.[12] Many other graph coloring heuristics are similarly based on greedy coloring for a specific static or dynamic strategy of ordering the vertices, these algorithms are sometimes calledsequential coloring algorithms.



Chromatic polynomial

All nonisomorphic graphs on 3 vertices and their chromatic polynomials. The empty graph $E3$ (red) admits a 1-coloring, the others admit no such colorings. The green graph admits 12 colorings with 3 colors.

The chromatic polynomial counts the number of ways a graph can be colored using no more than a given number of colors. For example, using three colors, the graph in the image to the right can be colored in 12 ways. With only two colors, it cannot be colored at all. With four colors, it can be colored in $24 + 4\cdot12 = 72$ ways: using all four colors, there are $4! = 24$ valid colorings (every assignment of four colors to any 4-vertex graph is a proper coloring); and for every choice of three of the four colors, there are 12 valid 3-colorings. So, for the graph in the example, a table of the number of valid colorings would start like this:

Available colors	1	2	3	4	…
Number of colorings	0	0	12	72	…

The chromatic polynomial is a function $P(G, t)$ that counts the number of t-colorings of G. As the name indicates, for a given $G$the function is indeed a polynomial in t. For the example graph, $P(G, t) = t(t − 1)^2(t − 2)$, and indeed $P(G, 4) = 72$.

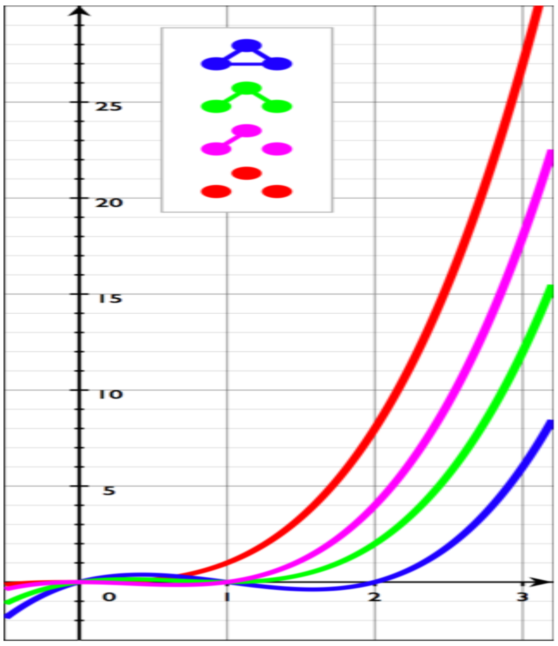
The chromatic polynomial includes at least as much information about the colorability of G as does the chromatic number. Indeed, χ is the smallest positive integer that is not a root of the chromatic polynomial

$$\chi(G) = \min\{k : P(G,k) > 0\}.$$

Chromatic polynomials for certain graphs

Triangle K_3	$t(t-1)(t-2)$
Complete	$t(t-1)(t-2)\cdots(t-(n-1))$

graph K_n	
Tree with n ver tices	$t(t-1)^{n-1}$
Cycle C_n	$(t-1)^n+(-1)^n(t-1)$
Petersen graph	$t(t-1)(t-2)(t^7-12t^6+67t^5-230t^4+529t^3-814t^2$



Conclusion

In this paper, we presented a novel approach based on machine learning to automate algorithm selection for the GCP. Given a set of algorithms and a setof speci c features of a particular instance, such a system selects the algorithm which is predicted to show the best performance on that instance. Our proposed approach applies a classication algorithm as selection procedure that assigns a new instance to one of the available algorithms based on a previously learned model. For this purpose, we identied 78 attributes for the GCP that can be calculated in reasonable time and that have impact on solving of the GCP.
To demonstrate our approach, we evaluated the performance of six state-of-the-art (meta)heuristics on three publicly available sets of instances and showed that no algorithm is dominant on all instances. We further applied machine learning to build an automated selection procedure based on the obtained data. For that purpose, we experimented with six well known classication algorithms that are used to predict on a new instance the most appropriate algorithm. Our experiments clearly showed that a solver that applies machine learning yield a signicant better performance compared with any single heuristic. We further demonstrated that using data discretization increases the accuracy of most classiers.

Regarding future work, we plan to investigate a regression-based approach using runtime and solution quality predictions. This technique, which is suc-cessfully used for other systems, is an alternative to our classication-based ap-proach. Also worth considering is a hybridization of our method with automated parameter selection and the combination of heuristic and exact techniques for the GCP in a system that applies automated algorithm selection.

References

- I. Bl•ochliger and N. Zuerey. A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & Operations Research, 35(3):960{975, 2008.
- R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. Weka manual (3.6.6), Oct. 2011.
- D. Brelaz. New methods to color the vertices of a graph. Commun. ACM, 22:251{ 256, Apr. 1979.

- K. L. Brown, E. Nudelman, and Y. Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. J. ACM, 56(4):1{52, 2009.
- G. Chaitin. Register allocation and spilling via graph coloring. SIGPLAN Not., 39(4):66{74, Apr. 2004.
- M. Chiarandini. Stochastic Local Search Methods for Highly Constrained Combi-natorialOptimisation Problems. PhD thesis, TU Darmstadt, Aug. 2005.

- Barenboim, L.; Elkin, M. (2009), "Distributed ($\Delta + 1$)-coloring in linear (in Δ) time",Proceedings of the 41st Symposium on Theory of Computing, pp. 111–120,doi:10.1145/1536414.1536432, ISBN 978-1-60558-506-2
- Panconesi, A.; Srinivasan, A. (1996), "On the complexity of distributed network decomposition", Journal of Algorithms 20
- Schneider, J. (2010), "A new technique for distributed symmetry breaking" (PDF),Proceedings of the Symposium on Principles of Distributed Computing
- Schneider, J. (2008), "A log-star distributed maximal independent set algorithm for growth-bounded graphs" (PDF), Proceedings of the Symposium on Principles of Distributed Computing
- Beigel, R.; Eppstein, D. (2005), "3-coloring in time $O(1.3289^n)$", Journal of Algorithms54 (2)): 168–204, doi:10.1016/j.jalgor.2004.06.008
- Björklund, A.; Husfeldt, T.; Koivisto, M. (2009), "Set partitioning via inclusion–exclusion", SIAM Journal on Computing 39 (2): 546–563, doi:10.1137/070683933
- Brélaz, D. (1979), "New methods to color the vertices of a graph", Communications of the ACM 22 (4): 251–256, doi:10.1145/359094.359101
- Brooks, R. L.; Tutte, W. T. (1941), "On colouring the nodes of a network", Proceedings of the Cambridge Philosophical Society 37 (2): 194–197,doi:10.1017/S030500410002168X
- deBruijn, N. G.; Erdős, P. (1951), "A colour problem for infinite graphs and a problem in the theory of relations" (PDF), Nederl. Akad. Wetensch. Proc. Ser. A 54: 371–373(= Indag. Math. 13)