GCP Approaches and Improved Results

Lungu Andrei, Isac Lucian, Fomin Bogdan

June 2025

1 Deep Q-Learning using GNNs

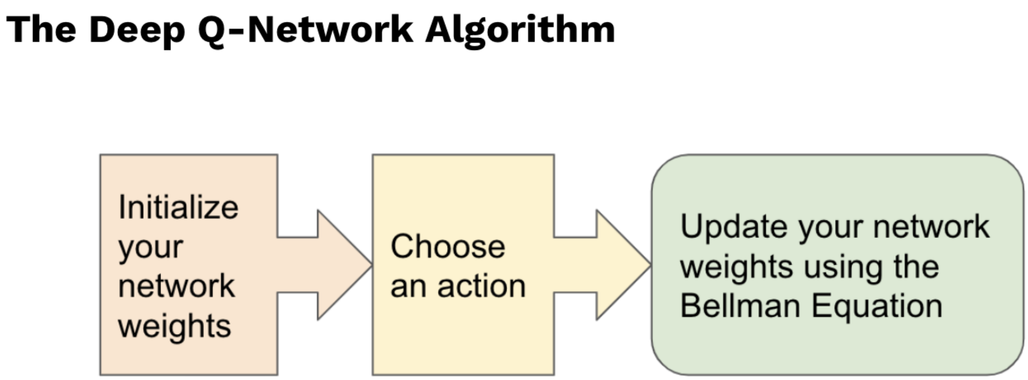


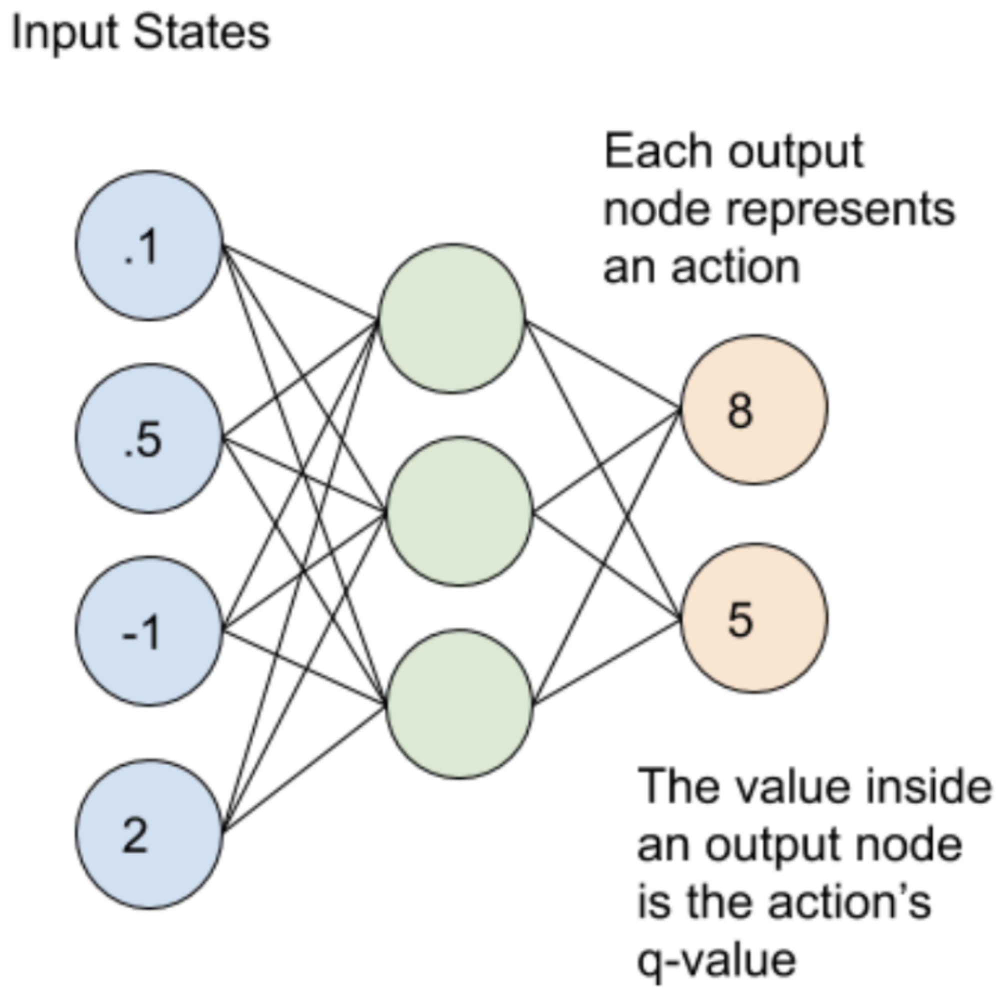
Figure 5: The Deep Q-Network Algorithm (Image by Author)

