Learning What to Defer for Maximum Independent Sets

Sungsoo Ahn [1] **Younggyo Seo** [2] **Jinwoo Shin** [2 1]

Abstract

Designing efficient algorithms for combinatorial optimization appears ubiquitously in various scientific fields. Recently, deep reinforcement learning (DRL) frameworks have gained considerable attention as a new approach: they can automate the design of a solver while relying less on sophisticated domain knowledge of the target problem. However, the existing DRL solvers determine the solution using a number of stages proportional to the number of elements in the solution, which severely limits their applicability to large-scale graphs. In this paper, we seek to resolve this issue by proposing a novel DRL scheme, coined learning what to defer (LwD), where the agent adaptively shrinks or stretch the number of stages by learning to distribute the element-wise decisions of the solution at each stage. We apply the proposed framework to the maximum independent set (MIS) problem, and demonstrate its significant improvement over the current state-of-the-art DRL scheme. We also show that LwD can outperform the conventional MIS solvers on large-scale graphs having millions of vertices, under a limited time budget.

1. Introduction

Combinatorial optimization is an important mathematical field addressing fundamental questions of computation, where its popular examples include the maximum independent set (MIS, Miller & Muller, 1960), satisfiability (SAT, Schaefer, 1978) and traveling salesman problem (TSP, Voigt, 1831). Such problems arise in various applications, e.g., sociology (Harary & Ross, 1957), operations research (Feo et al., 1994) and bioinformatics (Gardiner et al., 2000). However, most combinatorial optimization problems are NP-hard and exact solutions are typically intractable to find in practical situations. Over the past decades, researchers have made significant efforts for resolving this issue by designing fast heuristic solvers (Knuth, 1997; Biere et al., 2009; Mezard et al., 2009) that generate approximate solutions.

Recently, the remarkable progress in deep learning has stimulated increased interest in learning such heuristics based on deep neural networks (DNNs). Such learning-based approaches are attractive for being able to train a solver on a particular problem while relying less on expert knowledge. As the most straight-forward way, supervised learning schemes train the DNNs to imitate the solutions obtained from existing solvers (Vinyals et al., 2015; Li et al., 2018; Selsam et al., 2019). However, the resulting quality and applicability are constrained by those of existing solvers. An ideal direction is to discover new solutions in a fully unsupervised manner, potentially outperforming those based on domain-specific knowledge.

To this end, recent works (Bello et al., 2016; Khalil et al., 2017; Deudon et al., 2018; Kool et al., 2019) consider using deep reinforcement learning (DRL) based on the Markov decision process (MDP) naturally designed with rewards derived from the optimization objective of the target problem. Then, the corresponding agent can be trained based on existing training schemes of DRL, e.g., Bello et al. (Bello et al., 2016) trained a TSP solver based on actor-critic framework. Such DRL-based methods are especially attractive since they can even solve unexplored problems where domain knowledge is scarce, and no efficient heuristic is known.

Unfortunately, the existing DRL-based methods struggle to compete with the existing highly optimized solvers. In particular, the gap becomes significant when the problem requires solutions with higher dimensions. The reasoning is that they mostly emulate greedy algorithms (Bello et al., 2016; Khalil et al., 2017), i.e., choosing one element of the solution at each stage of MDP. Such a procedure quickly becomes computationally prohibitive for obtaining a large-scale solution. This motivates us to seek for an alternative DRL scheme.

Contribution. In this paper, we propose a new DRL framework, coined learning what to defer (LwD), for solving large-scale combinatorial optimization problems. Our framework is designed for locally decomposable problems,

[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea [2]Graduate School of AI, KAIST, Daejeon, South Korea. Correspondence to: Sungsoo Ahn <sungsoo.ahn@kaist.ac.kr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

where the feasibility constraint and the objective can be decomposed by locally connected variables (in a graph). Representative examples of such problems include satisfiability (Schaefer, 1978), maximum cut (MAXCUT, Garey & Johnson, 1979), and graph coloring (Brooks, 1941).[1] Among them, we focus on the maximum independent set (MIS) problem; it is a well-studied prototype of NP-hard combinatorial optimization with highly-optimized solvers (Andrade et al., 2012; Lamm et al., 2017; Chang et al., 2017; Hespe et al., 2019b), where we would like to showcase that our scheme can even compete with existing state-of-the-art solvers. The MIS problem has been used in various applications including classification theory (Feo et al., 1994), computer vision (Sander et al., 2008) and communication (Jiang & Walrand, 2010).

