Unsupervised Learning of Local Updates for Maximum Independent Set in Dynamic Graphs

Devendra Parkar*†‡, Anya Chaturvedi*‡, and Joshua J. Daymude*†
*School of Computing and Augmented Intelligence
†Biodesign Center for Biocomputing, Security and Society
Arizona State University, Tempe, AZ 85281
Emails: dparkar1@asu.edu, anya.chaturvedi@asu.edu, jdaymude@asu.edu

***Abstract*—We present the first unsupervised learning model for Maximum-Independent-Set (MaxIS) in dynamic graphs where edges change over time. Our method combines structural learning from graph neural networks (GNNs) with a learned distributed update mechanism that, given an edge addition or deletion event, modifies nodes' internal memories and infers their MaxIS membership in a single, parallel step. We evaluate our model against a mixed integer programming solver and a breadth of unsupervised and supervised learning models for combinatorial optimization on static graphs. Across dynamic graphs of 200–1,000 nodes, our model achieves approximation ratios that are competitive with the state-of-the-art models while running 1.91–6.70x faster. When generalizing to graphs with 100x more nodes than those used for training, our model produces MaxIS solutions 1.00–1.18x larger than all other unsupervised models, but is outperformed by the state-of-the-art supervised model. These results demonstrate that this novel, unsupervised, update-based learning approach to dynamic combinatorial optimization is a viable alternative to the naïve reapplication of analogous models for static graphs, leveraging temporal information to improve neural methods for combinatorial optimization.**

I. Introduction

Combinatorial optimization problems on graphs (e.g., Graph-Coloring, Traveling-Salesman, Vehicle-Routing, etc.) arise naturally in many practical domains [4, 40]. However, many of these problems are classically intractable to solve exactly, and some even resist efficient approximation. The situation is exacerbated by the fact that many real-world applications are *dynamic*, requiring algorithms to not only find solutions but also maintain them as the underlying structure evolves. While one could iteratively apply algorithms designed for static graphs at each time step, this approach is often computationally expensive and inefficient. Critically, this approach ignores the often close relationship between successive graph structures. A more effective strategy might avoid repeatedly recomputing solutions from scratch by instead leveraging these small structural changes to incrementally update an existing solution. Such *update algorithms* are an active area of theoretical algorithms research [5, 7, 11, 17, 21, 51], but no learning algorithms have yet taken this approach.

In this paper, we consider the Maximum-Independent-Set (MaxIS) problem in a dynamic setting. For static graphs, MaxIS is classically known to be NP-hard [18] and hard to approximate: for general graphs, deterministic approximation within a constant factor is impossible [35] and within an $n^{1-\varepsilon}$ factor is NP-hard [22]. Nevertheless, MaxIS has a diverse range of real-world applications, from identifying functional nodes in brain networks [1] to constructing diversified investment portfolios [23]. For example, different stocks can be modeled as nodes and those that are closely related have an edge between them to form a conflict graph; thus, the problem of building a diverse portfolio reduces to finding a high-value independent set. In many of these situations, the conflict graph changes dynamically over time due to factors such as neuronal development, stock market fluctuations, and other unpredictable events. Our goal is to solve MaxIS on such dynamically evolving graphs.

Recently, the learning community has developed several methods for solving combinatorial optimization problems on static graphs [8]. Central to many of these methods are Graph Neural Networks (GNNs) due to their efficiency in capturing the structural information present in real-world networks such as the Internet, social networks, and molecular interactions [30, 44, 50]. In the dynamic setting, recent models such as temporal GNNs are primarily concerned with tasks such as edge prediction, node classification, graph classification, etc. [36, 46]. Work considering combinatorial optimization problems in a dynamic setting remains limited, with few examples such as Gunarathna et al. [19], which is an indirect heuristic learning approach. No existing models learn update mechanisms comparable to traditional update algorithms.

We propose a model that solves MaxIS in dynamic graphs by directly learning an update mechanism instead of a heuristic. We utilize the power of message-passing GNNs [6] combined with sequence learning to learn an update mechanism that maintains an approximate MaxIS solution as the underlying graph changes. GNNs learn to aggregate structural information and update nodes' memberships in the MaxIS solution at each time step, while sequence learning extrapolates the effects of edge dynamics to changes in the MaxIS solution. Having learned an update mechanism, the model is able to update a MaxIS solution following a change in the dynamic graph in a single inference step, unlike more popular heuristic learning methods which require multiple inference steps and thus scale poorly with graph size. We underscore this as a necessary shift in perspective required to address some of the common criticisms of learning approaches to solving combinatorial

‡These authors contributed equally to this work.

optimization problems, namely, lack of scalability, computation inefficiency, and huge training sample requirements.

Contributions: We present the first unsupervised learning model for MAXIMUM-INDEPENDENT-SET (MaxIS) in dynamic graphs. The central idea of learning local, distributed update mechanisms for dynamic combinatorial optimization problems is novel, and we demonstrate its efficacy in our evaluations. Compared to the state-of-the-art models for combinatorial optimization in static graphs (OptGNN [47], unsupervised; and Fast T2T [28], supervised), our model maintains competitive approximation ratios across a variety of dynamic graph topologies. This ensemble of methods all match or outperform a commercial mixed integer programming solver [20] in a small fraction of its runtime, surpassing the abilities of earlier unsupervised models for combinatorial optimization [2, 9, 24]. One specific parameterization of our model, which strictly bounds message-passing and update learning in local neighborhoods around edge dynamics, is able to maintain this strong performance while running 1.91–6.70x faster than OptGNN or Fast T2T, showcasing the scalability of our approach. In generalization experiments considering graphs 100x the size of those used for training, our model produces MaxIS solutions 1.00–1.18x larger than the other unsupervised models; however, both OptGNN and Fast T2T achieve better performance–runtime tradeoffs.

