Graph Neural Networks and Policy Gradients to Solve the Maximum Independent Set Problem

Mai Pham Alexander Huang-Menders Jackson Yassin Benjamin Grayzel

Abstract

The Maximum Independent Set (MIS) problem is a classic NP-hard problem with applications across network analysis, scheduling, resource allocation, among other fields. Traditional methods, such as greedy algorithms and integer programming, have limitations either in adaptability to sparse graphs or scalability. In recent years, Graph Neural Networks (GNNs) have shown potential in tackling combinatorial optimization problems, although challenges remain, especially when handling sparse graphs. This project proposes a novel approach that combines GNNs with reinforcement learning (RL) using policy gradients to improve MIS solutions on both dense and sparse graphs. We outline a Markov Decision Process (MDP) framework tailored to the MIS problem, integrating graph structure into the policy network to enhance performance. Evaluation involved benchmarks against established methods, using the Facebook Social Circles dataset from the Stanford Large Network Dataset Collection. We preprocess this dataset to generate both a sparse and dense graph to demonstrate the generality of the approach.

1. Introduction

Combinatorial optimization has a wide range of applications, especially the NP-hard problems such as Traveling Salesman Problem (TSP), Satisfiability (SAT), Maximum Independent Set (MIS), Minimum Vertex Cover (MVC), and Maximum Clique (MC). All of these problems can be easily represented by a graph, meaning graph-based methods are appropriate for their solutions as well as understanding and addressing their computational challenge.

Solutions to the MIS problem have been studied for decades. Traditional methods employ classical algorithmic techniques, such as greedy and randomized algorithms, to achieve a significant advantage in runtime complexity over a brute force approach (Tarjan & Trojanowski, 1977; Luby, 1985; Xiao & Nagamochi, 2017). However, the best-known upper bound of $O(1.1996^n)$ still is prohibitive in most settings.

Due to the high degree of interest in solutions to MIS problem, finding a quality approximate algorithm has been an ongoing area of exploration within theoretical computer science. As Graph Neural Networks have demonstrated strong performance across a variety of other, similar problems, they have been naturally extended to this problem as well. This development has introduced a shift in solving not only the MIS problem, but other combinatorial optimization problems as well (Jin et al., 2024). In addition to classical methods, graph-based machine learning models have shown strong promise in accuracy, generalizability, and scalability to very large graphs. Many novel architectures have been developed to address specific combinatorial optimization problems and have empirically demonstrated a high degree of effectiveness (Vesselinova et al., 2020).

Despite the success of GNN-based methods, several challenges still persist. GNNs often struggle with specific graph structures, particularly sparse graphs, where their performance can lag behind traditional approaches. These challenges are compounded by the lack of scalability and the difficulty of ensuring generalization to unseen graph instances. In order to overcome these obstacles, researchers have introduced reinforcement learning integrations with GNNs (Dai et al., 2018). By framing combinatorial optimization problems within a sequential decision-making context, potential for addressing some of these issues has been shown.

Problem Statement: The MIS problem seeks to find the largest possible set of vertices on a graph such that no two vertices are adjacent. Let G be a graph. A subset $V' \subseteq V(G)$ is called an *independent set* if no two vertices in V' are connected by an edge. The independent set V' is *maximal* if adding any vertex $v \in V(G) \setminus V'$ would make the set no longer independent. A *maximum independent set* is one that has the largest possible size among all independent sets in the graph.

In this study, we propose an integration of GNNs and reinforcement learning within a Markov Decision Process (MDP) framework to address the MIS problem. By evaluating our method on both dense and sparse graphs, we aim to overcome the limitations of existing approaches and advance the state of the art in graph-based optimization.

2. Related Work

Machine learning approaches to solving combinatorial optimization problems face the problem of a lack of quality training data, especially on larger graphs. In order to overcome this limitation, a number of approaches have been proposed.

Semi-Supervised GCNs: Previous work into solving combinatorial optimization problems using semi-supervised learning have shown strong promise. Graph Convolution Networks (GCNs) have demonstrated performance on par with heuristic based solvers, generalizability across different data sets, and ability to scale to large graphs (Li et al., 2018). The central component of this model is a GCN that is trained to predict the likelihood that a particular vertex is included in the MIS. A diverse number of solutions are then generated and a recursive search tree is used to select the best one.

