FINDING GRAPH CHROMATIC NUMBER THROUGH DEEP LEARNING

S. Kholkin, A. Filimonov

National Research Lobachevsky State University of Nizhny Novgorod
603950, Nizhny Novgorod, Russia

DOI: 10.24412/2073-0667-2022-1-42-54

Deep learning in recent years has become state-of-the-art type of algorithms in various spheres such as computer vision and natural language processing. Number of deep learning architectures is growing and so is field of processing graphs through these architectures. Graph Neural Network (GNN) is deep learning architecture that have an ability to reflect graph structure and learn various patterns that are effectively used in graph related domains such as drug discovery or traffic forecasting. GNN embeds vertices as vectors that reflect vertex related features that are updated by function that depends on vertex neighbors, in some ways that can be interpreted as message passing between vertices, also update function is permutation invariant what gives GNN unique ability not to depend on vertices order or vertices number, graph of any size can be fed into particular GNN. Many studies on applying deep learning technics for combinatorial optimization is underway and since many combinatorial problems either originally represented in terms of graph theory or can be converted to such, GNN seems to be a natural choice for finding their approximate solutions. Recent studies confirm this by showing great results in solving Travelling Salesman Problem or Boolean Satisfiability Problem. In this paper we train GNN to find chromatic number of graph or to solve decision version of Graph Coloring Problem. Neural Network is trained in supervised manner as a binary classifier, given graph and number of colors model should return a probability that graph can or cannot be colored by given number of colors. Should be noticed that GNN finds chromatic number but doesn't find vertices coloring itself, hence GNN can predict lower chromatic. Model architecture reflect color-vertex relationship by assigning vector embedding to each color and passing them into update embeddings function so they are treated like vertices embeddings. GNN is built in a recurrent manner with adjustable number of timesteps that reflect depth of message passing between vertices. At every timestep of recurrency model updates vertex and colors embeddings through aggregating information from its neighbors and/or colors through recurrent unit that stands as an update function (Long Short-Term Memory network with LayerNorm in our case). For additional expressivity MLP layers were added before recurrent unit at each timestep. After embedding updates vertices embeddings are being fed into classifier MLP to get a probability that graph is able to be colored by given number of colors. For training GNN in supervised manner labeled data is needed. Natural datasets related to graph coloring are unsuitable for training and validation through heterogeneity and small size. Data was generated following phase transition paradigm, for each graph instance in dataset we have a paired one that differs only by one edge but have higher chromatic number and that pair reflects hard to color pair of graphs. Because of this data generation paradigm GNN optimization objective becomes more difficult to reach and hopefully makes GNN more knowledgeable. Two datasets were generated CHROM_40_60 and CHROM_35_45 with 40 to 60 or 35 to 45 number of vertices in graph instance respectively and with 3 to 8 chromatic numbers both. Instances of CHROM_35_45 dataset are supposed to be easier to color than CHROM_40_60 ones. GNN was trained using that data and compared to heuristics:

tabucol and greedy. Percentage of correctly predicted graphs chromatic numbers were taken as a comparison metric. Testing on CHROM_40_60 and CHROM_35_45 datasets showed that GNN have slightly better accuracy metric than tabucol: GNN correctly predict 69 % and 77 % of graph chromatic numbers, when tabucol correctly predict 66 % and 76 % of graph chromatic numbers, greedy is far behind. To test GNN generalization ability cross dataset experiments were performed, GNN trained on CHROM_40_60 dataset is validated on CHROM_35_45 dataset and vice versa. These experiments show that model trained on CHROM_40_60 harder instances show decent accuracy on CHROM_35_45 easier instances while model trained on CHROM_35_45 easier instances perform weaker on CHROM_40_60 hard instances but still keeps some accuracy hence we can say that model has an ability to generalize to graphs with other number of vertices. Testing on natural instances of COLOR02/03/04 dataset which instances differ a lot from generated train/test data was performed. GNN trained on CHROM_40_60 shows accuracy similar to tabucol and much better than greedy. GNN behaves unstable on instances with chromatic number higher that was seen in training dataset. We can say that GNN have an ability to generalize on graphs from other distributions with similar chromatic number and doesn't have an ability to generalize on graphs with unseen chromatic number. Referring to computational complexity execution time measurements show that GNN is approximately 100 times faster than tabucol and 30 times slower than greedy. There are problems with data generation because such algo of data generation have exponential asymptotics and hence data with number of vertices higher than 60 and chromatic number higher than 8 takes very long time that limits GNN train possibilities and performance on higher than 8 chromatic numbers. Addressing further work GNN model can be changed to more expressive, or manner of training could be changed to semi-supervised or supervised/semi-supervised hybrid frameworks that could lead not only to finding chromatic number but to finding vertex coloring itself. In total GNN was trained in supervised manner to find chromatic number of graph so that it have an ability to generalize on graphs with various sizes and shows accuracy on presented data similar to tabucol and at the same is much faster.