II. RELATED WORK

Traditional Algorithms for MaxIS: MAXIMUM-INDEPENDENT-SET (MaxIS) is a classical NP-complete problem in graph theory and has been explored since the 1970s in the field of combinatorial optimization. Algorithms seeking exact solutions use reduction rules [39], branching heuristics [16], and kernelization techniques [27] to reduce the size of the search space. Among exact algorithms, the state-of-the-art is Xiao and Nagamochi [45] which runs in $1.1996^n n^{O(1)}$ time and polynomial space, where n is the number of nodes in the graph. Extensions of MaxIS to dynamic graphs remain relatively unexplored. Zheng et al. [51] and Gao et al. [17] have developed update algorithms using rule-based optimization techniques for incrementally maintaining an existing solution in response to edge insertions or deletions. However, no such approaches have been taken by the learning community, which we discuss next.

Learning in Dynamic Graphs: Memory-based models utilize various forms of persistent, high-dimensional representations (i.e., memories) for nodes to learn the dynamics of a graph as it changes. These have been very successful in efficiently learning prediction and classification tasks in dynamic graphs [26, 36, 41, 48]. Some models also include a GNN-based graph embedding module to mitigate the problem of memory staleness, i.e., when a node's memory hasn't been updated for a long time. We utilize these innovations in our model. Other learning models for dynamic graphs do not involve memory [31, 33, 46] but have never been applied to combinatorial optimization problems.

Learning for Combinatorial Optimization: A vast body of machine learning research considers combinatorial optimization problems [8, 14, 34]. Among these, GNNs are the most common backbone given their excellent ability to characterize the structural information of graphs [44, 50]. For MaxIS specifically, autoregressive approaches (i.e., those that take multiple inference steps to output a solution) such as reinforcement learning (RL) and dynamic programming (DP) have emerged as natural choices due to their easy constraint enforcement and a priori integral solution output [2, 9, 49]. Yet, these methods suffer from sample inefficiency, requiring a number of inference steps that scales linearly with problem size [24, 34]. Non-autoregressive approaches (i.e., "single-shot" methods) have also been proposed, including some supervised, diffusion-based models [28, 38] and a broader body of work on unsupervised, GNN-based methods [24, 29, 37, 43, 47]. However, these only apply to static graphs. To our knowledge, there is only one other learning model that explicitly considers dynamic graphs [19], and it applies RL to dynamic versions of the TRAVELING-SALESMAN and VEHICLE-ROUTING problems. Ours is the first non-autoregressive, unsupervised learning method for a combinatorial optimization problem in dynamic graphs.

III. MAXIMUM INDEPENDENT SETS IN DYNAMIC GRAPHS

Consider an undirected, static graph $G = (V, E)$ with nodes V and edges E. An *independent set* $I \subseteq V$ of G is a set of nodes no two of which are adjacent, i.e., for all $u, v \in I$, we have $(u, v) \notin E$. Its *size* is its number of nodes, denoted $|I|$. A *maximum independent set* (MaxIS) of G is any independent set I of G satisfying $|I| = \max\{|I'| : I' \text{ is an independent set of } G\}$.

Dynamic graphs (also called dynamic networks, temporal graphs, or evolving graphs) are graphs whose nodes and/or edges change over time [10, 25, 36]. For this work, we consider dynamic graphs where at most one edge changes per time and the underlying set of nodes is fixed. Formally, a dynamic graph $\mathcal{G}$ is a finite sequence of *graph snapshots* $(G_0, G_1, \ldots, G_T)$ where each snapshot $G_t = (V, E_t)$ is an undirected, static graph on nodes V and, for all time steps $0 < t \leq T$, we have $|E_{t-1} \oplus E_t| = 1$, i.e., there is exactly one edge addition or one edge deletion per time step. We denote the *edge event* at time t which transitions G_{t-1} to G_t as $\mathcal{E}_t$. The *neighborhood* of a node v at time t is $\mathcal{N}_t(v) = \{u \in V \mid (u, v) \in E_t\}$ and its *degree* is denoted $d_t(v) = |\mathcal{N}_t(v)|$. The x-*neighborhood* of an edge event $\mathcal{E}_t$ are all nodes u such that the shortest path in G_t from u to an endpoint of the edge being added or deleted in $\mathcal{E}_t$ contains at most x edges. The diameter of graph snapshot G_t, denoted $\text{diam}(G_t)$, is the maximum shortest hop-distance between any two nodes in G_t.

