Combining Reinforcement Learning and Heuristic Optimization: A Model Based on a Deep Q-Network and Graph Neural Networks for Graph Coloring

SeokJin Kwon
School of Software
Kwangwoon University
Seoul, Republic of Korea
cleverlazjin@gmail.com

Yong-Hyuk Kim
School of Software
Kwangwoon University
Seoul, Republic of Korea
yhdfly@kw.ac.kr

***Abstract*—We present a hybrid approach combining Reinforcement Learning (RL) with the TabuCol, which is a version of tabu search specifically designed for the Graph Coloring Problem (GCP), enhanced by Graph Neural Networks (GNNs), to tackle the GCP. Our model, referred to as RLTB, integrates a Deep Q-Network architecture with multiple GNN layers to select promising (node, color) pairs, guiding the RL agent towards optimal solutions. Experimental evaluations on standard DIMACS graph instances demonstrate that RLTB significantly outperforms existing heuristic algorithms such as Tabucol and a greedy algorithm, surpasses GNN methods from other papers, and shows competitive performance against other outstanding hybrid methods.**

***Index Terms*—Graph Coloring Problem, Reinforcement Learning, Deep Q-Network, Graph Neural Networks, Tabu Search**

I. Introduction

Many optimization problems can be represented as graph problems, among which the Graph Coloring Problem (GCP) is one of the most well-known. GCP involves coloring the vertices of a graph such that no adjacent nodes share the same color while minimizing the total number of colors used. GCP has been extensively researched and applied in various real-world fields, such as register allocation [1], frequency assignment [2], and scheduling [3]. GCP is particularly significant because it is an NP-hard problem, meaning it plays a central role in computational complexity theory, especially in the context of the P versus NP question. Since GCP is NP-hard, no algorithm is known to solve it in polynomial time, making it a key target for heuristic and approximation methods.

In recent years, machine learning algorithms have seen explosive development across various fields. Reinforcement Learning (RL), in particular, has emerged as a powerful tool for solving combinatorial optimization problems [4], [5]. Additionally, Graph Neural Networks (GNNs) [6], [7], which are neural networks designed to handle graph-structured data, have been developed and applied in many areas. Given that GCP is inherently a graph problem, GNNs are particularly well-suited for this task.

In this paper, we present a model that combines the reinforcement learning mechanism with the heuristic algorithm called TabuCol [8]. We also integrate GNN structures to enhance the performance of the RL agent. Our approach combines RL and heuristic optimization to tackle GCP efficiently.

We conducted our experiments in two stages. First, we compared our model with existing heuristic algorithms such as the Greedy algorithm and TabuCol, as well as GNN methods from [9], using relatively easy DIMACS graph instances [10], [11]. In the second stage, we tested our model against other hybrid methods on more challenging graphs.

The results of our experiments indicate that our model, called RLTB, outperformed other heuristic methods and GNN approaches on the easier graph instances. Furthermore, when evaluated on the more difficult graphs, our model performed competitively against existing hybrid approaches, demonstrating its robustness and effectiveness.

This paper makes the following key contributions:

- We propose a novel hybrid model, RLTB, which RL with the TabuCol heuristic algorithm, enhanced by GNNs, to address the GCP. This hybrid approach leverages the strengths of both machine learning and traditional heuristic optimization techniques.
- Extensive experiments on standard DIMACS graph instances demonstrate that RLTB significantly outperforms existing heuristic algorithms (e.g., TabuCol, Greedy) and GNN-based methods, particularly on small and medium-sized graphs. Also when compared to other hybrid approaches, RLTB performs competitively, especially on more challenging graph instances.

The paper is organized as follows: in Section 3, we present some preliminary backgrounds. Section 4 describes the proposed approach in detail. In Section 5, we show computational results of our model. Conclusions are given in

the last section.

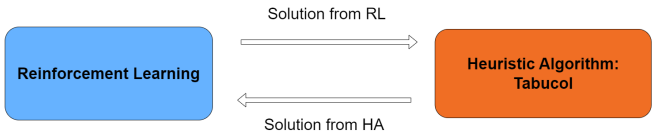

Fig. 1: Overview of our model framework, which is based on RLHO framework [5]. We implemented an exponential probability function for the RL steps, causing the probability of selecting RL steps to decrease as the process continues, while the probability of selecting Heuristic Optimizer steps increases over time.