1. Initialize your Main and Target neural networks
2. Choose an action using the Epsilon-Greedy Exploration Strategy
3. Update your network weights using the Bellman Equation

A core difference between Deep Q-Learning and Vanilla Q-Learning is the implementation of the Q-table. Critically, Deep Q-Learning replaces the regular Q-table with a neural network. Rather than mapping a state-action pair to a q-value, a neural network maps input states to (action, Q-value) pairs.

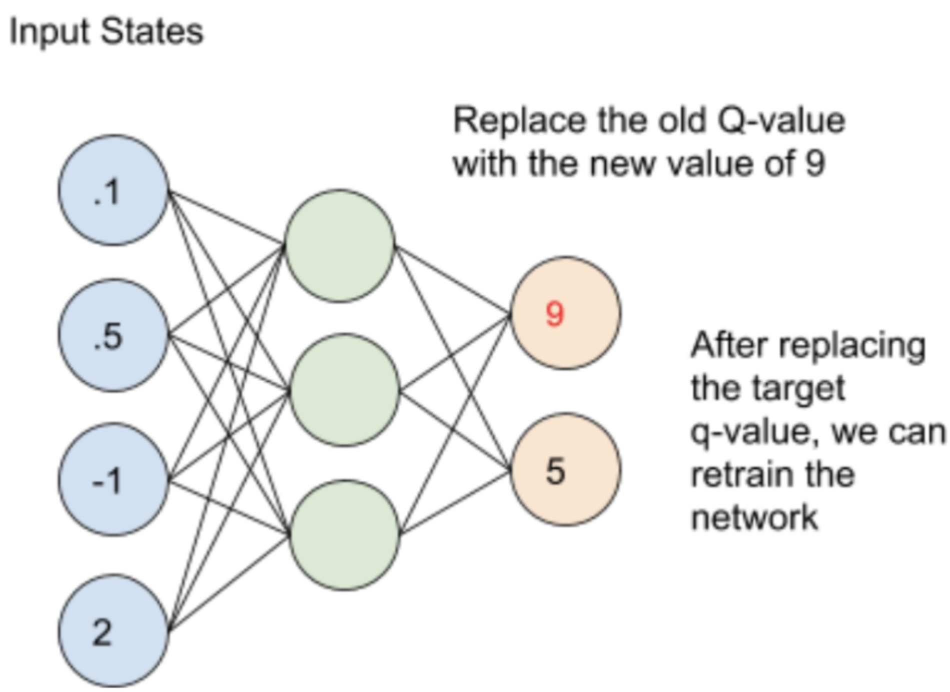
One of the interesting things about Deep Q-Learning is that the learning process uses 2 neural networks. These networks have the same architecture but different weights. Every N steps, the weights from the main network are copied to the target network. Using both of these networks leads to more stability in the learning process and helps the algorithm to learn more effectively. In our implementation, the main network weights replace the target network weights every 100 steps.

Experience Replay is the act of storing and replaying game states (the state, action, reward, next-state) that the RL algorithm is able to learn from. Experience Replay can be used in Off-Policy algorithms to learn in an offline fashion. Off-policy methods are able to update the algorithm's parameters using saved and stored information from previously taken actions. Deep Q-Learning uses Experience Replay to learn in small batches in order to avoid skewing the dataset distribution of different states, actions, rewards, and next-states that the neural network will see. Importantly, the agent doesn't need to train after each step. In our implementation, we use Experience Replay to train on small batches once every 4 steps rather than every single step. We found this trick to really help speed up our Deep Q-Learning implementation.