We define the DYNAMIC-MAXIMUM-INDEPENDENT-SET (DynMaxIS) problem as follows: Given a dynamic graph $\mathcal{G} = (G_0, G_1, \ldots, G_T)$—or, equivalently, an initial graph snapshot G_0 and a sequence of edge events $(\mathcal{E}_1, \ldots, \mathcal{E}_T)$—obtain a MaxIS for each snapshot G_t. This problem is clearly NP-hard since it contains the NP-complete MaxIS problem as a special case ($T = 0$), so we are realistically interested in obtaining a large *approximate* MaxIS for each snapshot.

IV. Unsupervised Learning Model for DynMaxIS

As discussed in Section II, memory-based models have been quite successful in recent years. Motivated by their effectiveness in predictive learning of dynamics, we present an unsupervised learning model focused on the MaxIS problem in dynamic graphs. Specifically, we repurpose the two key modules of Temporal Graph Networks (TGN) [36], memory and graph embedding, to learn an update mechanism that maintains a candidate MaxIS solution as the underlying graph changes. Moreover, inspired by the use of messages (seen as random perturbations) to improve GNN performance for NP-hard problems [32], we couple node memory to an event handling module to inform nodes about nearby edge events.

Our model responds to edge events (i.e., edge additions or deletions) over time (see Fig. 1). At each time step, the *event handling module* is responsible for signaling all nodes within the edge event's α-neighborhood that the event occurred. These nodes then use the *memory module* to update their internal representations using this signal. Finally, nodes within the edge event's β-neighborhood use the *local aggregation (graph embedding) module* to update their probabilities of membership in the MaxIS solution (i.e., an *estimate*) based on their memory, degree, and memories of their neighbors.

Event Handling Module: This module provides each node v in the α-neighborhood of the edge event $\mathcal{E}_t$ with a signal $s_t(v)$ encoding the type of event (addition or deletion) and the node's normalized distance from the event. Formally, the signal for node v at time t is

$$s_t(v) = [\texttt{enc}(\mathcal{E}_t) \mid\mid r_t(v)], \tag{1}$$

where $\texttt{enc}$ encodes edge deletions as $[0, 1]$ and edge additions as $[1, 0]$, and $r_t(v)$ is the hop-distance from v to $\mathcal{E}_t$ normalized by α and linearly interpolated into $[-1, 1]$.

Memory Module: A node's memory is a high-dimensional representation maintained throughout the model's execution that captures relevant structural changes affecting the node. At each time t, this module updates the memories of nodes in the α-neighborhood of the edge event $\mathcal{E}_t$. Updates are performed using a GRU cell [12]. The memory of a node v at time t is

$$m_t(v) = \begin{cases} \texttt{GRU}\big([m_{t-1}(v) \mid\mid s_t(v)]\big) & \text{if } v \text{ in } \alpha\text{-nbrhd.;} \\ m_{t-1}(v) & \text{otherwise.} \end{cases} \tag{2}$$

Local Aggregation Module: This module updates the estimates—i.e., probabilities of MaxIS membership—of all nodes in the β-neighborhood of the edge event $\mathcal{E}_t$. First, each node v aggregates its immediate neighbors' memories:

$$\tilde{h}_t(v) = \texttt{ReLU}\left(\sum_{u \in \mathcal{N}_t(v)} \mathbf{W}_1 m_t(u)\right), \tag{3}$$

where $\mathbf{W}_1$ are learnable weights. Next, node v combines these aggregated memories with its own memory and degree information to obtain a local embedding

$$h_t(v) = \mathbf{W}_2 \left[\tilde{h}_t(v) \mid\mid m_t(v) \mid\mid d_t(v)\right], \tag{4}$$

where again $\mathbf{W}_2$ are learnable weights. Finally, node v passes its embedding through a step-down MLP and a sigmoid function σ to obtain its estimate in $[0, 1]$:

$$p_t(v) = \sigma\big(\texttt{MLP}(h_t(v))\big). \tag{5}$$

Any node v beyond the edge event's β-neighborhood simply retains its previous estimate, i.e., $p_t(v) = p_{t-1}(v)$.

Update Training: We first define a loss function from a node's local perspective and then define a cumulative loss function from these local losses. This loss function is similar to those of other single-shot unsupervised methods, but enabling its calculation by individual nodes allows the memory and local aggregation modules to synergize, learning a distributed update mechanism. The loss for a node v at time t is

$$\ell_t(v) = -p_t(v) + \frac{c}{2d_t(v)} \sum_{u \in \mathcal{N}_t(v)} p_t(u) p_t(v). \tag{6}$$

The $-p_t(v)$ term rewards nodes with large estimates (i.e., probabilities of MaxIS membership closer to 1), aligning with the goal of finding a large independent set. This is counterbalanced by the $\sum_{u \in \mathcal{N}_t(v)} p_t(u) p_t(v)$ term which penalizes violations of independence, i.e., when neighbors u and v both have large estimates. This constraint violation term is normalized by the degree of v and is then halved to compensate for double-counting the loss from the perspectives of both u and v. Finally, the constant c is a hyperparameter balancing the two terms' loss contributions; in practice, $c = 3$ worked reasonably well across our evaluations (Section V).

The cumulative loss at time t is the sum of all local losses $\ell_t(v)$ for nodes v in the edge event's β-neighborhood (i.e., all nodes that updated their estimates at time t). This set of nodes at this one time step forms a single batch in our training process. By setting β as an appropriate function of dynamic graph size (see Model Variants, below), batch sizes will automatically scale to any input dynamic graph and provide sufficient inference samples for a training step. The sequential processing of all edge events in a dynamic graph's training set forms a single epoch in our training process. In a single run, we train until the losses for epochs have stabilized and select the epoch with the least loss.

