Graph Coloring

Doncean Șerban-Gabriel, Oleniuc Iulian, Panaite Doru-Răzvan

14 March 2024

1 SOTA

"Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems" [1] This paper explores the intersection of graph coloring problems and deep learning techniques, focusing on the effectiveness of Graph Neural Network (GNN) models.

The model initially assigns the same randomly initialized embedding in R^d to all vertices in the graph. This embedding becomes a learned parameter as the model trains. To facilitate communication between neighboring vertices and between vertices and colors, the model requires both a vertex-to-vertex adjacency matrix and a vertex-to-color adjacency matrix. After initialization, adjacent vertices and colors communicate and update their embeddings over several iterations. The resulting vertex embeddings are then fed into a Multi-Layer Perceptron (MLP) to compute a logit probability, indicating whether the graph accepts a C-coloration (corresponding to the model's prediction of the answer to the decision problem: "does the graph G accept a $C-coloration$?"). The Graph Neural Network (GNN) model learns message functions, implemented as MLPs, to translate color embeddings into messages understandable by a vertex update function ($C : R^d \to R^d$), and to translate vertex embeddings into messages ($V : R^d \to R^d$). In addition, it learns functions (RNNs) for updating vertices ($V_u : R^{2d} \to R^d$) and colours ($C_u : R^{2d} \to R^d$) given their hidden states and received messages. The model uses 64-dimensional embeddings for both vertices and colours. The message computing MLPs consist of three layers (64, 64, 64) with ReLU nonlinearities, except for the linear activation in the output layer. The RNN cells used are basic LSTM cells with layer normalisation and ReLU activation. Training involves Stochastic Gradient Descent using TensorFlow's Adam optimizer for the message computing MLPs and RNNs.

The loss function is the binary cross-entropy between the model's final prediction and the ground truth, which is a Boolean indicating whether the graph accepts the target colourability in a given Graph Coloring Problem (GCP) instance.

Instance	Size	χ	Computed χ		
			GNN	Tabucol	Greedy
queen5_5	25	5	6	**5**	8
queen6_6	36	7	**7**	8	11
myciel5	47	6	5	**6**	**6**
queen7_7	49	7	8	8	10
queen8_8	64	9	8	10	13
1-Insertions_4	67	4	**4**	5	5
huck	74	11	8	**11**	**11**
jean	80	10	7	**10**	**10**
queen9_9	81	10	9	11	16
david	87	11	9	**11**	12
mug88_1	88	4	3	**4**	**4**
myciel6	95	7	**7**	**7**	**7**
queen8_12	96	12	10	**12**	15
games120	120	9	6	**9**	**9**
queen11_11	121	11	12	NA	17
anna	138	11	**11**	**11**	12
2-Insertions_4	149	4	**4**	5	5
queen13_13	169	13	14	NA	21
myciel7	191	8	NA	**8**	**8**
homer	561	13	14	**13**	15

Figure 1: Comparative results between Model, Tabucol and Greedy

The model performs better than Tabucol and greedy for some instances, although for some instances the model tends to underestimate its responses, erring on the minimum number of colours by predicting a lower number.

"Solving the graph coloring problem via hybrid genetic algorithms" [2] This paper proposed a hybrid genetic algorithm based on a local heuristic called DBG to give approximate values for the cromatic number of the graph. It achieves highly competitive results comparable with the best existing algorithms, but has a high runtime.

The method described in the article integrates a genetic algorithm (GA) with a local search heuristic known as DBG (Douiri and Elbernoussi, 2011) to tackle the graph coloring problem (GCP) in more detail. Initially, the algorithm starts by generating an initial population of individuals, where each individual represents a potential coloring configuration for the vertices of the graph. The number of colors initially used is often set to an arbitrary value, typically equal to the upper bound of the chromatic number of the graph. The objective function of the algorithm evaluates the quality of each individual solution by measuring the number of conflicts between adjacent vertices that have been assigned the same color. During the selection phase, individuals are chosen for reproduction based on their fitness, which is determined by the objective function. Higher fitness individuals are more likely to be selected for reproduction. The crossover operator takes two parent individuals from the population and combines parts of their

Table 1 Experimental results.