II. Related Work

A. Heuristic algorithms for the Graph Coloring Problem

The most basic and traditional algorithms are the greedy and DSATUR (degree of saturation) [12] algorithms. As greedy algorithms construct solutions step by step by choosing the locally optimal choice at each step, in the GCP, a greedy method colors each vertex in a sequence by choosing the smallest number of colors that does not conflict with other adjacent vertices. DSATUR algorithm is a more sophisticated greedy algorithm that selects the next vertex to color based on the saturation degree, which is the number of differently colored neighbors the vertex has. In this manner, the vertex with the highest saturation degree is colored first. Local search methods are also prominent approaches used to tackle the GCP. Local search methods, such as tabu search [8], [13] and simulated annealing (SA) [14], iteratively improve a solution by searching neighbor solutions.

B. Machine Learning Methods for Combinatorial Optimization Problems including GCP

Graph Neural Networks (GNNs) have emerged as powerful tools for solving Combinatorial Optimization Problems (COPs) due to their ability to capture complex graph structural information. For example, pure deep learning mechanism stacking GNN layers is used by [9], and Watkins et al. [15] used RL method using Deep Q-Network(DQN) with GNNs. This model added GNN structure to Q-Network [16]. But graphs they tested were only small and medium sized graphs. Also the results they made were not better than existing heuristic algorithms such as tabucol, DSATUR, greedy, and so on.

Hybrid methods that combine RL and Heuristic Optimization (HO) have been widely studied for various COPs. These approaches leverage the strenghs of RL for learning effective strategies and HO for efficiently refining solutions. Cai et al. [5] constructed RLHO framework, taking RL for x steps and HO for y steps iteratively. They applied this framework to the bin packing problem, using Proximal Policy Optimization (PPO) for the RL component and SA for the HO component. This framework is also applied for the GCP in [17]. They used PPO for the RL component and Tabucol as the heuristic component. Their model achieved quite good results on large and challenging graphs. Zhou et al. [18] employed probability learning to guide local search in COPs. Building on this approach, Zhou et al. [19] adapted probability learning to generate starting solutions for tabu search, specifically addressing the GCP. Zhou et al. [19] introduced two key improvements over [18]: a group matching procedure that accounts for the interchangeability of color groups in the GCP, and the replacement of the basic descent-based local search in [18] with a more effective tabu search-based coloring algorithm.

Our model, RLTB, closely relates to the work [5], [17], as it also employs the RLHO framework and uses Tabucol as the heuristic algorithm. However the key difference from [17] and RLTB lies in the RL method. We adopted DQN enhanced with GNNs while they used PPO method. We chose to incorporate GNNs because GCP inherently deals with graph structures, and GNNs are well-suited to capture and exploit the relational information within graphs.

III. Background

In this section, we briefly introduce the main concepts and basics of our study.

A. Graph Coloring Problem

Let G be an undirected graph, $G = (V, E)$ with vertex set V and edge set E. Now we define a mapping function f, where $f : V \mapsto \{1, 2, ..., k\}$. Then the value $f(v)$ of vertex v is the color of v. If two adjacent vertices x and y have the same color j, then it is called a *conflict*. A coloring with no conflicts is called a proper-coloring. The chromatic number of G, denoted by $\chi(G)$, is the smallest k for which there exists a k-coloring of G. Based on these definitions, GCP is the problem of finding the chromatic number of given graph G, which is the minimum number of colors required to color the vertices of G such that no two adjacent vertices share the same color.

B. Tabu Search

Tabu search [13], [20] is one of the global search methods that is designed to overcome the limitations of traditional local search algorithms. Tabu search begins with an intial feasible solution and iteratively explores the solution space by moving to neighboring solutions. To overcome the drawbacks of other local search methods, which often get stuck in local optima, tabu search tries to escape local optima by using a short-term memory, known as the tabu list. The tabu list is a memory structure that temporarily forbids certain moves, preventing the algorithm from revisiting recently explored solutions. This helps in avoiding cycles and encourages the exploration of new areas of the solution space.