Despite strong performance on some datasets and a high degree of popularity, the GCN model proposed by *Li et al. 2018* does not have reproducible results (Böther et al., 2022). The GCN used in the tree search has been shown to not learn a meaningful representation of the graph and equivalent performance can be achieved by replacing it with random values (Böther et al., 2022). The performance of the tree search is instead based upon algorithmic techniques like graph kernelization to find optimal solutions. A more general issue with this type of approach is that it is unlikely for individuals, even if highly experienced and qualified, to be able to pick out complex patterns and hand-select the most useful properties on large or diverse datasets. Additionally, the need for ground-truth labels makes these models extremely limited in the datasets they can learn on.

Self-Supervised and Unsupervised Approaches: Self-Supervised or unsupervised approaches are attractive as they allow for models to be more broadly applicable across data sets and problem types. However, previous models in this category (Wang & Li, 2023; Schuetz et al., 2021; Brusca et al., 2023) have not demonstrated a significant improvement over other graph-based methods and underperform the greedy and linear programming approach. While they present novelty in their approaches to overcoming the issues of supervised learning through inspiriation in other areas like physics (Schuetz et al., 2021), classical algorithm design (Brusca et al., 2023), and deep learning (Wang & Li, 2023), there is not an empirical basis in their performance to demosntrate that self-supervised or unsupervised methods will allow for better applicability across different data sets or problem times.

Reinforcement learning: Various reinforcement learning methods have been proposed to empower GNNs, including Structure2Vec with Q-learning (Dai et al., 2018), actor-critic (Dong et al., 2023), and others (Darvariu et al., 2024; Kong et al., 2023; Dax et al., 2024). The idea behind reinforcement learning methods in solving combinatorial optimization problems is to learn an algorithm and then make predictions. In many real-world applications, the same optimization problem is repeatedly solved with the only delimiting factor being the data. This allows for learning heuristic algorithms that exploit the structure of the recurring problems. These models have shown a strong ability to learn a variety of different algorithms and perform with a high degree of accuracy. As such, reinforcement learning driven approaches have been applied to a wide variety of combinatorial optimization problems, including MIS. Reinforcement learning faces challenges, though, with reward optimization and exploration-exploitation balance. Additionally, these models rely on large amounts of data in order to properly train which raises questions over generalizability to novel problems.

Our contribution: We attempt to contribute to the three research gaps: Firstly, we solve a constrained combinatorial problem. Many of the studies mentioned above mainly address unconstrained combinatorial optimization problems, such as the TSP, which only focus on the improvement of the solution. The MIS problem, however, takes into account the feasibility of the solution. Secondly, despite these advances, existing solutions to the MIS have notable limitations, especially when applied to sparse graphs, as pointed out by (Angelini & Ricci-Tersenghi, 2022). Our experiments evaluate both dense and sparse cases, where our approach yields consistent results. Lastly, to our knowledge, none of the existing studies in combinatorial optimization leverage the power of Graph Attention Network (GAT) in the policy network for node selection. The closest idea is found in (Ahn et al., 2020) where GCN is used for MIS problem, where the MDP formulation is slightly different from ours (allowing decision deference). Our proposed approach integrates GNNs, in particular, GCN and GAT, with reinforcement learning within a Markov Decision Process framework.

3. Proposed Method

Our approach involves a multi-step pipeline that includes graph pre-processing, construction, and solving the MIS problem through reinforcement learning. We formulate the problem as a Markov Decision Process (MDP). The training loop, visualized in Figure 1, is similar to (Hu et al., 2020). However, they attempt to do classification task, whereas our goal is finding an optimal set (i.e. our reward designs are different).