Graph	$\|V\|$	$\|E\|$	k^*	k_{Ours}	$T_{avg}(s)$	succ
anna	138	493	11	11	11.63	12/15
david	87	406	11	11	10.89	14/15
huck	74	301	11	11	11.10	15/15
2-Inser-3	37	72	4	4	2.23	15/15
3-Inser-3	56	110	4	4	3.37	15/15
jean	80	254	10	10	9.92	15/15
queen5.5	25	160	5	5	1.20	15/15
queen6.6	36	290	7	7	1.32	15/15
queen7.7	49	476	7	7	6.91	15/15
queen8.8	64	728	9	9	9.87	11/15
queen9.9	81	1056	10	10	13.96	13/15
miles250	128	387	8	8	4.39	11/15
miles500	128	1170	20	20	14.48	10/15
games 120	120	638	9	9	10.77	15/15
mug88-1	88	146	4	4	4.05	15/15
mug88-25	88	146	4	4	3.67	13/15
mug 100-1	100	166	4	4	8.34	11/15
mug 100-25	100	166	4	4	8.21	13/15
mycie13	11	20	4	4	0.01	15/15
mycie14	23	71	5	5	0.89	15/15
mycie15	47	236	6	6	5.41	15/15
mycie16	95	755	7	7	12.77	11/15
mycie17	191	2360	8	8	20.19	9/15
dsjc 125.1	125	736	5	6	13.12	14/15
dsjc1 125.5	125	3891	17	17	19.71	8/15
dsjc1 125.9	125	6961	44	44	35.87	7/15
dsjc1250.1	250	3218	8	8	26.07	10/15
dsjc1250.9	250	27897	72	72	75.81	6/15
fpso12.i.1	496	11654	65	65	82.03	2/15
fpso12.i.2	451	8691	30	30	69.79	6/15
fpso12.i.3	425	8688	30	30	67.14	4/15
zeroin.i.1	211	4100	49	49	28.24	5/15
zeroin.i.2	211	3541	30	30	83.39	9/15
zeroin.i.3	206	3540	30	30	27.06	6/15
mulsol.i.1	197	3925	49	49	25.75	2/15
mulsol.i.2	188	3885	31	31	22.07	7/15
mulsol.i.3	184	3916	31	31	23.49	5/15
mulsol.i.4	185	3946	31	31	25.22	4/15
mulsol.i.5	186	3973	31	31	28.33	7/15

Figure 2: Results of the Hybrid GA Approach

solutions at a randomly chosen crossover point to create offspring individuals. These offspring solutions undergo adjustments to ensure that they adhere to the constraints of the problem. Mutation introduces small changes to individual solutions to explore new regions of the solution space. In this method, mutation involves changing the color of a vertex in an existing solution, guided by the objective function to retain only beneficial changes. The genetic algorithm iterates through selection, crossover, and mutation operators, continually evaluating the fitness of resulting individuals. Additionally, the DBG local search heuristic is applied at various stages to further improve solution quality by reducing conflicts. The termination condition of the algorithm is met when a satisfactory solution with minimal conflicts (ideally, no conflicts) is found, or after reaching a predefined maximum number of iterations. By combining genetic algorithms with the DBG local search heuristic, the method efficiently explores the solution space of the graph coloring problem, aiming to find high-quality solutions within reasonable computational time.

The article assesses this new method's performance against existing algorithms on standard benchmark graphs. The results likely show it achieving competitive results, potentially finding the minimum number of colors for most graphs. This highlights the effectiveness of the combined genetic algorithm and local search heuristic. However, the efficiency of genetic algorithms can be sensitive to parameters, so for some complex graphs the solution might take longer

or get stuck on suboptimal colorings. Overall, the research suggests this hybrid approach has promise for solving graph coloring problems, with further exploration needed to optimize its performance on various graph structures.

"A Discrete Firefly Algorithm Based on Similarity for Graph Coloring Problems" [3] The Firefly Algorithm, inspired by the flashing behavior of fireflies, is an optimization technique where fireflies in a search space represent potential solutions, attracted to brighter ones while adjusting their positions iteratively, aiming to find optimal or near-optimal solutions to various optimization problems. This paper proposes a novel non-hybrid discrete firefly algorithm for solving planar graph coloring problems. It discretizes the firefly algorithm using the *similarity* metric:

$$\text{similarity}_{ij} = 1 - \frac{H(x_i, x_j)}{n},$$

where x_i and x_j are two colorings.