C. Deep Q-Network

A deep Q-Network [16] is a type of reinforcement learning algorithm that combines Q-Learning with deep neural networks to handle continuous state spaces. The main idea is

to use a neural network to approximate the Q-value function, which is the expected cumulative reward for taking a given action in a given state and following the optimal policy thereafter.

A neural network with parameters θ, known as the Q-network, is adopted to approximate the Q-value function. Training the Q-network involves minimizing a loss function $L(\theta)$, which is computed based on the discrepancy between the predicted Q-values and the target values that reflect the expected rewards.

$$L(\theta) = \mathbb{E}_{s,a\sim\rho}[(y - Q(s, a : \theta))^2] \tag{1}$$

In this context, the target value y is defined as $y = \mathbb{E}_{s'\sim\epsilon}[r + \gamma \max_{a'} Q(s', a' : \theta)|s, a]$, where γ, the discount factor, determines the importance of future rewards, and $\rho(s, a)$ represents the probability distribution over the state-action pairs, often referred to as the behavior distribution. The target value y is computed based on the immediate reward r and the discounted maximum Q-value for the next state s', guiding the network toward optimal decision-making [16].

The two major features of DQN are Experience Replay and Target Network. By using experience replay method, we store trajectories (state, action, reward, next state) in a replay buffer instead of updating the Q-network after each action. During training, random samples from this buffer are used to update the Q-network. This helps to break the correlation between consecutive samples and leads to more stable learning. A separate target network with parameters $\hat{\theta}$ is used to generate the target values y_i. The parameters of the target network are periodically updated to match the parameters of the Q-Network. This helps to stabilize the learning process.

D. Graph Neural Network

A graph neural network is a type of artificial neural networks for processing data that can be represented as graphs. Given graph G, attributes of graph G in GNN are feature of nodes, feature of edges and graph feature. Each feature makes a graph consisted of nodes and each node has a feature vector, which is called a *message*, that is sent to all its neighbors. After receiving messages from neighbors in each node, aggregate and update functions are executed so we get changed feature graph. This process is called the Message Passing, which is the core of GNN [6], [21].

IV. Methodology

A. Deep Q-Network Architecture for Graph Coloring

In our approach, we applied DQN to solve the GCP by defining a single Q-Network. This network is designed to evaluate and select the most promising (node, color) pair during the action selection process. The overall architecture of the Q-Network is illustrated in Figure 2.

The Q-Network is responsible for assessing the potential of each (node, color) combination within the graph. Inputs to the network include both node features and color features. These features are concatenated to form a comprehensive representation of each possible (node, color) pair.

The Q-Network processes these combined features through multiple layers, including GNN layers for message passing and feature aggregation, followed by fully connected layers. The final output layer produces Q-values corresponding to each (node, color) pair, representing the expected future rewards of selecting that specific action in the current state.

The implementation of our DQN model was based on the CleanRL framework [22], which provides a robust and efficient foundation for constructing reinforcement learning algorithms.

B. Applying Graph Neural Network

We designed three types of features to effectively represent the graph structure and color assignments: graph features, node features, and color features. Specifically, we integrated both node and color features within our Q-Network to evaluate and select the most promising (node, color) pairs for action selection.

Each node in the graph is characterized by four features: degree, degree centrality, the number of conflicts with adjacent nodes, and the number of colors used by neighboring nodes. These node features provide essential information about each node's connectivity and its interactions within the graph. Additionally, we introduced three color features for each color: the number of conflicts that would arise when using a specific color, the number of nodes currently assigned that color, and the ratio of usage of that color relative to the total number of nodes in the graph. These color features capture the prevalence and potential conflicts associated with each color choice.

The node and color features are concatenated to form comprehensive (node, color) pair representations. These combined features are then passed through multiple GNN layers, particularly Graph Convolutional Network (GCN) layers, within our Q-Network. In these GCN layers, message passing is performed to aggregate and update feature representations based on the graph's connectivity, effectively capturing both local and global structural information. This process results in an updated feature graph that integrates the nuanced interactions between nodes and colors.

C. Combining RL and Heuristic Algorithm

1) Heuristic Algorithm Based on TabuCol: TabuCol is a heuristic algorithm that applies the principles of tabu search to the GCP, which is first introduced by Glover et al. [20]. It utilizes the tabu list to avoid revisiting recently explored solutions and enhance the search process. Following the method of Hertz and Werra [8], we incorporate an *aspiration level* function within the TabuCol method. This function allows move that is classified as tabu to be accepted if it meets certain aspiration criteria, typically if it results in a solution better than any previously found solution. This mechanism ensures that the search does not overlook potentially optimal solutions.