Input States
Each output
node represents
an action
.1
.5
-1
2
8
5
The value inside
an output node
is the action’s
q-value

Bellman Equation

Just like with vanilla Q-Learning, the agent still needs to update our model weights according to the Bellman Equation.



$$(R_t + \lambda * max_a Q(S_{t+1}, a))$$

State: One-hot color encoding per node.

```
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
[[0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
```

Action: **Legal** color to choose from. Example: pick color 3 from legal [1, 3, 5, 6]
Reward framework:

- Reward -4 for coloring a node with a new color introduced to the graph.
- Reward 0 for coloring a node with a color that's not new to the graph.
- Reward -10 for coloring with an illegal color.

Experience replay: Array of past 10000 events. That being state for the current node, action, reward, next-state, edge-index.

2 Integer Linear Programming Models

2.1 Classical Assignment Model

The assignment model defines binary variables $x_{i,c}$ indicating whether vertex i is assigned color c. The constraints ensure that each vertex receives exactly one color and that adjacent vertices do not share the same color.

$$\min z_{max} \tag{1}$$

s.t.

$$\sum_{c=1}^{H} x_{i,c} = 1 \qquad \forall i \in V \tag{2}$$

$$x_{i,c} + x_{j,c} \leq 1 \qquad \forall (i,j) \in E; 1 \leq c \leq H \tag{3}$$

$$z_{max} \geq c x_{i,c} \qquad \forall i \in V; 1 \leq c \leq H \tag{4}$$

$$x_{i,c} \in \{0,1\} \qquad \forall i \in V; 1 \leq c \leq H \tag{5}$$

$$z_{max} \in \mathbb{R} \tag{6}$$

where:

- z_{max} represents the maximum possible color.
- H is an upper bound on the number of colors.
- Constraint (2) ensures that each vertex is assigned exactly one color.
- Constraint (3) ensures adjacent vertices do not share the same color.

2.2 Hybrid Partial-Ordering Based ILP Model

The Hybrid Partial Order Method is a novel Integer Linear Programming (ILP) formulation introduced to solve the vertex coloring problem by combining aspects of the classical assignment model and partial ordering approaches. The goal is to minimize the number of colors required to properly color the graph while maintaining a partial ordering structure.

$$\min 1 + \sum_{i=1}^{H} y_{i,q} \tag{1}$$

s.t.

$$z_{v,1} = 0 \qquad \forall v \in V \tag{2}$$

$$y_{H,v} = 0 \qquad \forall v \in V \tag{3}$$

$$y_{i,v} - y_{i+1,v} \geq 0 \qquad \forall v \in V,\ i = 1, \ldots, H-1 \tag{4}$$

$$y_{i,v} + z_{v,i+1} = 1 \qquad \forall v \in V,\ i = 1, \ldots, H-1 \tag{5}$$

$$x_{v,i} = 1 - (y_{i,v} + z_{v,i}) \qquad \forall v \in V, i = 1, \ldots, H-1 \tag{6}$$

$$x_{u,i} + x_{v,i} \leq 1 \qquad \forall (u,v) \in E,\ i = 1, \ldots, H \tag{7}$$

$$y_{i,q} - y_{i,v} \geq 0 \qquad \forall v \in V,\ i = 1, \ldots, H-1 \tag{8}$$

$$y_{i,v}, z_{v,i} \in \{0,1\} \qquad \forall v \in V,\ i = 1, \ldots, H \tag{9}$$

where:

- Because q is singled out to hold the maximum color, the summation of $y_{i,q}$ effectively measures how many colors up to H are actively “dominating” higher indices on q.