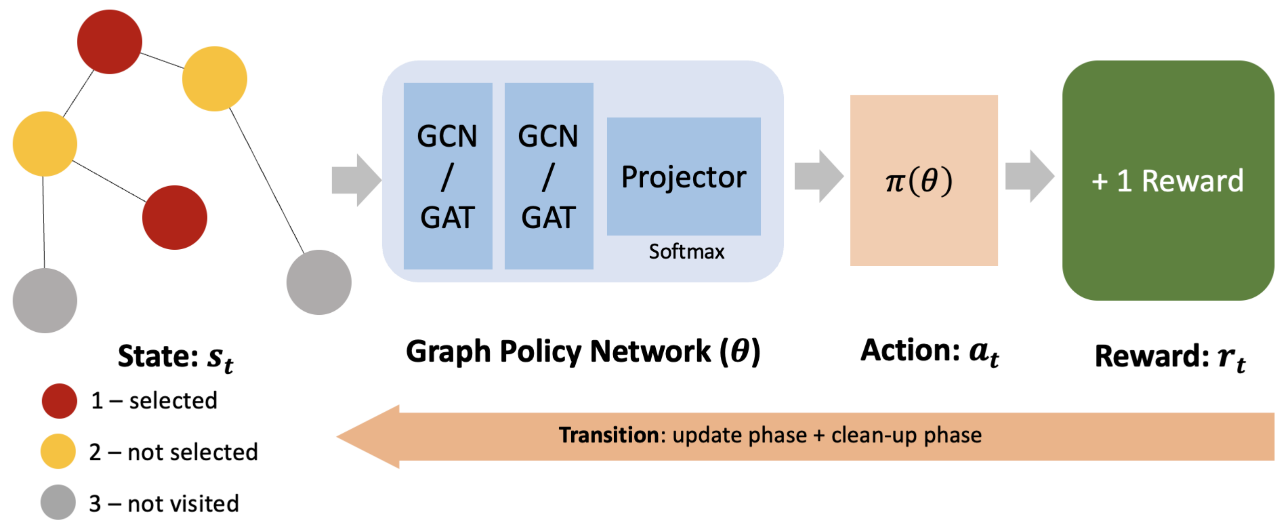

Figure 1. Graph Policy Network Training Procedure

3.1. Data preparation

We noticeably removed isolated nodes (nodes with zero degree). After pre-processing, from the original graph, we sample nodes based on connectivity properties and constructed two sub-datasets: a dense graph dataset and a sparse graph dataset with inductive node sampling. Subsequently, we used random node sampling to create smaller sub-graphs.

3.2. Markov Decision Process

The proposed algorithm to solve the MIS problem is formulated as a MDP and solved using a policy gradient method, see figure 1. At a high level, MDPs model the behavior of an agent interacting with a stochastic environment where states transition based on the actions taken. At each time step t, the agent takes an action, observes the resulting state in a finite state space, and utilizes this information to learn an optimal policy through a neural network. At termination, a solution is found. Let V be the set of nodes, V^* be the set of remaining nodes, and is updated with state transition.

State Each state is represented as a binary node-state vector of length equal to the number of nodes in the sub-graph $s_t = [s_i : i \in K] \in \{0, 1\}^K$, where K is the set of nodes in the sub-graph. Here, the node $i \in K$ is considered excluded from the independent set if $s_i = 0$ and included if $s_i = 1$. The MDP starts with all node-states set to be excluded, meaning $s_i = 0$ for every $i \in K$, and it terminates when there are no node remaining.

Action The action represents the decision made by the agent regarding which node to select next from the available. The action at time step t can be defined as $a_t \in \{1, 2, \ldots |V|\}$.

Transition For two successive states s_t and s_{t+1}, along with the corresponding assignment a_t^*, the transition $P_{a_t^*}(s_t, s_{t+1})$ involves two deterministic steps: the update phase and the clean-up phase. The update phase incorporates the assignment a_t^* produced by the policy for the relevant vertices V^*, resulting in an intermediary node-state s_{t+1}, where $s_{a_t^*} = 1$ if $*$ is in V^*. The clean-up phase ensure once the node has been selected, its neighbors are removed from the remaining node set V^*, and thus the solutions are always *feasible*.

Reward We define the cardinality reward $r(s_t, s_{t+1})$ as the increase in the number of included vertices. This way, the total reward of the MDP reflects the *cardinality* of the independent set produced by our algorithm.

3.3. Reinforcement Learning-based GNNs

Our goal is to learn an optimal policy with graph policy networks, the elements of which are GCN and GAT layers. The goal of these models is leveraging both the dynamics of the MDP and the structural information from graph data. We call these models GCN+RL and GAT+RL. At the end of the policy networks, we use a linear projector followed by the *softmax* operation, to estimate a randomized policy $\pi(\theta)$ where θ is the learned parameters. Next, a_t is sampled from $\pi(\theta)$ and the immediate reward r_t and next state s_{t+1} is observed. The learning loop is continued until convergence.

GCN + RL This model uses a GCN for message passing, leveraging graph structure to encode features like node degree and centrality. The policy network operates within a MDP framework, where the state space captures the current graph configuration, and the action space represents node selection decisions. Rewards are designed to optimize the size of the independent set. The GCN layers aggregate neighborhood features to guide the RL agent toward selecting the next vertex in the independent set.

GAT + RL Similar to the GCN+RL, this approach uses a GAT to incorporate attention mechanisms for prioritizing key nodes during message passing. The GAT-based policy network enhances decision-making by dynamically assigning weights to edges based on their importance, allowing the model to focus on critical substructures within the graph.