The main novelty of LwD is automatically stretching the determination of the solution throughout multiple steps. In particular, the agent iteratively acts on every undetermined vertex for either (a) determining the membership of the vertex in the solution or (b) deferring the determination to be made in later steps (see Figure 1 for illustration). Inspired by the celebrated survey propagation (Braunstein et al., 2005) for solving the SAT problem, LwD could be interpreted as prioritizing the "easier" decisions to be made first, which in turn simplifies the harder ones by eliminating the source of uncertainties. Compared to the greedy strategy (Khalil et al., 2017) which determines the membership of a single vertex at each step, our framework brings significant speedup by learning to make many element-wise decisions at once (and deferring the rest).

Based on such a speedup, LwD can solve the optimization problem by generating a large number of candidate solutions in a limited time budget, then reporting the best solution among them. For this scenario, it is beneficial for the algorithm to generate diverse candidates. To this end, we additionally give a novel diversification bonus to our agent during training, which explicitly encourages the agent to generate a large variety of solutions. To be specific, we create a "coupling" of MDPs to generate two solutions for the given MIS problem and reward the agents for a large deviation between the solutions. The resulting reward efficiently improves the performance at the evaluation.

We empirically validate LwD on various types of graphs including the Erdös-Rényi (Erdős & Rényi, 1960) model, the Barabási-Albert (Albert & Barabási, 2002) model, the SATLIB (Hoos & Stützle, 2000) benchmark and real-world graphs. Our algorithm shows consistent superiority over the existing state-of-the-art DRL method (Khalil et al., 2017). Remarkably, it often outperforms the state-of-the-art MIS solver (KaMIS, Hespe et al., 2019a), particularly on large-scale graphs, e.g., in our machine, LwD achieves better objectives $3637/276 \approx 13$ times faster, compared to KaMIS on the Barabási-Albert graph with two million vertices. Furthermore, we also show that our fully learning-based scheme generalizes well even to graph types unseen during training and works well even for other locally decomposable combinatorial optimization: the maximum weighted independent set problem, the prize collecting maximum independent set problem (Hassin & Levin, 2006), the MAXCUT problem (Garey & Johnson, 1979), and the maximum-a-posteriori inference problem for the Ising model (Onsager, 1944).

[1] We note that TSP is not locally decomposable since it has a global constraint of forming a Hamiltonian cycle.

2. Related Works

The maximum independent set (MIS) problem is a prototypical NP-hard task where its optimal solution cannot be approximated by a constant factor in polynomial time unless P = NP (Hastad, 1996). Since the problem is NP-hard even to approximate, existing methods (Tomita et al., 2010; San Segundo et al., 2011) for exactly solving the MIS problem suffer from a prohibitive amount of computation in large graphs. To resolve this, researchers have developed a wide range of approximate solvers for the MIS problem (Andrade et al. 2012, Lamm et al. 2017, Chang et al. 2017, Hespe et al. 2019b). Notably, Lamm et al. (2017) developed a combination of an evolutionary algorithm with graph kernelization techniques for the MIS problem. Later, Chang et al. (2017) and Hespe et al. (2019b) further improved the graph kernelization technique by introducing new reduction rules and parallelization based on graph partitioning, respectively.

In the context of solving combinatorial optimization using neural networks, Hopfield & Tank (1985) first applied the Hopfield-network for solving the traveling salesman problem (TSP). Since then, several works also tried to utilize neural networks in different forms, e.g., see Smith (1999) for a review of such papers. Such works mostly solve combinatorial optimization through online learning, i.e., training was performed for each problem instance separately. More recently, Vinyals et al. (2015) and Bello et al. (2016) proposed to solve TSP by training attention-based neural networks with offline learning. They showed promising results that stimulated many other works to use neural networks for solving combinatorial problems (Khalil et al. 2017, Li et al. 2018, Selsam et al. 2019, Deudon et al. 2018, Amizadeh et al. 2018, Kool et al. 2019). Importantly, Khalil et al. (2017) proposed a reinforcement learning framework for solving the minimum vertex cover problem, which is equivalent to solving the MIS problem. They query the agent for each vertex to add as a new member of the vertex cover at each step of the Markov decision process. However, such a greedy procedure hurts the scalability to large-scale graphs, as we mentioned in Section 1. Next, Li et al. (2018) aim