Key words: neural network, deep learning, chromatic number, combinatorial optimization.

References

1. Silver D., Schrittwieser J., Simonyan K., Antonoglou I., Huang A., Guez A. et al. Mastering the game of go without human knowledge // Nature. 2017. Vol. 550, N 7676, P. 354.

2. Selsam D., Lamm M., Bunz B., Liang P., de Moura L., and Dill D. Learning a sat solver from single-bit supervision, arXiv preprint arXiv:1802.03685, 2018.

3. Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. 2019. An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227 (2019).

4. Scarselli F., Gori M., Tsoi A., Hagenbuchner M., and Monfardini G. The graph neural network model // IEEE Tran. Neural Networks. 2009. Vol. 20, N 1, P. 61–80.

5. Barnier N. and Brisset P. Graph coloring for air traffic flow management // Annals of Operations Research. Aug 2004. Vol. 130, N 1, P. 163–178.

6. Thevenin S., Zufferey N., and Potvin J. Graph multi-coloring for a job scheduling application // Discrete App. Math., 2018. Vol. 234, P. 218–235.

7. Chen W., Lueh G., Ashar P., Chen K., and Cheng B. Register allocation for intel processor graphics. In CGO 2018, P. 352–364. [Electron. Res.]: `http://doi.acm.org/10.1145/3168806`.

8. Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems // In IEEE International Conference on Tools with Artificial Intelligence (ICTAI). 2019. P. 879–885.

9. [Electron. Res.]: `https://mat.tepper.cmu.edu/COLOR02/`.

10. Sepp Hochreiter, Jurgen Schmidhuber. LONG SHORT-TERM MEMORY // Neural Computation. 1997. N 9 8. P. 1735–1780, [Electron. Res.]: https://www.bioinf.jku.at/publications/older/2604.pdf.

11. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv, abs/1607.06450.

12. Hertz A. and de Werra D. Using tabu search techniques for graph coloring // Computing. Dec. 1987. Vol. 39, N 4, P. 345–351. [Electron. Res.]: http://dx.doi.org/10.1007/BF02239976.

НАХОЖДЕНИЕ ХРОМАТИЧЕСКОГО ЧИСЛА ГРАФА С ПОМОЩЬЮ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ

С. Д. Холькин, А. В. Филимонов

Нижегородский государственный университет им. Н. И. Лобачевского
603950, Нижний Новгород, Россия

УДК 004.855.5
DOI: 10.24412/2073-0667-2022-1-42-54

Алгоритмы глубокого обучения сильно развились в последнее десятилетие и стали стандартом во многих сферах. Притом количество архитектур глубокого обучения растет и существуют модели, работающие со структурой графа Graph Neural Network или GNN, которые показали свою эффективность в различных доменах. Также глубокое обучение применяют и для решения задач комбинаторной оптимизации. Поскольку многие задачи комбинаторной оптимизации изначально формулируются в терминах теории графов или же могут быть конвертированы в подобное представление, то архитектура GNN может стать эффективным методом для их приблизительного решения. В этой работе рассматривается задача о нахождении хроматического числа графа и ее приблизительное решение с помощью GNN. Вершины и цвета, в которые предположительно можно раскрасить граф, задаются случайными эмбеддингами, далее GNN, с учетом структуры графа, преобразовывает все эмбеддинги и производит на их основе бинарную классификацию, может граф быть раскрашен в данное количество цветов или нет. Данные для обучения сети являются сгенерированными и представляют собой сложные случаи раскраски. Также для тестирования обобщенности приведены замеры на данных, сильно отличающихся от тренировочных. Натренированная на синтетических данных GNN сравнивается по точности и времени исполнения с эвристиками: tabucol и жадный алгоритм.

Ключевые слова: нейронная сеть, глубокое обучение, хроматическое число графа, комбинаторная оптимизация.

