A Novel Potential-based Algorithm for the Vertex Coloring Problem and Its Application to Timetabling Online Platform DaAlgo

Hyunwoo Jung[1]

KSA of KAIST, Baegyanggwanmun-ro 105-47, Busanjin-gu, Busan, Republic of Korea
aryonhwjung@ksa.kaist.ac.kr

Abstract. Given an undirected graph $G(V, E)$, the Vertex Coloring Problem(VCP) requires to assign a color to each vertex in such a way that no two adjacent vertices have the same color and the number of colors used is minimized. A novel heuristic for graph coloring problem and its application to a timetable construction online platform is presented in this paper. It shows better results than the previous heuristic DSatur [9]. The online platform shows very fast timetable construction and very convenient UX. We want to demonstrate our platform in front of timetable specialists.

Keywords: Graph Coloring, Heuristic Algorithm, Online Platform

1 Introduction

Given an undirected graph $G(V, E)$, the Vertex Coloring Problem(VCP) requires to assign a color to each vertex in such a way that no two adjacent vertices have the same color and the number of colors used is minimized. The Vertex Coloring Problem is a well-known NP-hard Problem [1] with real world applications in many areas including scheduling [2], timetabling [3], register allocation [4], and communication networks [5]. Despite its importance to real-world applications, few exact algorithms for VCP have been found, and are able to solve only small instances up to 100 vertices for random graphs[6,7,8]. Many heuristic approaches have been proposed to deal with graphs of hundreds or thousands of vertices. The first heuristic approaches are derived based on greedy construction algorithms. The best-known greedy techniques are the maximum saturation degree(DSatur) and the Recursive Largest First(RLF) heuristics proposed by Brelaz [9] and by Leighton [10], respectively. Many meta-heuristic algorithms have been proposed for VCP based on tabu search [11], simulated annealing [12], genetic algorithm [13], linear programming [15], etc. In this paper, a novel heuristic for VCP with good performance is designed. In section 2, a novel heuristic **PAE** is shown. In section 3, a computational comparison result between **PAE** and **DSatur** is shown. In section 4, we show our timetable construction online platform that uses the algorithm **PAE**.

2 A Potential-based Algorithm for the Vertex Coloring Problem

In the following, a novel potential-based algorithm **PAE**(**G**, **k**) for VCP with exponential decay is described. The algorithm **PAE**(**G**, **k**) is checking whether the given graph **G** can be colored using at most k colors.

Algorithm 2: Potential-based Algorithm for the Vertex Coloring Problem with Exponential Decay(PAE)

Algorithm: **PAE**(**G**, **k**)
Data: **G**(**V**, **E**) and **k**
Result: colorability of vertices in **G**(**V**, **E**) using at most k colors
$n \leftarrow len(V)$
$Stack \leftarrow []$
for $v \leftarrow 1$ *to* n **do**
  $\phi_v = n$
end
Q is a priority queue containing all the vertices with their priorities which means the number of distinct colors of adjacent vertices for each vertex
while *there are any uncolored vertices* **do**
  pop vertices from Q until we get a vertex v with the recent update
  color v with the lowest usable color
  if *the number of used colors is greater than k* **then**
    if ϕ_v *is zero* **then**
      return *False*
    end
    $\phi_v = \phi_v // 2$
    pop vertices from Stack uncoloring(updating) the vertices up to Stack[ϕ_v]
    color v with the lowest usable color
  else
    $\phi_v = min(\phi_v, len(Stack))$
  end
  append v to Stack
end
return *True*

Intuitively, the potential ϕ_v for each vertex v means the highest possible order of v that leads to valid coloring using at most k colors. We do a binary search on the number of usable colors k to determine the tight k, this algorithm is called **PAE**. In this paper, as the priority of each element in the queue of the algorithm **PAE**, we are using the priority of DSatur [9] heuristic which is the number of distinct colors of adjacent vertices for each vertex. When the number of distinct colors of adjacent vertices for each vertex are same, the tie is broken by the higher degree of each vertex. The algorithm **PAE** is very simple and shows better performance than DSatur heuristic [9]. The time complexity of **PAE** is $O(|V|^2 \log(|V| + |E|))$. It can deal with graphs with thousands of vertices.

3 Experimental Results

We compare the performance of PAE and DSatur using the subset of the DIMACS benchmark graph coloring instances. The result is shown in the Table 1. We implemented the algorithms PAE and DSatur using Python. We run the program on a laptop machine with an Intel Core i7 CPU 2.30 GHz, and 24GB RAM. Large instances from the DIMACS benchmark whose number of vertices is greater than 100 are used for the comparison test. For all the cases, the algorithm PAE showed better or equal results on the aspect of the number of colors used. In the comparison table, boldfaced numbers mean that PAE shows better results than DSatur. The time in the table means the elapsed seconds until the program is finished. Note that the program is always finished with a coloring. Especially, in the class scheduling graphs shool1 and shool1_nsh, PAE showed almost optimal results. This is important since the algorithm **PAE** of this paper is used for an online platform that services the automatic construction of an exam timetable. χ^* means the best-known solution according to the paper by Malaguti et. al [16].