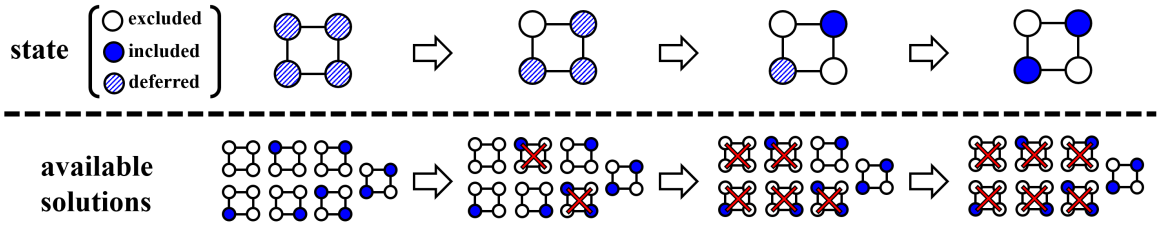


Figure 1. Illustration of the proposed Markov decision process.

developing a supervised learning framework for solving the MIS problem. At an angle, their framework is similar to ours; they allow stretching the determination of the solution over multiple steps. However, their scheme for stretching the determination is hand-designed and not trainable from data. Furthermore, their scheme requires supervisions, which are (a) highly sensitive to the quality of solvers used for extracting them and (b) often too expensive or almost impossible to obtain.

3. Learning What to Defer

In this section, we focus on describing the learning what to defer (LwD) framework for the maximum independent set (MIS) problem. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$, an *independent set* is a subset of vertices $\mathcal{I} \subseteq \mathcal{V}$ where no two vertices in the subset are adjacent to each other. A solution to the MIS problem can be represented as a binary vector $\boldsymbol{x} = [x_i : i \in \mathcal{V}] \in \{0, 1\}^{\mathcal{V}}$ with maximum cardinality $\sum_{i \in \mathcal{V}} x_i$, where each element x_i indicates the element-wise decision on the membership of vertex i in the independent set $\mathcal{I}$, i.e., $x_i = 1$ if and only if $i \in \mathcal{I}$. Initially, the algorithm has no assumption about its output, i.e., both $x_i = 0$ and $x_i = 1$ are possible for all $i \in \mathcal{V}$. At each iteration, the agent acts on each undetermined vertex i by either (a) deciding its membership to be a certain value, i.e., set $x_i = 0$ or $x_i = 1$, or (b) deferring the decision to be made later iterations. The agent repeats the action until all the membership of vertices in the independent set is determined. One can interpret such a strategy as progressively narrowing down the set of candidate solutions at each iteration (see Figure 1 for illustration). Intuitively, the act of deferring prioritizes to choose the values of the "easier" vertices first. After each decision, "hard" vertices become easier since decisions on its surrounding easy vertices are better known.

We note that LwD is applicable to combinatorial optimization problems other than the MIS problem, e.g., by considering the decision of non-binary vector $\boldsymbol{x}$. Especially, the LwD framework is attractive to apply for locally decomposable combinatorial problems, i.e., problems where the feasibility constraints are decomposable by locally connected variables. Popular problems such as satisfiability (Schaefer, 1978), maximum cut (Garey & Johnson, 1979), and graph coloring (Brooks, 1941) also fall into this category. For these problems, making multiple "local" decisions on variables at once likely does not violate the feasibility, and this is what LwD needs for its efficient implementation.

3.1. Deferred Markov Decision Process

We formulate the proposed algorithm as a pair of a MDP and an agent, i.e., a policy. At a high level, the MDP initializes its states on the given graph and generates a solution at termination. We further decompose the MIS objective into a summation of rewards distributed over the MDP.

State. Each state of the MDP is represented as a *vertex-state* vector $\boldsymbol{s} = [s_i : i \in \mathcal{V}] \in \{0, 1, *\}^{\mathcal{V}}$, where the vertex $i \in \mathcal{V}$ is determined to be excluded or included in the independent set whenever $s_i = 0$ or $s_i = 1$, respectively. Otherwise, $s_i = *$ indicates that the determination has been deferred and expected to be made in later iterations. The MDP is initialized with the deferred vertex-states, i.e., $s_i = *$ for all $i \in \mathcal{V}$, and terminated when (a) there is no deferred vertex-state left or (b) time limit is reached.