Введение. В последнее время стал активно развиваться класс алгоритмов приблизительного решения задач, основанных на машинном обучении (Machine Learning), а также подкласс самых тяжеловесных из них — алгоритмов на основе глубокого обучения (Deep Learning). Они уже показали множество удивительных результатов, например глубокая нейронная сеть AlphaGo [1] смогла победить профессионала мирового уровня в игру Го, также глубокие нейронные сети показали большие успехи в компьютерном зрении и обработке естественного языка. Есть также попытки применения глубоких нейронных сетей для комбинаторной оптимизации, к примеру для решения задачи о выполнении булевых формул (SAT) [2] или для решения задачи о коммивояжере (TSP) [3]. Также недавно начали очень активно развиваться графовые нейронные сети GNN [4], которые работают со структурой графа, что открывает новые возможности в комбинаторной оптимизации с помощью глубокого обучения.

В этой работе будет рассмотрена задача о нахождении хроматического числа графа. У этой задачи есть применение, например, при построении расписания в аэропортах [5], при

работе планировщика задач [6] или при распределении регистров [7]. Однако сложность этой задачи имеет экспоненциальную асимптотику, поэтому эта задача преимущественно решается приблизительно, используя различные эвристики с полиномиальной асимптотикой.

Архитектура представленной нейронной сети основывается на [8], проверяется на синтетически сгененированных наборах примеров раскраски и ограниченном количестве примеров раскраски из Color02/03/04 датасета [9], и, наконец, сравнивается с другими эвристиками, которые также решают задачу GCP.

1. Формальная постановка задачи. Найти такое минимальное $C|C \in N \vee C > 0$ для графа $G(V, E)$, что

$$\exists f: \ v \to c, v \ \in V, c \in \{1, 2..C\}, \ \neg\exists \ i, j \ \ v_i,v_j \ \to \ c, e_{ij} \in E.$$

В терминологии машинного обучения: дан граф G(V, E) и, перебирая C от 2 до $|V|$, нужно выбрать минимальное количество цветов C, которое будет классифицировано моделью как возможное для C-раскраски G(V, E). При этом сама раскраска, то есть отображение $V \to \{1, 2, \ldots, C\}$, не находится.

2. Модель. 2.1. *GNN.* GNN — это архитектура нейронной сети, которая работает над графом G(V, E), где у каждой вершины есть векторное представление $\mathbf{E}m(\mathbf{V_i}) \in \mathbf{R^d}| \ \forall \mathbf{V_i} \in \mathbf{V}$, которое итерационно обновляется, используя информацию о структуре графа. Обновление зависит от векторного представления самой вершины, а также от векторных представлений всех ее соседей. Поскольку существует множество изоморфных графов, то обновление $\mathbf{E}m(\mathbf{V_i})$ должно для всех изоморфизмов быть одинаковым, и следующая далее функция обновления это обеспечивает.

Функция обновления:

$$Em\,(V_i)^{t+1} = Update\left(Em\,(V_i)^t, M_{vv} \times Em\,(V)^t\right) \mid \forall V_i \in V,$$

где $Em(V_i)^t \in R^d$ — векторное представление вершины V_i в момент t, $Em(V)^t \in R^{|V|xd}$ — векторные представления всех вершин в момент t, M_{vv} — матрица смежностей графа $G(V, E)$.

Также стоит заметить, что приведенная выше функция обновления не зависит от количества вершин в графе, то есть одна и та же конкретная GNN может работать на графах любого размера, что и будет далее использовано в экспериментах.

В этой работе будет использоваться несколько модифицированная версия GNN. Чтобы имитировать зависимость векторных представлений вершин от цветов, доступных для раскраски графа, вводятся векторные представления цветов раскраски $Em(C_j) \in R^d| \ C_j \in \{1, 2..C\}$ ($Em(C) \in R^{C \ x \ d}$ — для всех цветов вместе подобно векторным представлениям $Em(V_i) \in R^d$ для вершин, а также функция обновления для цветов.

Тогда функции обновления:

$$Em\,(V_i)^{t+1} \ = \ UpdateV(Em\,(V_i)^t, \ M_{vv} \times Em\,(V)^t, \ M_{vc}^T \times C_{msg}(Em(C)^t))| \ \forall V_i \in V$$

$$Em\,(C_j)^{t+1} = UpdateC(Em\,(C_j)^t, \ M_{vc}^T \times V_{msg}(Em\,(V)^t))| \ C_j \in \{1, 2..C\},$$