To effectively evaluate the quality of a coloring solution, we introduce a conflict function f. This function calculates the

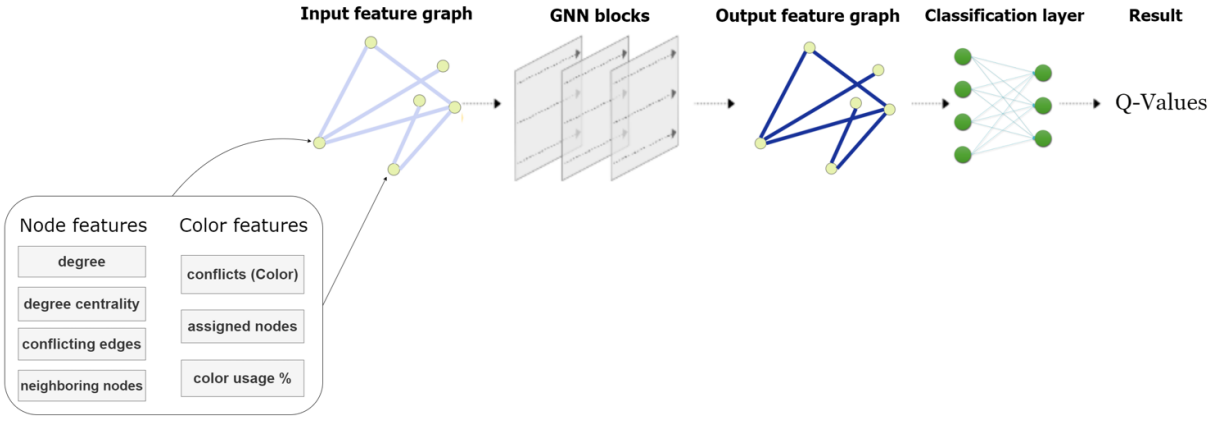

Fig. 2: Diagram illustrating the architecture of the Q-Network designed for the Graph Coloring Problem. The input feature graph consists of node features and color features, which are concatenated to form comprehensive (node, color) pair representations. These combined features are processed through multiple GNN layers, where message passing enables the aggregation and updating of node features based on their neighbors and associated colors. The transformed node feature graph is then passed through fully connected linear layers, culminating in the generation of Q-values for each (node, color) pair. The network identifies and outputs the most promising node-color pair based on the highest Q-value.

total number of conflicting edges in the current graph coloring. Formally, for a given coloring C, the conflict function is defined as:

$$f(C) = \sum_{\{u,v\} \in E} \delta(u, v)$$

where δ is the indicator function that returns 1 if if both vertices u and v are assigned the same color and 0 otherwise.

Based on these ideas, pseudocode for our heuristic algorithm is in the following.

Algorithm 1 TabuCol algorithm [8]

Input: graph $G = (V, E)$, initial solution s_0, $reps$ and $maxiter$
Initialization $iter \leftarrow 0$, tabu list $T \leftarrow \emptyset$, best solution $s^* \leftarrow s_0$
while $iter < maxiter$ **do**
  Generate $reps$ number of neighboring solutions by changing the color of one vertex
  Select the best move s' in this iteration that is not in T or meets the aspiration criteria
  Update the coloring of G with the selected move s'
  if $f(s') \leq f(s^*)$ **then**
    $s^* \leftarrow s'$
  end if
  Add the selected move s' to T
  Remove the oldest move from T if it exceeds the length limit
end while
return the best solution s^* found

2) Combining RL and Heuristic Algorithm: Our approach combines a RL agent with TabuCol algorithm to solve the GCP. The combined structure consists of two main components: the RL agent and the TabuCol heuristic optimizer (HO).

The RL agent is responsible for generating initial solutions for the GCP. These initial solutions serve as the starting point for the TabuCol heuristic. Once the RL agent provides an initial coloring, the TabuCol algorithm takes over to refine and search for improved solutions, starting from the RL-generated initialization.