4. Experiments

To validate our proposed approach, we conducted extensive experiments benchmarking a wide range of algorithms and models. These experiments aimed to assess solution quality, scalability, and runtime, with a particular focus on evaluating the effectiveness of our graph policy network solutions, which use both GCN+RL and GAT-RL. For each policy network, we used 2 GNN layers with hidden size 128. To maximize the reward, we used Adam optimizer with learning rate 0.001 and the reward discount factor 0.5.

4.1. Benchmark Methods

We ran multiple benchmarks to compare the performance of different approaches. These included:

Integer Programming (IP) An exact method to find the ground truth, integer programming served as the baseline against which all other methods were scored. As a mathematical optimization approach, integer programming guarantees optimality by exhaustively searching the solution space within the constraints of the problem. While computationally expensive and often impractical for large graphs, its accuracy makes it a reliable ground truth against which the other heuristic and learning-based methods can be evaluated. By comparing the optimality gap of other methods, to integer programming, we can objectively assess the accuracy of our proposed GNN+RL methods and other benchmarks.

Annealing A probabilistic optimization method that explores solutions iteratively, often used for NP-hard problems like MIS.

Physics-based GNN Leveraging principles from physics, this model applies graph-based reasoning to solve combinatorial optimization problems (Schuetz et al., 2021).

Greedy Algorithm A heuristic-based method known for its simplicity and speed, offering a quick but suboptimal solution.

Dynamic Programming GNN A self-supervised GNN that combines dynamic programming techniques with neural architectures (Brusca et al., 2023).

4.2. Dataset

We evaluated the methods using the Facebook Social Circles dataset from the Stanford Large Network Dataset Collection. The dataset consists of 4039 nodes representing users and 88234 undirected edges representing the friendships between users. This dataset provided a diverse and realistic set of graph structures, enabling us to test model performance across both sparse and dense configurations. The degree distribution (Figure 2) reveals a right-skewed and power-law characteristic.

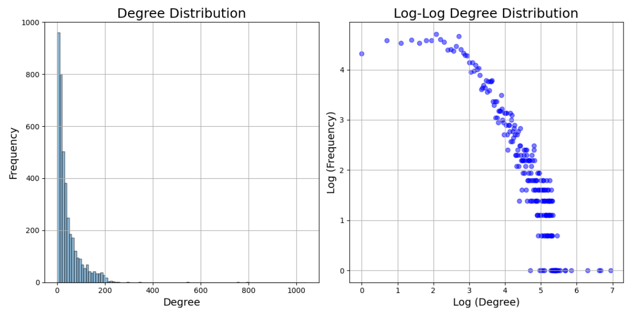

Figure 2. Degree distribution

After pre-processing, as discussed in Section 3, we created two smaller datasets: the dense graph dataset consists of 425-highest degree nodes and their one-hop neighbors. The sparse graph dataset contains 1000-lowest degree nodes and their one-hop neighbors. For each of these graphs, the original, dense, and sparse graphs, we split the graphs into smaller subgraphs of 500 nodes. The sub-graphs are used for model training and evaluation. Eventually, 8 original graphs, 3 dense graphs and 3 sparse graphs are created.

4.3. Evaluation Metrics

Performance was assessed using the following metrics:

Solution Quality: Measured as the size of the maximum independent set found, normalized against the integer programming baseline.

Generality: The ability to maintain performance across graphs of varying sizes and densities.

4.4. Experimental Design

To rigorously evaluate our proposed approach, we attempted to solve the MIS problem across multiple subgraphs derived from the original graph. As discussed in section 4.2, we partitioned into random subgraphs of around 500 nodes. The original graph was partitioned into eight random subgraphs. The dense graph was partitioned into three subgraphs. The sparse graph was partitioned into three subgraphs.

For each graph type (original, dense, and sparse), we applied all benchmarked methods to the respective subgraphs. The performance of each method was measured against the solution quality provided by the integer programming (IP) method, which served as the ground truth.

To ensure consistency and robustness in our evaluation, we computed the average performance of each method within each graph type by aggregating results across all subgraphs in that category. For each method, we calculated the mean error relative to the IP baseline, providing a normalized metric to compare solution quality across different graph structures.

This experimental setup enabled a comprehensive evaluation of the models' scalability and adaptability, highlighting their strengths and limitations in solving MIS across diverse graph configurations. By grouping subgraphs based on their original graph type, we ensured that the analysis reflected the structural characteristics of each graph while maintaining a fair comparison among the benchmarked methods.