Model Variants: The neighborhood sizes α and β control which nodes utilize an edge event to update their memories and/or update their estimates and contribute to cumulative loss, respectively. In the *bounded cascading* (BCAS) model variant, we set $\alpha = \beta = \gamma \cdot \text{diam}(G_0)$, where $\gamma \ll 1$, strictly bounding nodes' memory and estimate updates to a local region around each edge event. This variant mimics non-learning-based update algorithms for MaxIS which analogously spread out from the point of the edge event, updating nodes' memberships to generate a new MaxIS [7, 11]. Locally bounding these update regions typically yields very fast update operations, at the potential risk of missing better MaxIS solutions that require far-reaching updates.

In the *non-cascading* (No-CAS) model variant, $\alpha = 0$ and $\beta = \text{diam}(G_0)$; i.e., only the two nodes directly involved in the edge event update their memories, but all nodes in the graph

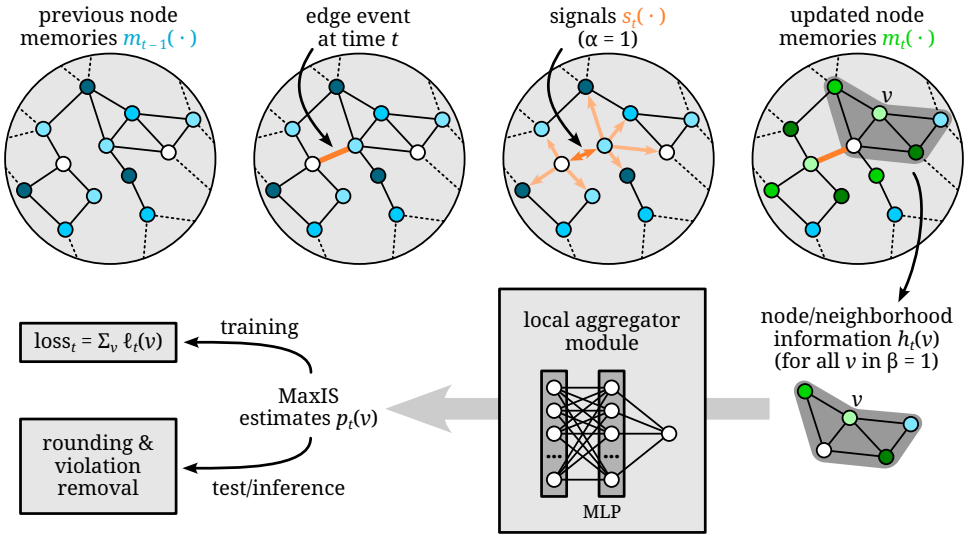

Fig. 1: An overview of our unsupervised learning model for MaxIS on dynamic graphs. Nodes maintain memories (node colors) which are updated by signals propagating in the α-neighborhoods of an edge event (orange arrows). Any node in the β-neighborhood of the edge event then locally aggregate its updated memory and those of its neighbors to estimate its new probability of MaxIS membership. These estimates are then used to compute loss (during training) or to generate an integral MaxIS solution (during testing and inference).

update their estimates and contribute to cumulative loss. In this variant, the local aggregation module is primarily responsible for the graph-wide learning of the update mechanism, with node memory aiding in the locality of the edge event. This variant is important for dense graph topologies and edge dynamics where frequent update cascades may disrupt relevant information in nodes' memories.

Initial MaxIS Generation and Unsupervised Learning: Update algorithms for MaxIS crucially require an existing MaxIS solution for the initial snapshot G_0 that is updated as the graph structure changes. We mitigate this (potentially very expensive) prerequisite by utilizing an initial MaxIS generation phase that efficiently produces memories for all nodes in G_0 and model weights that training can warm-start from. The idea is to build up G_0 one edge at a time (as a sequence of edge addition events), updating only the memories of the nodes involved in each edge addition. Once G_0 is fully constructed, estimates and corresponding losses are computed for all nodes. This constitutes a single epoch of this generation phase; in a single run of the generation phase, we execute epochs until the cumulative loss has stabilized. After executing multiple runs with random seeds, we extract the node memories and model weights from the run with the least cumulative loss to warm-start model training. Thus, the model as a whole learns to first find a candidate MaxIS (generation phase) and then maintain it (training phase) in an entirely unsupervised manner.

Integral Solution Generation: During testing and deployment, we use a simple rounding and violation removal procedure to convert the model's relaxed node estimates into an approximate MaxIS. For a given snapshot G_t, all nodes v with $p_t(v) \geq 0.5$ are initially included in the candidate solution.[1] Then, the following two steps are repeated until independence is achieved: (1) compute the number of *violations* for each node in the candidate solution, i.e., the number of neighboring nodes that are also in the candidate solution; (2) remove any node with the most violations. Such a procedure is a necessary part of single-shot learning methods that directly use relaxation to solve optimization problems with hard constraints [15, 24, 37, 43].