To explore and search vast and diverse areas of the solution space, we introduced a probabilistic model for selecting HO steps. In the initial phases, the RL steps are predominantly applied, allowing the RL agent to explore a wide range of possibilities. As the process progresses, HO steps, specifically the TabuCol algorithm, are increasingly employed. This means that our model selects the TabuCol algorithm with a probability p. Initially, the model mainly relies on RL steps, while the TabuCol algorithm is occasionally invoked. As the process continues, the frequency of HO steps increases, with the TabuCol algorithm being applied more regularly. The probability p for selecting the HO is defined as follows:

$$p_{\text{HO}} = 1 - e^{(\psi \times \frac{\text{processed steps}}{\text{global total steps}})} \tag{2}$$

where ψ is the parameter for adjusting how fast the shift is. This formula indicates that as the number of steps increases, the probability p for selecting the HO also increases, leading to a gradual shift from RL-driven exploration to HO-driven optimization.

3) Action: In our approach, the action space is composed of pairs of nodes and colors, representing potential moves in the GCP. Specifically, the RL agent selects an action from this space by choosing the most promising (node, color) pair.

4) Reward Function: The default reward for per step is set to -1.0 e-4. The reward consists of two parts, RL reward and heuristic (TabuCol) reward. In our RL framework, the reward is designed to encourage the reduction of conflicts. The conflict function $f(s)$ returns the number of conflicts given

Algorithm 2 Pseudocode of RLTB Model

Input: graph $G = (V, E)$, number of colors k, total steps T, parameter ψ
Initialization $s, s^* \leftarrow$ random k-coloring
for $t = 1$ to T **do**
  $p_{\text{HO}} \leftarrow 1 - e^{(\psi \times \frac{t}{T})}$
  $r \leftarrow \text{Random}(0, 1)$
  if $r < p_{\text{HO}}$ **then**
    $s \leftarrow \text{TabuCol}(s)$
  else
    $s \leftarrow \text{RL Step}(s)$
  end if
  $reward \leftarrow reward_{\text{RL}} + reward_{\text{TB}}$
  Give $reward$ to the RL agent
  if $f(s) \leq f(s^*)$ **then**
    $s^* \leftarrow s$
  end if
end for
return s^*

state s. To motivate the progress by the RL agent, we defined the reward as:

$$reward_{\text{RL}} = \lambda \times (f(s) - f(s')) \tag{3}$$

where $f(s)$ and $f(s')$ are the numbers of conflicts in states s and s', respectively and λ is a scaling factor that adjusts the magnitude of the reward. In the heuristic (TabuCol) reward, the reward is calculated based on the improvement achieved by the TabuCol algorithm. Specifically, it is defined by the difference between the number of conflicts in the initial solution provided to the HO and the number of conflicts after the TabuCol algorithm has been applied. Thus the reward function is expressed as:

$$reward_{\text{TB}} = f(s_{\text{initial}}) - f(s_{\text{final}}) \tag{4}$$

The final reward considering these two rewards is as follows:

$$reward = reward_{\text{RL}} + reward_{\text{TB}} \tag{5}$$

V. Experiments and Results

To assess the effectiveness of our model in solving the GCP, we conducted experiments using a diverse set of DIMACS graph instances, including *dsjc*, *queen*, *flat*, and *game* graphs. These graph types are commonly utilized to evaluate performance on the GCP, providing a comprehensive assessment of our model's generalization capabilities across various structural complexities. The experiments were conducted on a machine with an Intel Core i5-12600K CPU at 3.70 GHz, 32GB of RAM, an NVIDIA GeForce RTX 3060 GPU, and running Ubuntu 22.04 LTS.

Table1 presents the key parameter values used in our experiments. We set the learning rate to 2.5e-4, the batch size to 128, psi (ψ) to -0.5 for controlling the probability of Heuristic Optimization (HO) steps, and lambda (λ) to 0.01 for weighting the reward function.

TABLE I: Key Parameters for Training the RLTB Model

Parameter	**Value**
Learning Rate (α)	2.5e-4
Batch Size	128
Psi (ψ)	-0.5
Lambda (λ)	0.01

Our evaluation was divided into two main parts. First, we compared our model with existing heuristic algorithms and GNN methods on relatively easy graphs. Second, we tested our model on more challenging graphs, comparing its performance with other hybrid approaches. Each test on a graph was executed with a maximum time limit of 7 hours.