4.5. Results

The optimality gap of each method on the three different graph types is presented in Table 1.

5. Analysis

The experimental results provide valuable insights into the performance of different methods for solving the MIS problem across various graph types and configurations. A detailed analysis is presented below.

5.1. Annealing Method and Integer Programming Baseline

Integer Programming (IP) method achieved the optimal solution across all graphs. As such, it is validated to serve as the ground truth. Among the heuristic and learning-based approaches, the **Annealing** method performed exceptionally well, matching the performance of the IP baseline. Its ability to explore a wide solution space while avoiding local optima explains its strong performance, particularly on sparse and combined graphs. The method demonstrated competitive results with minimal optimality gaps, making it a robust and efficient benchmark.

5.2. Performance of GNN-Based Methods and RL-Based Methods

Among the these methods, our proposed approaches—**Graph Convolutional Network (GCN) + Reinforcement Learning (RL)** and **Graph Attention Network (GAT) + RL**—consistently outperformed the GNN-based methods, such as the Physics-based GNN and Dynamic Programming GNN. Both GCN+RL and GAT+RL demonstrated key strengths in scalability, adaptability to various graph types, and the ability to generalize across different subgraph configurations.

GCN+RL This method uses convolutional layers to aggregate neighborhood features, enabling it to capture the local structure of the graph. While effective, its reliance on uniform aggregation across neighbors limited its performance in dense graphs, where node connectivity is highly variable. Nonetheless, GCN+RL demonstrated strong results with low optimality gaps, particularly in sparse graphs where its architecture was well-suited to handle lower connectivity.

GAT+RL This model outperformed GCN+RL across all graph types. By incorporating attention mechanisms, GAT+RL could prioritize critical nodes and edges during the message-passing process, resulting in better-informed decision-making within the reinforcement learning framework. This advantage was particularly pronounced in dense graphs, where the ability to weigh node importance provided significant performance gains. GAT+RL's superior handling of complex graph structures aligns with expectations, confirming the added value of attention mechanisms in graph optimization tasks.

5.3. Comparative Insights

The following observations summarize the comparative performance of the methods:

1. **Annealing vs. Learning-Based Models:**
    - The annealing method was highly competitive and achieved optimal solutions across graph types, matching the IP baseline.
    - Learning-based models (GCN+RL and GAT+RL) showed potential for scalability and adaptability, with GAT+RL reaching optimal performance in the sparse graph.
2. **GNN-Based Methods vs RL-based Methods:**
    - Among these methods, GCN+RL and GAT+RL demonstrated the best performance, far exceeding the Physics-based GNN and Dynamic Programming GNN by a large margin in terms of solution quality and optimality gap.
    - The performance gap between GCN+RL and GAT+RL highlights the importance of incorporating attention mechanisms for graph optimization tasks. GAT+RL's dynamic prioritization of nodes and edges allowed it to achieve superior results, particularly in sparse graphs.
3. **General Observations:**

Table 1. Optimality Gap Comparison Across Graph Types, Averaged Across Subgraphs

	Optimality Gap (%)		
Method	**Original Graph**	**Dense Graph**	**Sparse Graph**
Integer Programming (Baseline)	**0.00**	**0.00**	**0.00**
Annealing	**0.00**	**0.00**	**0.00**
Greedy	0.42	2.24	**0.00**
Physics-based GNN	3.42	70.99	2.46
Dynamic Programming GNN	66.65	52.91	70.47
GCN+RL	0.42	0.99	0.11
GAT+RL	0.32	0.98	**0.00**

- Both GCN+RL and GAT+RL demonstrated robust performance with low mean errors relative to the IP baseline. In all cases they reached $> 99\%$ accuracy relative to the IP baseline.
- The results affirm that reinforcement learning methods are the most effective among the learning-based and greedy approaches, balancing solution quality, scalability, and adaptability.

6. Conclusion

This paper propose an approach to find the maximum independent sets in graphs with graph policy networks, i.e. using GCN and GAT architectures within the RL algorithm, take in both structural and state information, to approximate an optimal policy of selecting nodes.

The results highlight the effectiveness of our proposed reinforcement learning-based approaches relative to other GNN-based models. While the Annealing method remains a strong benchmark for heuristic-based optimization, and the integer programming method is the ground truth, the GAT+RL model proved to be the most effective among the learning-based methods and the best among learning-based approaches. Offering a promising balance of scalability and solution quality, GAT+RL's use of attention mechanisms demonstrated the validity of a policy network in learning-based MIS solutions.