где $C_{msg} : R^d \to R^d$ — функция, транслирующая векторное представление цветов в сообщение, готовое к обработке UpdateV функцией, $V_{msg} : R^d \to R^d$ функция, транслирующая векторное представление вершин в сообщение, готовое к обработке UpdateC функцией. C_{msg} V_{msg} представлены 3х-слойными перцептронами.

```
float GNN(G(V, E), C)
{
    M_vv[i,j] ← 1 if (∃e ∈ E|e = (V_i,V_j))|∀V_i ∈ V,V_j ∈ V else 0
    M_vc[i,j] ← ∀V_i ∈ V 1 if C_j ∈ {1..C}
    Em(V_i)~N(0,1) | ∀V_i ∈ V
    Em(C_i)~U(0,1) if C_i ∈ {1..C}
    for t = 1 ... t_max do:
        Em(V_i)^{t+1} = UpdateV(Em(V_i)^t, M_vv × Em(V_i)^t, M_vc^T ×
C_msg(Em(C)^t))| ∀V_i ∈ V
        Em(C_j)^{t+1} = UpdateC(Em(C_j)^t, M_vc^T × V_msg(Em(V_i)^t))| C_j ∈ {1,..C}}
    V_log   = V_vote(Em(V)^{t_max})
    prediction = sigmoid(mean(V_logits))
    return prediction
}
```

Рис. 1. Алгоритм GNN

$$UpdateV : R^{3d} \to R^d$$

$$UpdateC : R^{2d} \to R^d$$

Функции UpdateV и UpdateC являются рекуррентными нейронными сетями (RNN), что и позволяет алгоритму быть итерационным, в нашем же конкретном случае это были сети, состоящие каждая из одного LSTM блока с LayerNorm [10, 11]. Таким образом сеть выучивает структурные паттерны в предоставленных графах, и далее, после обучения, в соответствии с этими паттернами выстраивает векторные представления вершин, на основе которых далее происходит классификация графа с помощью отдельной функции $V_{vote} : R^d \to R$, так мы получаем „голоса“ всех вершин, а далее ищем среднее $mean(V_{vote})$ и превращаем его в вероятность $sigmoid(mean(V_{vote}))$, которая является вероятностью того, можно ли раскрасить граф G(V, E) с помощью C цветов. Полный алгоритм на псевдо языке представлен на рис. 1, а блочная визуализация нейросети показана на рис. 2. Поскольку модель решает задачу бинарной классификации графа, то функцией потерь является бинарная кросс-энтропия.

2.2. *Датасет.* Для тренировки глубокой нейронной сети необходим датасет, то есть набор данных, на которых будет осуществляться обучение и тестирование GNN. Пригодных датасетов, состоящих из естественных графов, полученных из какой-либо предметной области, не было замечено, поэтому придется генерировать датасет. Однако просто так сгенерировать графы (например, с помощью модели Эрдеша-Реньи) и раскрасить не получится, так как эти случаи будут достаточно легкими для раскраски. Увеличить сложность раскраски можно с помощью особой модели генерации, основанной на фазовых сдвигах [8]. Берется тот или иной сгенерированный граф, раскрашивается с помощью прямого алгоритма, а далее находится и добавляется такое ребро, что меняет раскраску,

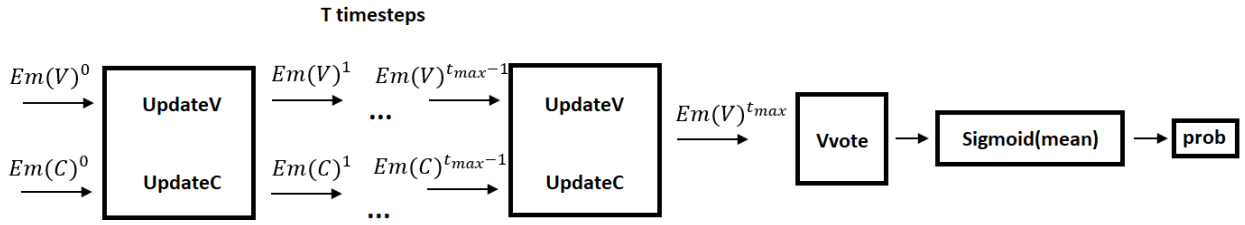

Рис. 2. Схема GNN

то есть увеличивает хроматическое число графа. Таким образом получается пара графов, которая описывает сложный случай раскраски.

2.3. *Конечная конфигурация модели.* Для обучения описанной выше модели был сгенерирован датасет CHROM_40_60, состоящий из 15710 тренировочных и 1024 тестовых пар графов, где $\mathbf{40} \leqslant |\mathbf{V}| \leqslant \mathbf{60}$ и $\mathbf{3} \leqslant \chi \leqslant \mathbf{8}$, а также датасет CHROM_35_45, состоящий из 8000 тренировочных и 1000 тестовых пар графов, где $\mathbf{35} \leqslant |\mathbf{V}| \leqslant \mathbf{45}$ и $\mathbf{3} \leqslant \chi \leqslant \mathbf{8}$.