TABLE II: Comparison of Chromatic Numbers produced by our model, RLTB, GNN [9], and two other heuristic algorithms on easy instances of DIMACS graphs

Instance	Order	χ/k^*	RLTB	GNN [9]	TabuCol [9]	Greedy [9]
queen5_5	25	5/5	**5**	6	5	8
queen6_6	36	7/7	**7**	7	8	11
myciel5	47	6/6	**6**	5	6	6
queen7_7	49	7/7	**7**	8	8	10
queen8_8	64	9/9	**9**	8	10	13
queen9_9	81	10/10	**10**	9	11	16
myciel6	95	7/7	**7**	7	7	7
games120	120	9/9	**9**	6	9	9
queen11_11	121	11/11	13	12	NA	17
myciel7	191	8/8	**8**	NA	8	8

Note: the value "NA" indicates that we were unable to find the corresponding results either from any other papers or from our own experiments.

The results in Table 2 indicate that our model, RLTB, consistently outperformed existing heuristic algorithms such as Tabucol and a greedy algorithm, as well as GNN methods from [9] on the easier DIMACS graph instances. RLTB was able to determine the chromatic number for all easy DIMACS graphs, except the *queen11_11* graph.

TABLE III: Comparison of Chromatic Numbers produced by our model, RLTB, RLTCOL [17], RLS [18] and PLSCOL [19] on some challenging instances of DIMACS graphs

Instance	Order	χ/k^*	RLTB	RLTCOL [17]	RLS [18]	PLSCOL [19]
dsjc125.1	125	?/5	**5**	NA	5	5
dsjc125.5	125	?/17	**17**	18	17	17
dsjc250.1	250	?/8	**8**	8	8	8
dsjc250.5	250	?/28	29	28	29	28
flat300_28_0	300	28/28	**30**	30	32	30
le450_5c	450	5/5	**5**	NA	NA	NA
le450_15b	450	15/15	16	NA	15	15
le450_15d	450	15/15	**15**	16	15	15
le450_25a	450	25/25	**25**	25	26	25
le450_25b	450	25/25	**25**	25	26	25
dsjc500.1	500	?/12	13	12	13	12
dsjc500.5	500	?/47	**48**	48	50	48

Note: the symbol "?" indicates that the chromatic number for the corresponding graph is not known and value "NA" indicates that we were unable to find the corresponding results either from any other papers or from our own experiments.

When tested on more challenging graphs, RLTB demonstrated robust and consistent performance, often surpassing other existing hybrid methods. In comparison with RLTCOL

[17], which also applied the RLHO framework, our model showed superior results on the *dsjc125.5* and *le450_15d* graphs, demonstrating its ability to efficiently handle certain types of challenging instances. However, RLTCOL performed better on the *dsjc250.5* and *dsjc500.1* graphs. These results indicate that while RLTB has robust performance and can surpass other hybrid methods on specific graphs, different approaches may excel depending on the characteristics of the graph instances. Overall, RLTB has demonstrated strong and consistent performance, particularly on smaller and medium-sized graphs. However, as the order and size of the graphs increase, the accuracy of RLTB tends to decrease.

VI. Concluding Remarks

In this paper, we introduced a hybrid model, RLTB, that combines reinforcement learning (RL) with heuristic algorithms to address the graph coloring problem (GCP). Our model integrates a deep Q-Network (DQN) architecture with multiple graph neural network (GNN) layers.

The experimental results demonstrate that RLTB significantly outperformed existing heuristic algorithms such as *TabuCol* and a greedy algorithm, as well as GNN methods from [9], [15], across almost all of easy DIMACS graph instances. When compared with other hybrid methods [17]–[19], RLTB showed robust performance, often matching or surpassing these models, particularly on more challenging graph instances. This indicates the effectiveness of our approach in leveraging both RL and GNNs to improve the quality of solutions in the GCP.