- Fix boundary condition for z.
- Fix boundary condition for y.
- Solve monotonicity of y.
- Solve linkage relation between y and z (this and previous constraint also imply the monotonicity of y).
- Tie these partial-order variables to the assignment variables (this is why this model is called hybrid). This is done to reduce the number of non-zero coefficients in the next constraint.
- Impose the color difference constraint
- Ensure node q has the highest color

2.3 Big-M Model

In this alternative formulation, each vertex $u \in V$ is assigned a numeric label x_u, and we introduce a binary variable $y_{u,v}$ for each edge $(u, v) \in E$:

$$y_{u,v} = \begin{cases} 0, & \text{if } x_u - x_v \geq d_{u,v}, \\ 1, & \text{otherwise.} \end{cases}$$

Additionally, let x be a continuous variable that represents an upper bound on all labels x_u.

$$\begin{aligned}
&\min x && && (1)\\
&\text{s.t.} && && \\
&\quad x_u - x_v + (H+1)y_{u,v} \geq 1 && \forall (u,v) \in E && (2)\\
&\quad x_u - x_v + (H+1)y_{u,v} \leq H && \forall (u,v) \in E && (3)\\
&\quad 1 \leq x_u \leq x && \forall u \in V && (4)\\
&\quad x \in \mathbb{R}_+,\, \forall u \in V,\, y_{u,v} \in \{0,1\}, && \forall uv \in E. && (5)
\end{aligned}$$

The objective function (1) is minimized, thus seeking the smallest feasible color range $[1, x]$ that still satisfies the constraints on every edge.

To enforce $|x_u - x_v| \geq 1$, we use two linear inequalities involving the large constant $H + 1$. For each edge (u, v), we have (2) and (3).

When $y_{u,v} = 0$, the first inequality implies $x_u - x_v \geq 1$. If $y_{u,v} = 1$, then effectively $x_u - x_v \leq -1$, ensuring $|x_u - x_v| \geq 1$. We also set $x_u \leq x$ for each u, and we require $x_u \geq 1$.

3 Experimental results

The graphs used in the experiments are taken from the work of Trick, M.A., 2002, titled "Computational Symposium: Graph Coloring and Its Generalization." These graphs are used to evaluate the performance of graph coloring algorithms. The table below summarizes the solutions and experimental results for the presented models.