Модель на обоих датасетах обучалась с количеством внутренних итераций $timesteps = 32$ и размерностью векторного представления в $d = 64$, в течение около 1000 эпох с батчем $batchsize = 16$, имея по 128 итераций в одной эпохе, с экспоненциальным learning rate расписанием от $lr = 1e{-}04$ $lr = 1e{-}06$. Метрика успешности обучения модели (точности):

$$acc = \frac{\sum_i^n f\left(ground_truth_i, pred_i\right)}{n},$$

где $groundtruth_i = 1$, {если $G(V, E)$ можно покрасить в C цветов, иначе 0}, $f(x, y) = \{1$, если $x = y$, иначе $0\}$, $pred_i$ — вероятность того, что $G(V, E)$ можно покрасить в C цветов, n — количество примеров в выборке.

3. Результаты экспериментов. 3.1. *Эвристики для сравнения.* Существует множество эвристик для поиска хроматического числа, но здесь будут рассматриваться две из них: жадный поиск и tabucol [12].

Жадный поиск „жадно“ раскрашивает вершины графа в доступные цвета. Алгоритм очень простой, но не очень эффективный в смысле точности получаемого решения.

Tabucol [12] — это эвристика локального поиска, основанная на поиске с запретами (табу-поиск).

3.2. *Синтетический датасет.* Процедура тренировки на датасете CHROM_40_60 была остановлена, когда модель достигла точности acc=0.706 на тренировочных данных и acc=0.681 на тестовых данных, а процедура тренировки на датасете CHROM_35_45 была остановлена, когда модель достигла точности в acc=0.761 на тренировочных данных и acc=0.756 на тестовых данных.

Далее приведены более подробные результаты работы алгоритмов на тестовых данных для CHROM_40_60 (см. табл. 1, 2, 3) и CHROM_35_45 (см. табл. 4, 5, 6). По вертикали изображены хроматические числа, предсказанные алгоритмом, а по горизонтали настоящие хроматические числа. На диагонали, подсвеченной зеленым, располагаются правильные ответы (предсказание алгоритма совпадает с тем, что есть на самом деле), соответственно все, что выше или ниже — это ошибка. Как метрика для общей оценки качества алгоритма используется Mean Absolute Error (MAE) или Среднее Абсолютное Отклонение.

Таблица 1

Результаты работы алгоритма GNN для CHROM_40_60

10	0,000	0,000	0,000	0,000	0,000	0,000
9	0,000	0,000	0,000	0,000	0,000	0,000
8	0,000	0,000	0,000	0,000	0,065	0,694
7	0,000	0,000	0,000	0,116	0,630	0,306
6	0,000	0,000	0,180	0,562	0,306	0,000
5	0,000	0,168	0,614	0,322	0,000	0,000
4	0,081	0,801	0,206	0,000	0,000	0,000
3	0,919	0,031	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,311					

Таблица 2

Результаты работы алгоритма tabucol для CHROM_40_60

10	0,000	0,000	0,000	0,000	0,000	0,000
9	0,000	0,000	0,000	0,000	0,000	0,000
8	0,000	0,000	0,000	0,000	0,065	1,000
7	0,000	0,000	0,000	0,116	0,847	0,000
6	0,000	0,000	0,180	0,651	0,000	0,000
5	0,000	0,168	0,550	0,000	0,000	0,000
4	0,081	0,620	0,000	0,000	0,000	0,000
3	0,774	0,000	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,342					

Как видно, жадный алгоритм очень сильно отстает по качеству от остальных алгоритмов. В общем tabucol и GNN имеют схожее качество, но стоит заметить, что GNN лучше работает на графах с низким хроматическим числом и хуже на графах с большим хроматическим числом. Также, поскольку tabucol и жадный алгоритм ищут полную раскраску и по ней определяют хроматическое число, то они не ошибаются в сторону уменьшения оценки хроматического числа, в отличие от GNN, тенденцию которой ошибаться в сторону уменьшения хорошо видно на графах с большими хроматическими числами. Также точность в 0.681 может показаться мало впечатляющей из-за того, что точность случайного выбора будет составлять 0.5, однако, как показывают приведенные выше наблюдения (см. табл. 1–6), GNN в результате ошибается только на +/- 1 относительно хроматического числа, что говорит о том, что GNN действительно понимает, какое примерно хроматическое число у графа.