V. Experiments

A. Experimental Methodology

Datasets: We evaluate model performance using various initial graph structures with synthetic dynamics. For initial graph structures, we consider RB-200 [42], which has been treated as the challenging evaluation instance for many recent combinatorial optimization learning methods [9, 28, 47]; BRAIN, a biological network with a sparse small-world topology [13]; and randomly generated Erdős–Rényi (ER) and Power Law (PL) graphs with 1,000 nodes each. Across all graphs, dynamics are produced by sampling edge additions or deletions from the initial snapshot's degree distribution[2] for a desired number of time steps; here, we use 50,000 time steps for RB-200 and 100,000 for the others.[3]

Our Model and Comparisons: We ran our complete training pipeline using the two model variants: BCAS and No-CAS. For

[1]Almost all estimates $p_t(v)$ are nearly-zero or nearly-one, so the threshold for inclusion could vary anywhere in $[0.25, 0.75]$ without changing the results.

[2]A naïve alternative is to sample edges uniformly at random, but this converges to homogeneous random snapshots over n nodes after n^2 rounds, in expectation. This makes studying different datasets uninteresting.

[3]These datasets' snapshots are divided chronologically by time step into 50:25:25 training, validation, and testing splits (70:15:15 for RB-200).

the BCAS variant, we set $\alpha = \beta = 0.25 \cdot \text{diam}(G_0)$, i.e., half of the radius of the first snapshot of the dynamic graph. For each variant, we performed three training runs with different random seeds and report results for the median run. Since ours is the only learning method for MaxIS in the dynamic setting, we compare our method against a commercial mixed integer programming solver and existing learning methods—both supervised and unsupervised—for MaxIS on static graphs applied to each snapshot of a dynamic graph independently.

- Gurobi [20] is the most commonly used and commercially available mixed integer programming solver. We use Gurobi's default settings for mixed integer program solving, which according to their documentation, is a branch-and-bound search.
- OptGNN [47] is the state-of-the-art GNN-based, single-shot, unsupervised model for combinatorial optimization in static graphs. It leverages a novel semidefinite programming formulation to efficiently approximate various problems, which we extended to address MaxIS.
- Erdős-GNN [24] is an earlier GNN-based, single-shot, unsupervised model for combinatorial optimization in static graphs. This also required new extensions for MaxIS.
- DP-GNN [9] combines dynamic programming with GNNs to form a multi-step, autoregressive comparator model.
- LwDMIS [2] is a deep reinforcement learning method that adaptively adjusts its stages and defers decisions about nodes' MaxIS membership to improve scalability.
- Fast T2T [28] is the state-of-the-art supervised diffusion model for combinatorial optimization in static graphs.

All methods were trained (if applicable) and tested on a machine with an AMD EPYC 7413 CPU and a 20 GB slice of one NVIDIA A100 GPU. For the learning methods, training was halted after seven days. For Gurobi, we enforced a time limit of 1 s per snapshot; this produces exact solutions on PL-1000, but yields approximations for the other datasets.

Metrics: We evaluate each method using three metrics: performance, runtime, and peak memory usage. Performance refers to the mean ratio of a method's candidate MaxIS size against that of Gurobi's across all snapshots in a dynamic graph.[4] Runtime refers to the mean time taken by a method to process a single snapshot and generate a candidate MaxIS across all snapshots in a dynamic graph (reported as seconds per graph, s/g). Peak memory usage denotes the maximum memory used by a method during training.

B. Results

Fig. 2 shows that our models match the performance of the state-of-the-art methods—both unsupervised (OptGNN) and supervised (Fast T2T)—with competitive or superior runtimes, depending on the dataset. Specifically, across all datasets, one or both of our BCAS and No-CAS variants achieve performance that is statistically indistinguishable from those of OptGNN and Fast T2T. When directly comparing mean performance ratios, BCAS's are 0.88–1.04x of OptGNN's and 0.90–1.02x of Fast T2T's; analogously, No-CAS's are 0.96–1.00x of OptGNN's and 0.91–1.00x of Fast T2T's. Taking a broader view, all four of these methods perform as well or better than Gurobi in a fraction of the time; on BRAIN in particular, this ensemble achieves a remarkable 1.34–1.41x improvement over the mixed integer programming solver. The same cannot be said of the earlier unsupervised methods (Erdős-GNN, DP-GNN, and LwDMIS), which often do not even achieve parity with Gurobi.