Filename	Graph		GCA		GCMO		Q_GNN_50	
	$\|V\|$	$\|E\|$	k^*	Avg Time [s]	k^*	Avg Time [s]	k^*	Avg Time [s]
1-FullIns_4.col	93	593	5	551.927	5	**220.613**	10	0.0432
1-FullIns_5.col	282	3247	-	**-**	-	-	13	0.2366
2-FullIns_4.col	212	1621	6	1542.793	6	**1405.000**	12	0.2386
3-FullIns_3.col	80	346	6	79.358	6	**35.660**	10	0.0304
3-FullIns_4.col	405	3524	7	6860.887	7	**40189.066**	12	0.3405
4-FullIns_3.col	114	541	7	74.884	7	**224.230**	10	0.04972
4-FullIns_4.col	690	6650	8	**139707.639**	8	326228.070	15	0.7301
4-Insertions_3.col	79	156	4	1500.394	4	891.687	4	**0.045**
5-FullIns_3.col	154	792	8	**156.001**	8	171.058	10	0.1124
DSJC125.1.col	125	736	5	**1481.863**	5	336.751	9	0.1001
DSJC125.5.col	125	3891	-	-	-	-	28	0.1273
DSJC125.9.col	125	6961	-	-	-	-	-	-
DSJC250.1.col	250	3218	-	-	-	-	-	-
anna.col	138	986	11	49.938	11	**38.422**	12	0.1034
david.col	87	812	11	**16.208**	11	69.604	12	0.0592
fpsol2.i.2.col	451	8691	30	8647.931	30	**18091.708**	40	0.5674
fpsol2.i.3.col	425	8688	30	**10971.131**	30	22261.259	39	0.5319
games120.col	120	1276	9	**30.828**	9	45.411	10	0.0917
homer.col	561	3258	-	-	-	-	19	0.3158
huck.col	74	602	11	17.492	11	50.015	11	**0.0427**
jean.col	80	508	10	19.249	10	34.108	10	**0.0567**
le450_15a.col	450	8168	-	-	-	-	23	0.5641
le450_15b.col	450	8169	15	**293544.982**	-	-	23	0.8354
le450_25a.col	450	8260	25	4797.352	25	**16937.937**	30	0.8444
le450_25b.col	450	8263	25	**1109.989**	25	6593.416	30	0.856
le450_5a.col	450	5714	5	91082.342	5	**132220.429**	14	0.5794
le450_5b.col	450	5734	5	**101281.902**	5	330164.498	14	0.5934
miles1000.col	128	6432	42	5535.919	42	**52164.028**	43	0.1001
miles250.col	128	774	8	**24.868**	8	48.515	10	0.0784
miles500.col	128	2340	20	55.109	20	**130.001**	22	0.1115
miles750.col	128	4226	31	722.343	31	**1036.207**	34	0.09801
mug100_1.col	100	166	4	464.081	4	**227.726**	6	0.0233
mug100_25.col	100	166	4	504.788	4	**256.657**	6	0.0220
mulsol.i.1.col	197	3925	49	**5354.243**	49	8128.514	49	0.1796
mulsol.i.2.col	188	3885	31	773.851	31	**1722.966**	36	0.1825
mulsol.i.3.col	184	3916	31	**1465.258**	31	2615.020	36	0.1763
mulsol.i.4.col	185	3946	31	**1460.488**	31	2159.745	36	0.1771
mulsol.i.5.col	186	3973	31	**1443.901**	31	1470.731	36	0.1710
myciel3.col	11	20	4	8.612	4	96.649	4	**0.0045**
myciel4.col	23	71	5	253.726	5	**239.840**	6	0.0067
myciel5.col	47	236	6	**1238.089**	6	1260.944	7	0.0162
myciel6.col	95	755	-	**-**	-	-	10	0.0327
myciel7.col	191	2360	-	**-**	-	-	9	0.113
queen10_10.col	100	2940	-	-	-	-	21	0.0633
queen11_11.col	121	3960	-	-	-	-	23	0.0723
queen12_12.col	144	5192	-	-	-	-	23	0.1179
queen13_13.col	169	6656	-	-	-	-	25	0.1637
queen14_14.col	196	8372	-	-	-	-	28	0.1947
queen5_5.col	25	320	5	**1.621**	5	2.311	6	0.0097
queen6_6.col	36	580	7	330.278	7	**236.386**	10	0.0146
queen7_7.col	49	952	7	**392.637**	7	434.787	12	0.0233
queen8_12.col	96	2736	12	658.831	12	**614.457**	19	0.0808
queen8_8.col	64	1456	9	**2397.146**	9	5499.739	16	0.0382
queen9_9.col	81	2112	10	**170747.021**	-	-	20	0.0542
zeroin.i.1.col	211	4100	49	**2705.986**	49	7396.734	49	0.1451
zeroin.i.2.col	211	3541	30	**2299.254**	30	3689.648	39	0.1357
zeroin.i.3.col	206	3540	30	**2416.741**	30	3761.858	39	0.1265

Table 1: Performance of Graph Coloring Methods with Predicted Chromatic Number K and Runtime (s)