3.3. *Кросс-датасет эксперименты.* Поскольку GNN является гибкой относительно размера входного графа нейросетью, то отдельный вопрос составляет обобщаемость GNN, натренированная на графах одного размера на графы с другим размером. Чтобы это проверить, мы можем взять GNN, натренированную на CHROM_40_60 данных, и протестировать ее на данных из CHROM_35_45 и наоборот (см. табл. 7).

Таблица 3

Результаты работы алгоритма greedy для CHROM_40_60

10	0,000	0,000	0,000	0,038	0,190	0,222
9	0,000	0,000	0,030	0,259	0,532	0,167
8	0,000	0,006	0,274	0,463	0,241	0,000
7	0,000	0,094	0,302	0,229	0,005	0,000
6	0,008	0,380	0,354	0,011	0,000	0,000
5	0,407	0,275	0,039	0,000	0,000	0,000
4	0,581	0,246	0,000	0,000	0,000	0,000
3	0,004	0,000	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,924					

Таблица 4

Результаты работы алгоритма GNN для CHROM_35_45

10	0,000	0,000	0,000	0,000	0,000	0,000
9	0,000	0,000	0,000	0,000	0,000	0,000
8	0,000	0,000	0,000	0,000	0,065	0,690
7	0,000	0,000	0,000	0,116	0,630	0,310
6	0,000	0,000	0,180	0,562	0,306	0,000
5	0,000	0,168	0,614	0,322	0,000	0,000
4	0,081	0,801	0,206	0,000	0,000	0,000
3	0,919	0,031	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,225					

Как видно, GNN, тренированная на данных CHROM_40_60, показывает себя на тестовых данных CHROM_35_45 даже лучше, чем специально натренированная на CHROM_35_45 данных сеть, что может говорить о сильной способности к обобщаемости моделей, тренированных на более сложных данных. GNN же, натренированная на CHROM_35_45 данных, имеет качество на CHROM_40_60 данных значительно меньшее, чем тренированная на CHROM_40_60 данных сеть, однако некоторое качество все равно сохраняется, и это означает: знания, полученные с тренировки на CHROM_35_45, переносятся на CHROM_40_60, однако только частично.

3.4. *COLOR02/03/04 datacet.* Чтобы посмотреть, как натренированная на синтетических данных GNN ведет себя на графах, сильно отличающихся от тех, что она видела при тренировке, проводились замеры на выборочных графах из публично открытого датасета Color02/03/04 [9] (см табл. 8).

Как видно, GNN неплохо показывает себя на графах с количеством вершин больше, чем у тренировочных графов, и выдает предсказания, близкие к реальности, однако еще чаще начинает ошибаться в сторону занижения оценки. По подсчитанной MAE также можно заметить, что жадный алгоритм сильно отстает, и GNN более консистентно сравнивать с tabucol. Хотя у GNN MAE меньше, чем у tabucol, из-за нестабильности выборки нельзя сказать, что на этих примерах GNN работает лучше.

Таблица 5

Результаты работы алгоритма tabucol для CHROM_35_45

10	0,000	0,000	0,000	0,000	0,000	0,000
9	0,000	0,000	0,000	0,000	0,000	0,010
8	0,000	0,000	0,000	0,000	0,422	0,990
7	0,000	0,000	0,000	0,174	0,578	0,000
6	0,000	0,000	0,324	0,826	0,000	0,000
5	0,000	0,251	0,676	0,000	0,000	0,000
4	0,068	0,749	0,000	0,000	0,000	0,000
3	0,932	0,000	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,239					

Таблица 6

Результаты работы алгоритма greedy для CHROM_35_45

10	0,000	0,000	0,000	0,011	0,295	0,583
9	0,000	0,000	0,010	0,093	0,342	0,205
8	0,000	0,000	0,169	0,418	0,261	0,013
7	0,000	0,059	0,356	0,455	0,025	0,000
6	0,008	0,375	0,383	0,024	0,000	0,000
5	0,286	0,285	0,082	0,000	0,000	0,000
4	0,676	0,281	0,000	0,000	0,000	0,000
3	0,030	0,000	0,000	0,000	0,000	0,000
2	0,000	0,000	0,000	0,000	0,000	0,000
pred/color	3	4	5	6	7	8
MAE	0,887					

Таблица 7

Результаты

Train data	Test data	Train acc	Test acc
CHROM_40_60	CHROM_35_45	0,7060	0,7660
CHROM_35_45	CHROM_40_60	0,7610	0,6348

К сожалению, работа GNN на графах с большими хроматическим числами оказалась очень нестабильной ввиду того, что сами графы очень разные и GNN при тренировке не видела ничего подобного, поэтому результаты не были представлены и, можно сказать, что на графах с хроматическими числами больше того, что она видела на тренировке, GNN работает плохо.