However, due to time constraints, we were unable to explore alternative RL mechanisms or heuristic algorithms within the RLTB framework. Exploring variations such as different RL algorithms or advanced heuristics could potentially yield significant performance improvements. Moving forward, we plan to experiment with alternative RL methods like Proximal Policy Optimization (PPO) and integrate heuristics such as Simulated Annealing or variations of TabuCol to enhance the model's capabilities. Additionally, we aim to extend the application of the RLTB model to other combinatorial optimization problems, further investigating its adaptability and effectiveness across diverse contexts.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(No. 2021R1F1A1048466). It was also supported by Korea Institute of Marine Science & Technology Promotion (KIMST) funded by the Ministry of Oceans and Fisheries, Korea (RS-2022-KS221629).

References

[1] Preston Briggs, Keith D. Cooper, and Linda Torczon, "Improvements to graph coloring register allocation", ACM Trans. Program. Lang. Syst. 16, 3 (may 1994), 428–455, 1994

[2] Taehoon Park and Chae Y. Lee, "Application of the Graph Coloring Algorithm to the Frequency Assignment Problem", Journal of the Operations Research Society of Japan 39, 2 (1996), 258–265, 1996

[3] Nicolas Zufferey, Patrick Amstutz, and Philippe Giaccari, "Graph colouring approaches for a satellite range scheduling problem", J. of Scheduling 11, 4 (August 2008), 263–277, 2008

[4] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev, "Reinforcement Learning for Combinatorial Optimization: A Survey", arXiv:2003.03600 [cs.LG], 2020

[5] Qingpeng Cai, Will Hang, Azalia Mirhoseini, George Tucker, Jingtao Wang, and Wei Wei, "Reinforcement Learning Driven Heuristic Optimization", arXiv:1906.06639 [cs.LG], 2019

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini, "The Graph Neural Network Model", IEEE Transactions on Neural Networks 20, 1 (2009), 61–80, 2009

[7] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li and Maosong Sun, "Graph Neural Networks: A Review of Methods and Applications", arXiv:1812.08434 [cs.LG], 2021

[8] Alain Hertz and D. Werra, "Using Tabu Search Techniques for Graph Coloring", Computing 39, 345-351. Computing 39 (December 1987), 1987

[9] H. Lemos, M. Prates, P. Avelar, and L. Lamb, "Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems", In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE Computer Society, Los Alamitos, CA, USA, 879–885, 2019

[10] Jin Kao Hao Daniel Porumbel and Pascale Kuntz, "DIMACS Graphs: Benchmark Instances and Best Upper Bounds", Accessed: 2024-08-30, 2011

[11] Carnegie Mellon University. [n. d.]. "Graph-Coloring-Instances", https://mat.tepper. cmu.edu/COLOR/instances.html, unpublished, Accessed: 2024-08-30.

[12] Daniel Brélaz, "New methods to color the vertices of a graph", Commun. ACM 22, 4 (apr 1979), 251–256, 1979

[13] Fred Glover and Manuel Laguna, "Tabu search I. Vol. 1.", 1999

[14] M. Chams, A. Hertz, and D. de Werra, "Some experiments with simulated annealing for coloring graphs", European Journal of Operational Research 32, 2 (November 1987), 260–266, 1987

[15] George Watkins, Giovanni Montana, and Juergen Branke, "Generating a Graph Colouring Heuristic with Deep Q-Learning and Graph Neural Networks" arXiv:2304.04051 [cs.LG], 2023

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller, "Playing Atari with Deep Reinforcement Learning", NIPS Deep Learning Workshop 2013 (12 2013).

[17] Adrian Salamon and Klara Sandström, "Reinforcement learning for improved local search : Applied to the graph coloring problem", unpublished, 2023

[18] Yangming Zhou, Jin-Kao Hao, and Béatrice Duval, "Reinforcement learning based local search for grouping problems: a case study on graph coloring", arXiv:1604.00377 [cs.AI], 2016.

[19] Yangming Zhou, Béatrice Duval, and Jin-Kao Hao, "Improving probability learning based local search for graph coloring" Appl. Soft Comput. 65, C (April 2018), 542–553, 2018

[20] Fred Glover, C. McMillan, and Beth Novick, "Interactive decision software and computer graphics for architectural and space planning" Annals of Operations Research 5 (October 1985), 557–573, 1985

[21] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce and Alexander B. Wiltschko, "A Gentle Introduction to Graph Neural Networks" Distill, 2021

[22] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye and Jeff Braga, "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms" arXiv:2111.08819 [cs.LG], 2021