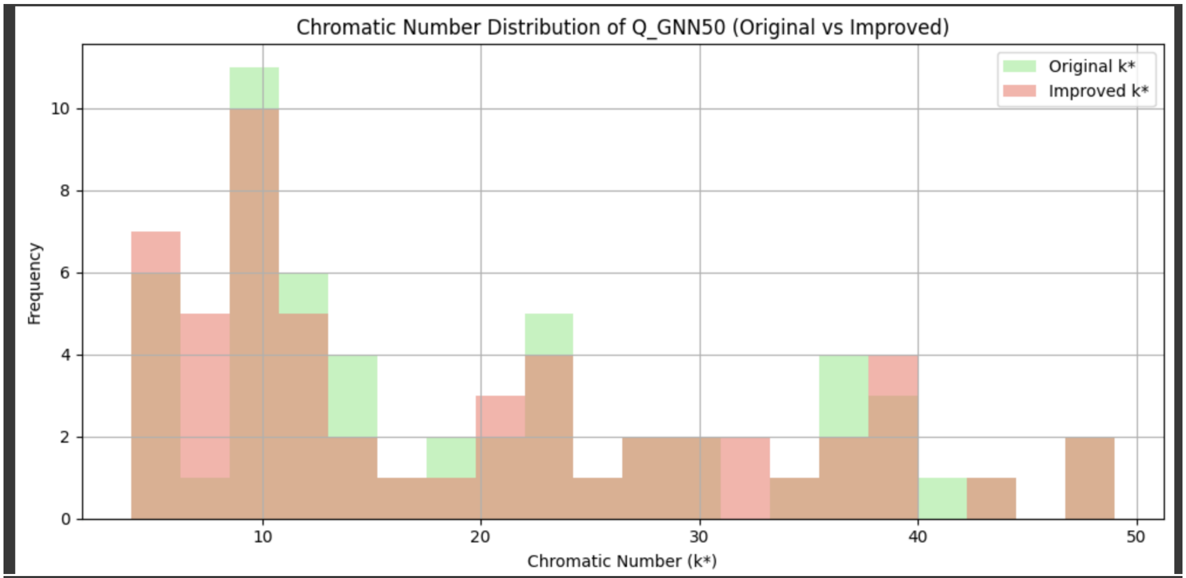
Filename	Graph		GCA		GCMO		Q_GNN_50	
	$\lvert V \rvert$	$\lvert E \rvert$	k^*	Avg Time [s]	k^*	Avg Time [s]	k^*	Avg Time [s]
1-FullIns_4.col	93	593	5	551.927	5	**220.613**	8	0.140
1-FullIns_5.col	282	3247	-	**-**	-	-	13	1.3570
2-FullIns_4.col	212	1621	6	1542.793	6	**1405.000**	9	0.6797
3-FullIns_3.col	80	346	6	79.358	6	**35.660**	7	0.1126
3-FullIns_4.col	405	3524	7	6860.887	7	**40189.066**	12	2.1627
4-FullIns_3.col	114	541	7	74.884	7	**224.230**	10	0.04972
4-FullIns_4.col	690	6650	8	**139707.639**	8	326228.070	12	6.1669
4-Insertions_3.col	79	156	4	1500.394	4	891.687	4	**0.045**
5-FullIns_3.col	154	792	8	**156.001**	8	171.058	8	0.2771
DSJC125.1.col	125	736	5	**1481.863**	5	336.751	8	0.2219
DSJC125.5.col	125	3891	-	-	-	-	28	0.1273
DSJC125.9.col	125	6961	-	-	-	-	-	-
DSJC250.1.col	250	3218	-	-	-	-	-	-
anna.col	138	986	11	49.938	11	**38.422**	11	0.2294
david.col	87	812	11	**16.208**	11	69.604	11	0.1489
fpsol2.i.2.col	451	8691	30	8647.931	30	**18091.708**	38	5.0611
fpsol2.i.3.col	425	8688	30	**10971.131**	30	22261.259	39	0.5319
games120.col	120	1276	9	**30.828**	9	45.411	9	0.20389
homer.col	561	3258	-	-	-	-	18	2.1759
huck.col	74	602	11	17.492	11	50.015	11	**0.0427**
jean.col	80	508	10	19.249	10	34.108	10	**0.0567**
le450_15a.col	450	8168	-	-	-	-	23	0.5641
le450_15b.col	450	8169	15	**293544.982**	-	-	23	0.8354
le450_25a.col	450	8260	25	4797.352	25	**16937.937**	30	0.8444
le450_25b.col	450	8263	25	**1109.989**	25	6593.416	30	0.856
le450_5a.col	450	5714	5	91082.342	5	**132220.429**	10	1.5660
le450_5b.col	450	5734	5	**101281.902**	5	330164.498	10	1.5986
miles1000.col	128	6432	42	5535.919	42	**52164.028**	43	0.1001
miles250.col	128	774	8	**24.868**	8	48.515	9	0.1369
miles500.col	128	2340	20	55.109	20	**130.001**	21	0.1613
miles750.col	128	4226	31	722.343	31	**1036.207**	34	0.09801
mug100_1.col	100	166	4	464.081	4	**227.726**	6	0.0233
mug100_25.col	100	166	4	504.788	4	**256.657**	6	0.0220
mulsol.i.1.col	197	3925	49	**5354.243**	49	8128.514	49	0.1796
mulsol.i.2.col	188	3885	31	773.851	31	**1722.966**	33	0.4674
mulsol.i.3.col	184	3916	31	**1465.258**	31	2615.020	33	0.4612
mulsol.i.4.col	185	3946	31	**1460.488**	31	2159.745	36	0.1771
mulsol.i.5.col	186	3973	31	**1443.901**	31	1470.731	36	0.1710
myciel3.col	11	20	4	8.612	4	96.649	4	**0.0045**
myciel4.col	23	71	5	253.726	5	**239.840**	5	0.0381
myciel5.col	47	236	6	**1238.089**	6	1260.944	6	0.0569
myciel6.col	95	755	-	**-**	-	-	10	0.0327
myciel7.col	191	2360	-	**-**	-	-	9	0.113
queen10_10.col	100	2940	-	-	-	-	21	0.0633
queen11_11.col	121	3960	-	-	-	-	23	0.0723
queen12_12.col	144	5192	-	-	-	-	23	0.1179
queen13_13.col	169	6656	-	-	-	-	25	0.1637
queen14_14.col	196	8372	-	-	-	-	28	0.1947
queen5_5.col	25	320	5	**1.621**	5	2.311	5	0.0423
queen6_6.col	36	580	7	330.278	7	**236.386**	8	0.0593
queen7_7.col	49	952	7	**392.637**	7	434.787	10	0.0691
queen8_12.col	96	2736	12	658.831	12	**614.457**	15	0.1899
queen8_8.col	64	1456	9	**2397.146**	9	5499.739	16	0.0382
queen9_9.col	81	2112	10	**170747.021**	-	-	20	0.0542
zeroin.i.1.col	211	4100	49	**2705.986**	49	7396.734	49	0.1451
zeroin.i.2.col	211	3541	30	**2299.254**	30	3689.648	39	0.1357
zeroin.i.3.col	206	3540	30	**2416.741**	30	3761.858	39	0.1265