Ввиду экспериментов на COLOR02/03/04 (см табл. 8) и кросс-датасет экспериментов (см. табл. 7) можно сделать вывод, что GNN в некоторых пределах слабо чувствительна к изменению вершин в графах, на которых применяется, пока хроматическое число остается близким к тренировочным примерам.

3.5. *Время работы алгоритмов.* Также был проведен замер времени работы алгоритмов в случае определения хроматического числа графа (см. табл. 9).

Таблица 8

Результаты на Color02/03/04 датасете

Graph	Size	C_numb	GNN	GREEDY	TABUCOL	GNN TIME	GREEDY TIME	TABUCOL TIME
1-FullIns_3	30	4	**4**	8	**4**	0.32	0.0039	2.73
1-Insertions_4	67	5	4	**5**	**5**	0.33	0.0045	15.74
2-FullIns_3	52	5	4	9	**5**	0.32	0.0029	9.51
2-Insertions_3	37	4	3	**4**	**4**	0.30	0.0016	4.62
3-Insertions_3	56	4	3	**4**	**4**	0.30	0.0044	9.72
DSJC125.1	125	5	**5**	8	8	0.37	0.014	58.36
games120	120	9	6	10	**9**	0.37	0.013	108.76
mug88_1	88	4	3	**4**	**4**	0.30	0.0071	22.62
mug88_25	88	4	3	**4**	**4**	0.30	0.007	25.2
myciel3	11	4	**4**	**4**	**4**	0.28	0.0012	0.56
myciel4	23	5	**4**	5	5	0.29	0.0017	2.64
myciel5	47	6	5	**6**	**6**	0.30	0.002	11.0
2-FullIns_4	212	6	**6**	13	15	0.41	0.022	74.3
5-FullIns_3	154	8	6	16	**9**	0.36	0.017	47.3
mug100_1	100	4	3	**4**	**4**	0.31	0.0086	30.44
mug100_25	100	4	3	**4**	**4**	0.32	0.0073	28.30
queen5_5	25	5	6	8	**5**	0.32	0.0017	2.99
queen6_6	36	7	**7**	10	8	0.36	0.0019	11.22
queen7_7	49	7	**8**	10	9	0.36	0.0032	25.04
queen8_8	64	9	**9**	13	12	0.35	0.0032	45.8
		MAE	0.81	2	0.9			
					AVG TIME	0.327	0.009	25.5

Как видно, GNN работает медленнее жадного алгоритма и гораздо быстрее Tabucol (в 145 и 90 раз соответственно). На примере графов из COLOR02/03/04 датасета видна похожая динамика. Вычислительная сложность GNN скалируется линейно относительно размера графа.

3.6. *Ограничения, связанные с данными.* Как уже было подмечено, массив данных для тренировки и тестирования модели был сгенерирован по специальной модели [8]. Однако хроматические числа для графов в этой модели в общем случае определяются с помощью полного перебора, поэтому сгенерировать датасет из графов с количеством вершин более 60 за адекватное время не представляется возможным. И хотя было показано, что GNN может быть скалируема относительно количества вершин, то относительно хроматического числа того же сказать нельзя. Ограниченное количество вершин в генерируемых данных также ограничивает и возможность генерировать качественные примеры раскраски с большими хроматическими числами, что пока что ставит под вопрос применение GNN для графов с большими хроматическими числами.

Заключение. Подход к раскраске графов через глубокое обучение показал себя вполне неплохо. GNN, тренируясь на синтетически сгенерированных данных, показывает себя как на синтетически сгенерированных датасетах, так и на реальных графах с увеличенным размером значительно лучше, чем жадный алгоритм, и сравнимо с эвристикой tabucol. GNN обладает большей вычислительной сложностью по отношению

Таблица 9

Время работы алгоритмов

Алгоритм	Данные	Время, сек
GNN	CHROM_40_60 (Test)	317
Tabucol	CHROM_40_60 (Test)	46071
Greedy	CHROM_40_60 (Test)	10.3
GNN	CHROM_35_45 (Test)	280
Tabucol	CHROM_35_45 (Test)	25206
Greedy	CHROM_35_45 (Test)	7.1