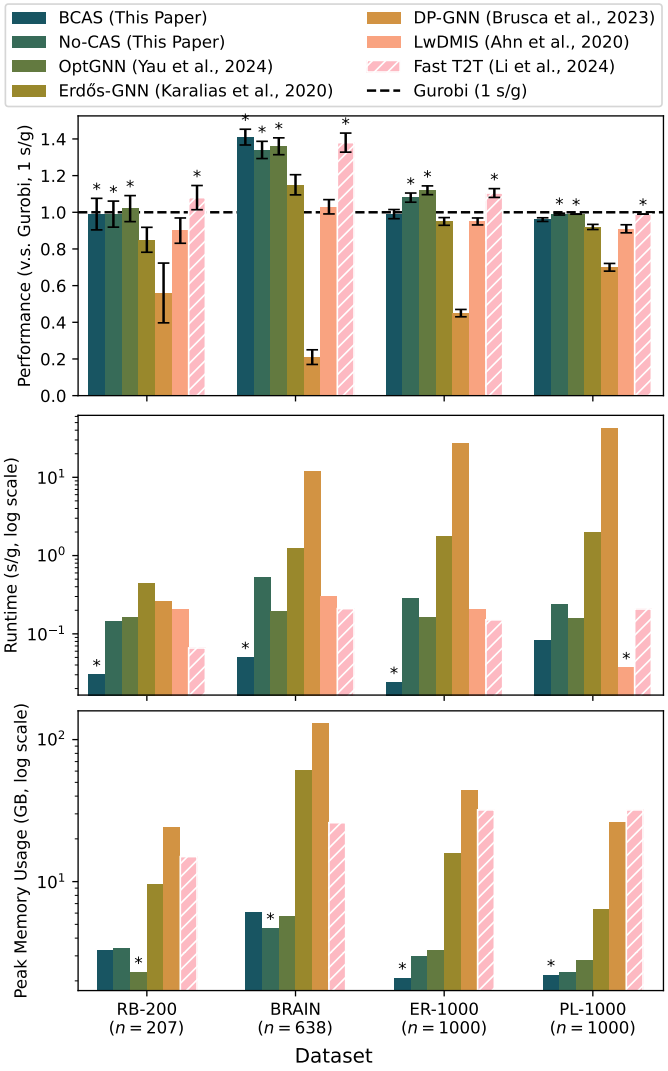

Fig. 2: Models' performance (top), runtime (middle), and peak memory usage (bottom) across four datasets. The best method for each metric and dataset is marked with a (∗); for performance, all methods whose performance ranges (error bars) are statistically indistinguishable from the method with the highest mean performance are considered "best".[5]

[4]Fast T2T does not specify a procedure for removing violations of independence, so we report the size of its candidate MaxIS as the number of nodes in its solution minus the number of pairwise violations.

[5]DP-GNN is very slow on the larger graphs (BRAIN, ER-1000, and PL-1000), so its results only reflect evaluation on these graphs' first 5,000 (20%) testing time steps. Also, LwDMIS failed to train on some graph snapshots as its search space collapsed to null; thus, we only report peak memory usage for methods that completed the entire training process.

TABLE I: Generalization results reported as mean ± std.dev. performance ratios vs. 10 s Gurobi runs followed by mean runtime per graph snapshot in parentheses. Each model was trained on dynamic graphs with 100 nodes and 35,000 time steps and tested on those with 10,000 nodes and 5,000 time steps. Notably, DP-GNN is prohibitively slow on these large graphs (taking nearly a week just to evaluate the first six snapshots of each dataset), so we omit its performance results.

Dataset (→) Method (↓)	ER-10k ($n = 10{,}000$)	PL-10k ($n = 10{,}000$)
BCAS	0.63 ± 0.085 *(0.015 s/g)*	**0.99 ± 0.002** (0.679 s/g)
No-CAS	**1.17 ± 0.024** (5.349 s/g)	**0.99 ± 0.001** (3.372 s/g)
OptGNN	1.12 ± 0.012 (0.353 s/g)	**0.99 ± 0.001** *(0.194 s/g)*
Erdős-GNN	1.07 ± 0.023 (19.202 s/g)	0.94 ± 0.009 (18.415 s/g)
DP-GNN	N/A (36,239.508 s/g)	N/A (39,994.408 s/g)
LwDMIS	0.99 ± 0.020 (0.541 s/g)	0.92 ± 0.003 (0.296 s/g)
Fast T2T	**1.27 ± 0.006** (1.29 s/g)	**0.99 ± 0.000** (1.84 s/g)

Our BCAS variant consistently achieves superior runtimes over OptGNN and Fast T2T (1.91–6.70x), offering a fast alternative to the state-of-the-art without sacrificing performance. Peak memory usage comparisons against OptGNN are more variable—sometimes better and sometimes worse across datasets—but our models are certainly more memory efficient than the supervised Fast T2T model (0.07–0.23x). Taken together, these performance and resource comparisons demonstrate the strength of our unsupervised update learning method as a viable alternative to repeatedly applying learning models for static graphs to individual graph snapshots.

Generalization: We evaluated each model's generalization ability using dynamic graphs with 100x more nodes than those used for training (Table I). Our No-CAS variant significantly outperforms the earlier unsupervised methods (1.05–1.18x better than Erdős-GNN, LwDMIS, and DP-GNN)—as does BCAS on the PL-10k dataset specifically—though neither are as fast as LwDMIS. OptGNN is better still, achieving similar generalized performance to No-CAS while running 1.53x faster than LwDMIS. The supervised method, Fast T2T, has the best generalization performance but middling runtimes (slower than BCAS, LwDMIS, and OptGNN, but faster than the rest). On the negative side, it is somewhat surprising how poorly BCAS generalizes on the ER-10k dataset given its strong performance on ER-1000 (Fig. 2). This may be caused by a mismatch of training vs. testing hyperparameters: in BCAS, both α and β are fractions of the initial graph snapshot's diameter, which generalization inflates when considering 100x more nodes.