7. Limitations and Future Work

While our model performs well on reasonable, real-world data sets, implying relevance, it was unable to run on very large datasets (SNAP ego-Gplus; SNAP twitch-gamers; SNAP gemsec-Facebook) in a reasonable amount of time (<24hrs) on basic hardware (Apple M1). Additionally, within the state-of-the-art, there are few examples of models that have been tested and can run on such large data sets. This is due to both limitations in hardware and training data. ”Ground truth” algorithms and approaches, such as mixed-integer linear programming, are unable to scale to this size meaning it is difficult to discern whether or not results are accurate. Additionally, the need for additional system and computational memory is insurmountable at larger scales.

With that said, some models have been able to achieve a high performance on very large graphs, such as the model proposed by (Dai et al., 2018). While their architecture was shown to not be an effective learning mechanism (Böther et al., 2022), their use of classical algorithmic techniques in addition to the GCN allowed their approach to handle the large data sets given powerful hardware. Future work could explore more nuanced relationships between classical algorithmic techniques and machine learning approaches. Since traditional optimization methods still outperform the state-of-the-art at many scales, leveraging these methods could help close the performance gap while providing efficient solutions for large-scale applications. Improvements over the RL algorithm are also promising, for example, (Ahn et al., 2020) used Actor-Critic (A2C) algorithm, in which their MDP formulation is slightly different (allowing deference). Finally, improving methods for obtaining high-quality datasets with ground-truth labels could enhance model training and exploration. While this area is based around the models it hopes to train, creative employments of synthetic dataset generators might allow for such datasets to be created.

References

Ahn, S., Seo, Y., and Shin, J. Learning what to defer for maximum independent sets, 2020. URL https://arxiv.org/abs/2006.09607.

Angelini, M. C. and Ricci-Tersenghi, F. Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nature Machine Intelligence*, 5(1), 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00589-y. URL http://dx.doi.org/10.1038/s42256-022-00589-y.

Brusca, L., Quaedvlieg, L. C., Skoulakis, S., Chrysos, G., and Cevher, V. Maximum independent set: Self-training through dynamic programming. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 40637–40658. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7fe3170d88a8310ca86df2843f54236c-Paper-Conference.pdf.

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. What's wrong with deep learning in tree search for combinatorial optimization, 2022. URL https://arxiv.org/abs/2201.10494.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs, 2018. URL https://arxiv.org/abs/1704.01665.

Darvariu, V.-A., Hailes, S., and Musolesi, M. Graph reinforcement learning for combinatorial optimization: A survey and unifying perspective, 2024. URL https://arxiv.org/abs/2404.06492.

Dax, V. M., Li, J., Leahy, K., and Kochenderfer, M. J. Graph q-learning for combinatorial optimization, 2024. URL https://arxiv.org/abs/2401.05610.

Dong, G., Wang, J., Wang, M., and Su, T. An improved scheduling with advantage actor-critic for storm workloads, 2023. URL https://arxiv.org/abs/2312.04126.

Hu, S., Xiong, Z., Qu, M., Yuan, X., Côté, M.-A., Liu, Z., and Tang, J. Graph policy network for transferable active learning on graphs, 2020. URL https://arxiv.org/abs/2006.13463.

Jin, Y., Yan, X., Liu, S., and Wang, X. A unified framework for combinatorial optimization based on graph neural networks, 2024. URL https://arxiv.org/abs/2406.13125.

Kong, L., Feng, J., Liu, H., Tao, D., Chen, Y., and Zhang, M. Mag-gnn: Reinforcement learning boosted graph neural network, 2023. URL https://arxiv.org/abs/2310.19142.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018.

Luby, M. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pp. 1–10, 1985.

Schuetz, M. J. A., Brubaker, J. K., and Katzgraber, H. G. Combinatorial optimization with physics-inspired graph neural networks. *CoRR*, abs/2107.01188, 2021. URL https://arxiv.org/abs/2107.01188.

Tarjan, R. E. and Trojanowski, A. E. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

Vesselinova, N., Steinert, R., Perez-Ramirez, D. F., and Boman, M. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416, 2020. ISSN 2169-3536. doi: 10.1109/access.2020.3004964. URL http://dx.doi.org/10.1109/ACCESS.2020.3004964.

Wang, H. and Li, P. Unsupervised learning for combinatorial optimization needs meta-learning, 2023. URL https://arxiv.org/abs/2301.03116.

Xiao, M. and Nagamochi, H. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017.