The firefly algorithm is compared with HDPSO (Hamming Distance Particle Swarm Optimization) and HDABC (Hamming Distance Artificial Bee Colony). The proposed DFA obtains an excellent success rate. When applied to relatively small graphs, say $n = 90$, the success rate of DFA is higher than HDPSO and nearly equal well with DABC at $d = 2.5$ (where d is the constraint density), which is the most difficult problem. However, as the size of graph increases, DFA brings obvious advantages not only at $d = 2.5$, but also at $d = 2$ and $d = 3$. For example, the succcess rate of DFA is 78% when the graph size is 150 and $d = 2.5$. This is much higher than DABC's 20% and HDPSO's 1%. On the other hand, its evaluation times of solving the most difficult problems are much larger than HDPSO and DABC.

2 Benchmark instances

In order to test our methods, we will used a number of 79 test instances from various sources presented below representing different graph typologies. Of these, for 56 the minimum number of colours required for colouring is known, for the rest an optimal value has not yet been determined. The test instances have as origin:

- DSJ (from David Johnsom) - 15 instances: random graphs, where DSJC (12 instances) are standard random graphs, DSJR (2 instances) are geometric graphs and DSJR.c (1 instances) are complements of geometric graphs. For all this the optimum solution is not found.
- CUL (from Joe Culberson) - 6 instances: quasi-random coloring problems
- REG (from Gary Lewandowski) - 14 instances: Problem based on register allocation for variables in real codes.

- LEI (from Craig Morgenstern) - 12 instances: Leighton graphs with guaranteed coloring size.
- SCH (from Gary Lewandowski) - 2 instances: Class scheduling graphs, with and without study halls.
- LAT (from Gary Lewandowski) - 1 instance: Latin square problem.
- SGB (from Michael Trick and Donald Knuth) - 24 instances: these can be divided into:
  - Book Graphs - 5 instances: Given a work of literature, a graph is created where each vertex represents a character. Two vertices are connected by an edge if the corresponding characters encounter each other in the book. Knuth creates the graphs for five classic works: Tolstoy's Anna Karenina (anna), Dicken's David Copperfield (david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck), and Hugo's Les Misérables (jean).
  - Game Graphs - 1 instance: A graph representing the games played in a college football season can be represented by a graph where the vertices represent each college team. Two teams are connected by an edge if they played each other during the season. Knuth gives the graph for the 1990 college football season.
  - Miles Graphs - 5 instances: These graphs are similar to geometric graphs in that vertices are placed in space with two vertices connected if they are close enough. These graphs, however, are not random. The vertices represent a set of United States cities and the distance between them is given by by road mileage from 1947. These graphs are also due to Kuth.
  - Queen Graphs - 13 instances: Given an n by n chessboard, a queen graph is a graph on n^2 vertices, each corresponding to a square of the board. Two vertices are connected by an edge if the corresponding squares are in the same row, column, or diagonal. Unlike some of the other graphs, the coloring problem on this graph has a natural interpretation: Given such a chessboard, is it possible to place n sets of n queens on the board so that no two queens of the same set are in the same row, column, or diagonal? The answer is yes if and only if the graph has coloring number n. Martin Gardner states without proof that this is the case if and only if n is not divisible by either 2 or 3. In all cases, the maximum clique in the graph is no more than n, and the coloring value is no less than n.
- MYC (from Michael Trick) - 5 instances: Graphs based on the Mycielski transformation. These graphs are difficult to solve because they are triangle free (clique number 2) but the coloring number increases in problem size.

References

[1] H. Lemos, M. Prates, P. Avelar, and L. Lamb, "Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems," 2019.

[2] S. M. Douiri and S. Elbernoussi, "Solving the graph coloring problem via hybrid genetic algorithms," *Journal of King Saud University-Engineering Sciences*, vol. 27, no. 1, pp. 114–118, 2015.

[3] K. Chen and H. Kanoh, "A discrete firefly algorithm based on similarity for graph coloring problems," in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2017, pp. 65–70.