	PAE		DSatur [9]		
instance	k	time	k	time	χ^*
DSJC1000.1	**25**	277	27	0.214	20
DSJC1000.5	**113**	3309	115	0.968	83
DSJC1000.9	**296**	8162	310	2.156	224
DSJC125.1	6	0.093	6	0.016	5
DSJC125.5	**20**	2.223	22	0.028	17
DSJC125.9	**48**	9.056	51	0.033	44
DSJC250.1	**10**	2.672	11	0.025	8
DSJC250.5	**35**	24	37	0.074	28
DSJC250.9	**84**	107	90	0.125	72
DSJC500.1	15	39	15	0.054	12
DSJC500.5	**62**	319	66	0.248	48
DSJR500.1	**12**	0.095	13	0.019	12
DSJR500.5	**125**	163	132	0.279	122
latin_square_10	**124**	2220	132	1.066	99
le450_15a	**16**	2.423	17	0.048	15
le450_15b	16	1.586	16	0.039	15
le450_15c	**23**	28	24	0.076	15
le450_15D	**23**	35	24	0.083	15
le450_25c	**27**	34	29	0.081	26
le450_25d	28	20	28	0.079	26
le450_5a	9	3.435	9	0.028	5
le450_5b	**9**	5	10	0.022	5
le450_5d	**7**	3.721	8	0.063	5
school1	**14**	3.877	22	0.077	14
school1_nsh	**15**	4.303	25	0.059	14

Table 1: Performace Comparison between PAE and DSatur

4 Implementation of the Online Timetabling Platform DaAlgo

By generalizing the graph coloring idea of this paper, an online timetabling platform https://www.daalgo.org is implemented. The online platform services making exam timetables automatically using the customer's data about students' registrations for courses. We use an Excel file for input data. We process the input data using an algorithm that generalizes PAE, and then render the exam timetable to a browser after getting the constructed timetable from the backend. For the backend, we used FAST API, Postgres SQL, and Python. For the front end, we used React. For the cloud, we used the AWS cloud. We used AWS RDS, S3, Amplify, EC2, etc. See the Fig. 1 for the AWS architecture of our platform. Fig. 2 shows a result of an exam timetable construction which is rendered on the Chrome web browser. After rendering the exam timetable on the web browser, we can use drag and drop to change the exam schedules of some courses. Finally, we can download the exam timetable as the Excel format. We can make the exam timetable automatically and modify the timetable on the same platform at once.

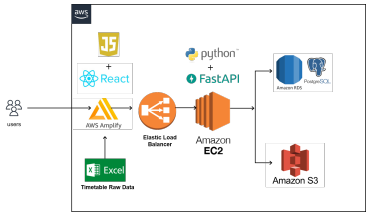

Fig. 1: AWS Architecture of DaAlgo Platform

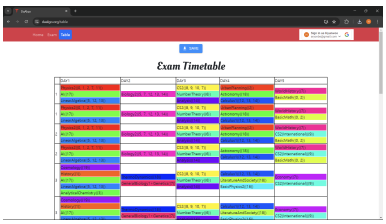

Fig. 2: The Result Exam Timetable

5 Concluding Remarks

In this abstract, we presented a novel heuristic for VCP which shows very good performance for school-related graphs. We apply the algorithm to implement an online service platform that constructs an exam timetable based on the data of students' registration information for courses. The online platform is very fast to make an exam timetable automatically and provides users with a very new UX. We will share our novel ideas and our creative timetable service platform.

Acknowledgements This research was supported by the research program at the Korea Science Academy of KAIST with funding from the Korean government (Ministry of Science and ICT).

References

1. M.R.Garey, D.S. Johnson.: Computers and Intractability: A Guide to the Theory of NP-Compleltеness. W.H.Freeman & Co., New York (1979).
2. F.T.Leighton.: A Graph Coloring Algorithm for Large Scheduling Problems. Journal of Research of the National Bureau of Standards, 84(6), 489–503 (1979).
3. D. De Werra.: An Introductoin to Timetabling. European Journal of Operational Research, 19:151-162 (1985).
4. T.K. Woo, S.Y.W. Su, and R. Newman Wolfe.: Resource Allocation in a Dynamically Partitionable Bus Network using a Graph Coloring Algorithm. IEEE Trans. Commun., 39(12), 1794-1801 (2002).
5. F.C. Chow and J.L.Hennessy.: The Priority-based Coloring Approach to Register Allocation. ACM Transactions of Programming Languages and Systems, 12(4), 501-536 (1990).
6. T.J. Sager and S. Lin.: A Pruning Procedure for Exact Graph Coloring. ORSA Journal on Computing, 3(3), 226-230 (1991).
7. E.C. Sewell.: An Improved Algorithm for Exact Graph Coloring. In D.S. Johnson and M.A. Trick, editors, Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challange, 1993, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 359-373 (1996).
8. II. Méndez-Díaz, P. Zabala.: A Branch-and-Cut Algorithm for Graph coloring, Discrete Applied Mathematics 154, 826–847 (2006).
9. D. Brɥelaz.: New Methods to Color the Vertices of a Graph. Communications of the ACM, 22(4), 251-256 (1979).
10. F.T. Leighton.: A Graph Coloring Algorithm for Large Scheduling Problems. Journal of Research of the National Bureau of Standards, 84(6), 489-503 (1979).
11. A. Hertz, D. de Werra.: Using Tabu Search Techniques for Graph Coloring, Computing 39, 345–351 (1987).
12. D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon.: Optimization by Simulated Annealing: an Experimental Evaluation; part II, Graph Coloring and Number Partitioning, Operations Research 39, 378–406 (1991).
13. L. Davis.: Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York (1998).
14. P. Galinier, J.K. Hao.: Hybrid Evolutionary Algorithms for Graph Coloring, Journal of Combinatorial Optimization 3, 379–397 (1999).
15. A. Mehrotra, M.A. Trick.: A Column Generation Approach for Graph Coloring, INFORMS Journal on Computing 8, 344–354 (1996).
16. E. Malaguti, M. Monaci, P. Toth.: A Metaheuristic Approach for the Vertex Coloring Problem, INFORMS Journal on Computing 20(2), 302–316 (2008).