VI. Conclusion

We present a single-shot, unsupervised learning model for the Maximum-Independent-Set (MaxIS) problem in dynamic graphs, where the topology evolves over time. Using a simple GNN backbone, our model achieves results competitive with both supervised (Fast T2T [28]) and unsupervised (Opt-GNN [47]) state-of-the-art methods, breaking fertile ground for exploration with advanced architectures, representations, and objective design. Unlike heuristic learning approaches, our method learns an update mechanism analogous to those used in traditional update algorithms for dynamic combinatorial optimization. Our experiments across various dynamic graph topologies demonstrate our model's competitive performance against the state-of-the-art, surpassing earlier unsupervised methods like DP-GNN [9], Erdős-GNN [24] and LwDMIS [2]. In future work, we plan to extend our approach into a general framework for learning update algorithms applicable to a broader class of combinatorial optimization problems, leveraging recent advances in neural models.

Limitations: Although a thorough comparison to traditional, non-learning-based algorithms is beyond the scope of this paper, recent advances in update algorithms for MaxIS [17, 51] and past learning vs. non-learning comparisons in the static setting [3] invite some reflection on this point. We evaluated a simple greedy algorithm for MaxIS on our four dynamic graph topologies and found that it also achieves performance statistically indistinguishable from state-of-the-art, matching the claims we make about our own model. For RB-200 in particular, the greedy algorithm runs 6.00x faster than the fastest learning model for this dataset (BCAS). Thus, even with these preliminary investigations, there is an evident gap to close between learning-based approaches to combinatorial optimization and their traditional counterparts.

There are various aspects of our approach that should be investigated further. The BCAS and No-CAS variants exemplify two specific instantiations of our model in terms of α and β hyperparameters, but a broader characterization of model behavior as a function of these propagation radii may reveal further performance and runtime improvements. Also, the dynamics for all our datasets sample only one edge event per time, albeit in a degree distribution-preserving manner. It would be interesting to evaluate our model on large dynamic graphs whose structure and dynamics are both empirical. Finally, our model makes limited use of temporal dynamics when learning its update mechanism. Leveraging this rich source of information using recent advances in neural architectures and attention mechanisms is an important direction for future work.

Acknowledgments & Code Availability

This work is supported in part by NSF award CCF-2312537. Source code for our method and complete reproducibility instructions for dataset processing and experiments is available at https://github.com/DaymudeLab/LearnDynamicMaxIS.

References

[1] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A Biological Solution to a Fundamental Distributed Computing Problem. *Science*, 331(6014):183–185, 2011. doi:10.1126/science.1193210.

[2] S. Ahn, Y. Seo, and J. Shin. Learning What to Defer for Maximum Independent Sets. In *Proceedings of the 37th International Conference on Machine Learning*, pages 134–144, 2020.

[3] M. C. Angelini and F. Ricci-Tersenghi. Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nature Machine Intelligence*, 5(1):29–31, 2022. doi:10.1038/s42256-022-00589-y.

[4] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*, volume 17 of *Princeton Series in Applied Mathematics*. Princeton University Press, Princeton, NJ, USA, 2007.

[5] S. Assadi, K. Onak, B. Schieber, and S. Solomon. Fully Dynamic Maximal Independent Set with Sublinear in n Update Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1919–1936, 2019. doi:10.1137/1.9781611975482.116.

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, and V. Zambaldi *et al.* Relational inductive biases, deep learning, and graph networks, 2018. doi:10.48550/arxiv.1806.01261.

[7] S. Behnezhad, M. Derakhshan, M. Hajiaghayi, C. Stein, and M. Sudan. Fully Dynamic Maximal Independent Set with Polylogarithmic Update Time. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 382–405, 2019. doi:10.1109/FOCS.2019.00032.

[8] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. doi:10.1016/j.ejor.2020.07.063.

[9] L. Brusca, L. C. Quaedvlieg, S. Skoulakis, G. Chrysos, and V. Cevher. Maximum Independent Set: Self-Training through Dynamic Programming. In *Advances in Neural Information Processing Systems*, volume 36, pages 40637–40658, 2023.

[10] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-Varying Graphs and Dynamic Networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

[11] S. Chechik and T. Zhang. Fully Dynamic Maximal Independent Set in Expected Poly-Log Update Time. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 370–381, 2019. doi:10.1109/FOCS.2019.00031.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 2014. doi:10.48550/arxiv.1412.3555.

[13] N. A. Crossley, A. Mechelli, P. E. Vértes, T. T. Winton-Brown, A. X. Patel, C. E. Ginestet, P. McGuire, and E. T. Bullmore. Cognitive relevance of the community structure of the human brain functional coactivation network. *Proceedings of the National Academy of Sciences*, 110 (28):11583–11588, 2013. doi:10.1073/pnas.1220826110.

[14] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems*, volume 30, pages 1–11, 2017.

[15] P. Donti, D. Rolnick, and Z. Kolter. DC3: A learning method for optimization with hard constraints. In *Proceedings of the 9th International Conference on Learning Representations*, pages 1–17, 2021.

[16] F. V. Fomin and P. Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.

[17] X. Gao, J. Li, and D. Miao. Dynamic Approximate Maximum Independent Set on Massive Graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1835–1847, 2022. doi:10.1109/ICDE53745.2022.00183.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., United States, 1979.

[19] U. Gunarathna, R. Borovica-Gajic, S. Karunasekera, and E. Tanin. Dynamic graph combinatorial optimization with multi-attention deep reinforcement learning. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, pages 1–12, 2022. doi:10.1145/3557915.3560956.