Action. Actions correspond to new assignments for the next state of vertices. Since vertex-states of included and excluded vertices are immutable, the assignments are defined only on the deferred vertices. It is represented as a vector $\boldsymbol{a}_* = [a_i : i \in \mathcal{V}_*] \in \{0, 1, *\}^{\mathcal{V}_*}$ where $\mathcal{V}_*$ denotes a set of current deferred vertices, i.e., $\mathcal{V}_* = \{i : i \in \mathcal{V}, x_i = *\}$.

Transition. Given two consecutive states $\boldsymbol{s}, \boldsymbol{s}'$ and the corresponding assignment $\boldsymbol{a}_*$, the transition $P_{\boldsymbol{a}_*}(\boldsymbol{s}, \boldsymbol{s}')$ consists of two deterministic phases: the *update phase* and the *clean-up phase*. The update phase takes account of the assignment $\boldsymbol{a}_*$ generated by the policy for the corresponding vertices $\mathcal{V}_*$ to result in an intermediate vertex-state $\widehat{\boldsymbol{s}}$, i.e., $\widehat{s}_i = a_i$ if $i \in \mathcal{V}_*$ and $\widehat{s}_i = s_i$ otherwise. The clean-up phase modifies the intermediate vertex-state vector $\widehat{\boldsymbol{s}}$ to yield a valid vertex-state vector $\boldsymbol{s}'$, where the included vertices are only adjacent to the excluded vertices. To this end, whenever there exists a pair of included vertices adjacent to each other, they are both mapped back to the deferred vertex-state. Next, the MDP excludes any deferred vertex neighboring with an included

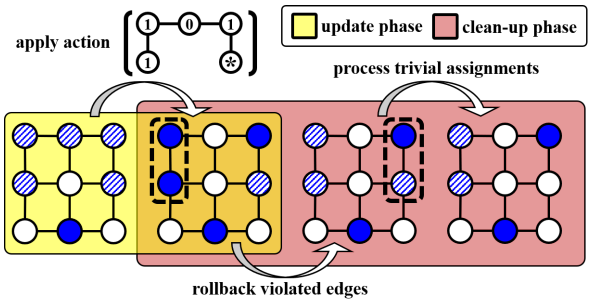


Figure 2. Illustration of the transition function.

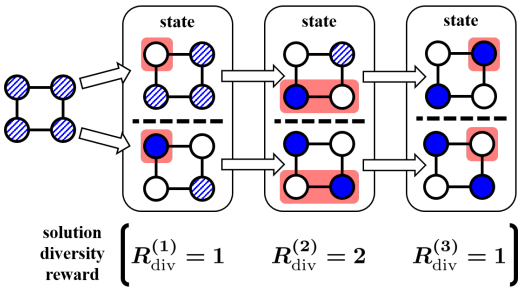


Figure 3. Illustration of the solution diversity reward.

vertex.[2] See Figure 2 for a more detailed illustration of the transition between two states.

When the MDP makes all the determination, i.e., at termination, one can (optionally) improve the determined solution by applying the 2-improvement local search algorithm (Feo et al., 1994; Andrade et al., 2012); it increases the size of the independent set greedily by removing one vertex and adding two vertices until no modification is possible.

Reward. Finally, we define the *cardinality reward* $R(\boldsymbol{s}, \boldsymbol{s}')$ as the increase in cardinality of included vertices. To be specific, we define it as $R(\boldsymbol{s}, \boldsymbol{s}') = \sum_{i \in \mathcal{V}_* \setminus \mathcal{V}'_*} s'_i$, where $\mathcal{V}_*$ and $\mathcal{V}'_*$ are the set of vertices with deferred vertex-state with respect to $\boldsymbol{s}$ and $\boldsymbol{s}'$, respectively. By doing so, the overall reward of the MDP corresponds to the cardinality of the independent set returned by our algorithm.