к жадному алгоритму, но меньшей по отношению к tabucol. Также GNN достаточно хорошо скалируется относительно количества вершин в графе, что расширяет ее границы применимости. В итоге GNN, натренированная на специально подготовленных для какой-либо задачи данных с не сильно большим количеством вершин и примерно таким же хроматическим числом, что и в представленных синтетических данных, предположительно, будет показывать себя лучше, чем жадный алгоритм, при этом затрачивая чуть больше времени и сравнимо по качеству с tabucol, при этом работая гораздо быстрее его. В качестве дальнейшего улучшения можно было бы попробовать другие, более мощные или приспособленные конкретно к этой задаче архитектуры графовых нейронных сетей, которые, предположительно, поднимут качество модели, однако, как уже отмечалось выше, существуют ограничения по данным, или можно попробовать найти способ расширить модель не только на решение задачи о поиске хроматического числа, а еще и на нахождение раскраски графа.

Список литературы

1. Silver D., Schrittwieser J., Simonyan K., Antonoglou I., Huang A., Guez A. et al. Mastering the game of go without human knowledge // Nature. 2017. Vol. 550, N 7676, P. 354.

2. Selsam D., Lamm M., Bunz B., Liang P., de Moura L., and Dill D. Learning a sat solver from single-bit supervision, arXiv preprint arXiv:1802.03685, 2018.

3. Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. 2019. An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227 (2019).

4. Scarselli F., Gori M., Tsoi A., Hagenbuchner M., and Monfardini G. The graph neural network model // IEEE Tran. Neural Networks. 2009. Vol. 20, N 1, P. 61–80.

5. Barnier N. and Brisset P. Graph coloring for air traffic flow management // Annals of Operations Research. Aug 2004. Vol. 130, N 1, P. 163–178.

6. Thevenin S., Zufferey N., and Potvin J. Graph multi-coloring for a job scheduling application // Discrete App. Math., 2018. Vol. 234, P. 218–235.

7. Chen W., Lueh G., Ashar P., Chen K., and Cheng B. Register allocation for intel processor graphics. In CGO 2018, P. 352–364. [Electron. Res.]: `http://doi.acm.org/10.1145/3168806`.

8. Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems // In IEEE International Conference on Tools with Artificial Intelligence (ICTAI). 2019. P. 879–885.

9. [Electron. Res.]: `https://mat.tepper.cmu.edu/COLOR02/`.

10. Sepp Hochreiter, Jurgen Schmidhuber. LONG SHORT-TERM MEMORY // Neural Computation. 1997. N 9 8. P. 1735–1780, [Electron. Res.]: https://www.bioinf.jku.at/publications/older/2604.pdf.

11. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv, abs/1607.06450.

12. Hertz A. and de Werra D. Using tabu search techniques for graph coloring // Computing. Dec. 1987. Vol. 39, N 4, P. 345–351. [Electron. Res.]: http://dx.doi.org/10.1007/BF02239976.

Холькин Сергей Денисович — e-mail: kholkinsd@gmail.com; телефон: +7-903-054-66-76.

Студент 4-го курса бакалавриата по направлению „Прикладная информатика“ Нижегородского государственного университета. Научные интересы: искусственный интеллект, глубокое обучение, компрессия глубоких нейронных сетей, комбинаторная оптимизация, машинное обучение на графах.

Kholkin Sergei Denisovich — fourth year bachelor student in the field of „Applied Informatics“ of National Research Lobachevsky State University of Nizhniy Novgorod. Research interests: artificial intelligence, deep learning, compression of deep learning models, combinatorial optimization, graph representation learning.

Филимонов Андрей Викторович — e-mail: andrey.v.filimonov@gmail.com; телефон: +7-903-601-28-87.

Окончил ВМК ННГУ в 2004 году по специальности „Информатика“, в 2008 году защитил кандидатскую диссертацию по теме гиперграфов и дискретной оптимизации. С тех пор продолжает сотрудничество с кафедрой Информатики и автоматизации научных исследований Института информационных технологий, математики и механики ННГУ. Автор более 30 публикаций и 35 патентов на изобретения.

Filimonov Andrey Viktorovich — Graduated from Faculty of Computational Mathematics and Cybernetics Nizhniy Novgorod State University in 2004. In 2008 defended Ph. D. thesis in hypergraphs and discrete optimization. Continues to collaborate with Informatics and Automatization of Scientific Research department in NNSU. Author of more than 35 invention certificates and patterns, published more than 30 papers.

Дата поступления — 01.02.2022