[20] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[21] K. Hanauer, M. Henzinger, and C. Schulz. Recent Advances in Fully Dynamic Graph Algorithms – A Quick Reference Guide. *ACM Journal of Experimental Algorithmics*, 27:1–45, 2022. doi:10.1145/3555806.

[22] J. Håstad. Clique is hard to approximate within n^(1-ε). In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636, 1996. doi:10.1109/SFCS.1996.548522.

[23] R. Hidaka, Y. Hamakawa, J. Nakayama, and K. Tatsumura. Correlation-Diversified Portfolio Construction by Finding Maximum Independent Set in Large-Scale Market Graph. *IEEE Access*, 11:142979–142991, 2023. doi:10.1109/ACCESS.2023.3341422.

[24] N. Karalias and A. Loukas. Erdős Goes Neural: An Unsupervised Learning Framework for Combinatorial Optimization on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, pages 40637–40658, 2020.

[25] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[26] S. Kumar, X. Zhang, and J. Leskovec. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019. doi:10.1145/3292500.3330895.

[27] S. Lamm, P. Sanders, C. Schulz, D. Strash, and R. F. Werneck. Finding Near-Optimal Independent Sets at

Scale. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 138–150, 2016. doi:10.1137/1.9781611974317.12.

[28] Y. Li, J. Guo, R. Wang, H. Zha, and J. Yan. Fast T2T: Optimization Consistency Speeds Up Diffusion-Based Training-to-Testing Solving for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 30179–30206, 2024. doi:10.52202/079017-0950.

[29] Z. Li, Q. Chen, and V. Koltun. Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search. In *Advances in Neural Information Processing Systems*, volume 31, pages 1–10, 2018.

[30] H. Liu, Y. Zhang, T. Wang, R. Jiang, and Y. Yu. Weighted Graph Clustering via Scale Contraction and Graph Structure Learning. In *Proceedings of the ACM Web Conference 2026*, pages 1004–1014, 2026. doi:10.1145/3774904.3792363.

[31] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini. Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities. *Transactions on Machine Learning Research*, pages 1–24, 2023.

[32] P. A. Papp, K. Martinkus, L. Faber, and R. Wattenhofer. DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 21997–22009, 2021.

[33] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5363–5370, 2020. doi:10.1609/aaai.v34i04.5984.

[34] Y. Peng, B. Choi, and J. Xu. Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art. *Data Science and Engineering*, 6(2):119–141, 2021. doi:10.1007/s41019-021-00155-3.

[35] J. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986. doi:10.1016/0196-6774(86)90032-5.

[36] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs, 2020. doi:10.48550/arxiv.2006.10637.

[37] M. J. A. Schuetz, J. K. Brubaker, and H. G. Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022. doi:10.1038/s42256-022-00468-6.

[38] Z. Sun and Y. Yang. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 3706–3731, 2023.

[39] R. E. Tarjan and A. E. Trojanowski. Finding a Maximum Independent Set. *SIAM Journal on Computing*, 6(3): 537–546, 1977. doi:10.1137/0206038.

[40] P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2014. doi:10.1137/1.9781611973594.

[41] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. DyRep: Learning Representations over Dynamic Graphs. In *Proceedings of the 7th International Conference on Learning Representations*, pages 1–25, 2019.

[42] H. Wang and P. Li. Unsupervised Learning for Combinatorial Optimization Needs Meta Learning. In *Proceedings of the 11th International Conference on Learning Representations*, pages 1–18, 2023.

[43] H. Wang, N. Wu, H. Yang, C. Hao, and P. Li. Unsupervised Learning for Combinatorial Optimization with Principled Objective Relaxation. In *Advances in Neural Information Processing Systems*, volume 35, pages 31444–31458, 2022.

[44] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020. doi:10.1109/TNNLS.2020.2978386.

[45] M. Xiao and H. Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255(1):126–146, 2017. doi:10.1016/j.ic.2017.06.001.

[46] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Inductive representation learning on temporal graphs. In *Proceedings of the 8th International Conference on Learning Representations*, pages 1–19, 2020.

[47] M. Yau, N. Karalias, E. Lu, J. Xu, and S. Jegelka. Are Graph Neural Networks Optimal Approximation Algorithms? In *Advances in Neural Information Processing Systems*, volume 37, pages 73124–73181, 2024. doi:10.52202/079017-2328.

[48] L. Yu, L. Sun, B. Du, and W. Lv. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. In *Advances in Neural Information Processing Systems*, volume 36, pages 67686–67700, 2023.

[49] D. Zhang, H. Dai, N. Malkin, A. C. Courville, Y. Bengio, and L. Pan. Let the Flows Tell: Solving Graph Combinatorial Problems with GFlowNets. In *Advances in Neural Information Processing Systems*, volume 36, pages 11952–11969, 2023.

[50] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: A comprehensive review. *Computational Social Networks*, 6(1):11, 2019. doi:10.1186/s40649-019-0069-y.

[51] W. Zheng, C. Piao, H. Cheng, and J. X. Yu. Computing a Near-Maximum Independent Set in Dynamic Graphs. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 76–87, 2019. doi:10.1109/ICDE.2019.00016.