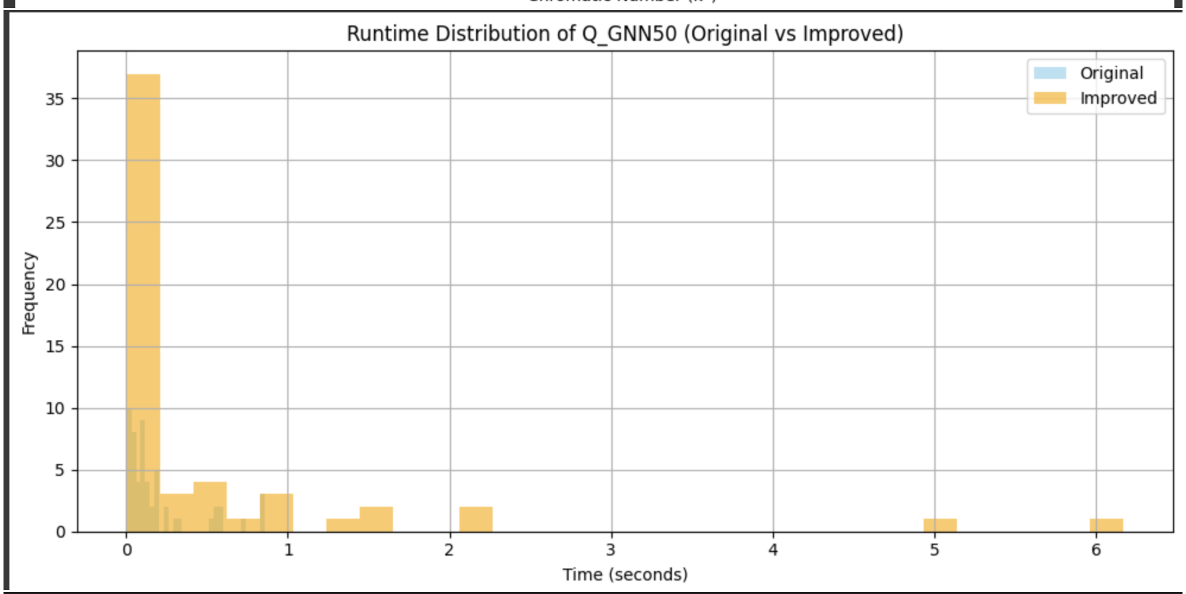
Table 2: Improvements - Performance of Graph Coloring Methods with Predicted Chromatic Number K and Runtime (s)

Filename	Graph		GCA		GCMO		BigM		Q_GNN_50		ReLcol	
	$\|V\|$	$\|E\|$	k^*	Avg Time [s]	k^*	Avg Time [s]	k^*	Avg Time [s]	k^*	Avg Time [s]	k^*	Avg Time [s]
queen5 5_.col	25	320	5	1.181	5	1.175	5	81.023	5	0.0423	5	-
queen6 6_.col	36	580	7	27.832	7	10.929	7	3602.144	8	0.0593	7	-
myciel5_.col	47	236	6	1238.089	6	1260.944	6	3603.082	6	0.0569	6	-
queen7 7_.col	49	952	7	338.174	7	27.256	-	-	10	0.0691	7	-
queen8 8_.col	64	1456	9	2397.146	9	5499.739	-	-	16	0.0432	9	-
1-Insertions 4_.col	701	7065	-	-	-	-	-	-	-		4	-
huck_.col	74	602	11	4.549	11	3.043	11	3602.789	11	0.0427	11	-
jean_.col	80	508	10	4.341	10	2.441	10	3600.217	10	0.0567	10	-
queen9 9_.col	81	2112	10	170747.021	-	-	-	-	20	0.0542	10	-
david_.col	87	812	11	14.245	11	6.675	11	3602.376	11	0.1489	11	-
mug88 1_.col	88	146	4	33.102	4	5.912	4	3600.143	-	-	4	-
myciel6_.col	95	755	7	3599.447	7	3600.133	7	3600.498	10	0.0327	7	-
games120_.col	120	1276	9	4.328	9	3.212	9	3601.716	9	0.20389	9	-
queen8 12_.col	96	2736	12	658.831	12	501.890	-	-	15	0.0691	12	-
queen11 11_.col	121	3960	-	-	-	-	-	-	23	0.0723	11	-
anna_.col	138	986	11	12.771	11	10.868	11	3603.031	11	0.0427	11	-
2-Insertions 4_.col	149	541	-	-	-	-	-	-	-	-	4	-
queen13 13_.col	169	6656	-	-	-	-	-	-	25	0.1637	13	-
myciel7_.col	191	2360	8	3597.965	8	3598.875	8	3600.789	9	0.113	8	-
homer_.col	561	3258	-	-	-	-	-	-	18	2.1759	13	-

Table 3: Comaparison table with reference paper

4 Tests






```
=== Runtime Comparison ===
Paired t-test: t = -2.5606, p = 0.01328
Wilcoxon signed-rank: W = 0.0, p = 1.228e-05

=== Chromatic Number Comparison ===
Paired t-test: t = 5.1169, p = 4.232e-06
Wilcoxon signed-rank: W = 0.0, p = 2.148e-05
```

References

[1] Olivier Goudet, Cyril Grelier, and Jin-Kao Hao. A deep learning guided memetic framework for graph coloring problems. *arXiv e-prints*, page arXiv:2109.05948, September 2021.

[2] Olivier Goudet, Cyril Grelier, and Jin-Kao Hao. A deep learning guided memetic framework for graph coloring problems. *Knowledge-Based Systems*, 258:109986, 10 2022.

[3] Anar Jabrayilov and Petra Mutzel. New integer linear programming models for the vertex coloring problem. *European Journal of Operational Research*, 257(1):47–58, 2017.

[4] Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luís Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. *arXiv e-prints*, pages 879–885, 11 2019.

[5] Sanjay Mehrotra and Michael A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4):344–354, 1996.