3.2. Diversification Reward

Next, we introduce an additional *diversification reward* for encouraging diversification of solutions generated by the agent. Such a regularization is motivated by our evaluation method, which samples multiple candidate solutions to report the best one as the final output. For such scenarios, it would be beneficial to generate diverse solutions of a high maximum score, rather than ones of similar scores. To this end, we "couple" two copies of MDPs defined on an identical graph $\mathcal{G}$ (as in Section 3.1) into a new MDP. Then the new MDP is associated with a pair of distinct vertex-state vectors $(\boldsymbol{s}, \bar{\boldsymbol{s}})$ from the coupled MDPs. Furthermore, the corresponding agents work independently to result in a pair of solutions $(\boldsymbol{x}, \bar{\boldsymbol{x}})$. Then, we directly reward the deviation between the coupled solutions in terms of ℓ_1-norm, i.e., $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_1$. To be specific, the deviation is decomposed into rewards in each iteration of the MDP defined by

$$R_{\text{div}}(\boldsymbol{s}, \boldsymbol{s}', \bar{\boldsymbol{s}}, \bar{\boldsymbol{s}}') = \sum_{i \in \widehat{\mathcal{V}}} |s'_i - \bar{s}'_i|,$$

where $\widehat{\mathcal{V}} = (\mathcal{V}_* \setminus \mathcal{V}'_*) \cup (\bar{\mathcal{V}}_* \setminus \bar{\mathcal{V}}'_*)$ and $(\boldsymbol{s}', \bar{\boldsymbol{s}}')$ denotes the next pair of vertex-states in the coupled MDP. Note that $\widehat{\mathcal{V}}$ indicates the most recently updated vertices in each MDP. In practice, such a reward R_{div} can be used along with the maximum entropy regularization to achieve the best performance. See Figure 3 for an example of coupled MDP with the proposed reward.

We remark that the entropy regularization (Williams & Peng, 1991) of the policy plays a similar role (of encouraging exploration) to our diversification reward. However, the entropy regularization attempts to generate diverse trajectories of the same MDP, which does not necessarily lead to diverse solutions at last, since there exist many trajectories resulting in the same solution (see Section 3.1). We instead directly maximize the diversity among solutions by a new reward term.

3.3. Training with Proximal Policy Optimization

Our algorithm is based on actor-critic training with policy network $\pi(\boldsymbol{a}|\boldsymbol{s})$ and value network $V(\boldsymbol{s})$ following the GraphSAGE architecture (Hamilton et al., 2017). Each network consists of multiple layers $h^{(n)}$ with $n = 1, \cdots, N$ where the n-th layer with weights $\mathbf{W}_1^{(n)}$ and $\mathbf{W}_2^{(n)}$ performs the following transformation on input $\mathbf{H}$:

$$h^{(n)}(\mathbf{H}) = \text{ReLU}\big(\mathbf{H}\mathbf{W}_1^{(n)} + \mathbf{D}^{-\frac{1}{2}}\mathbf{B}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}\mathbf{W}_2^{(n)}\big).$$

Here $\mathbf{B}$ and $\mathbf{D}$ correspond to adjacency and degree matrix of the graph $\mathcal{G}$, respectively. At the final layer, the policy and value networks apply softmax function and graph readout function with sum pooling (Xu et al., 2019) instead of ReLU to generate actions and value estimates, respectively. We utilize vertex degrees and the current iteration-index of the MDP as the input features of the neural network. The iteration-index of MDP used for input of the policy and

[2] We note that such a clean-up phase can be replaced by training the agent with a soft penalty for solutions corresponding to an invalid independent set. In our experiments, such an algorithm also performs well with only marginal degradation in its performance.

value networks was normalized by the maximum number of iterations. We only consider the subgraph that is induced on the deferred vertices $\mathcal{V}_*$ as the input of the networks since the determined part of the graph no longer affects the future rewards of the MDP.

To train the agent, we use the proximal policy optimization (Schulman et al., 2017). To be specific, we train the networks to maximize the following objective:

$$\mathcal{L} := \mathbb{E}_t\left[\min\left(A^{(t)}\prod_{i\in\mathcal{V}} r_i^{(t)}, A^{(t)}\prod_{i\in\mathcal{V}}\widetilde{r}_i^{(t)}\right)\right],$$

$$A^{(t)} = \sum_{\ell=0}^{T-t}\left(R^{(t+\ell)} + \alpha R_{\text{div}}^{(t+\ell)}\right) - V(\boldsymbol{s}^{(t)}),$$

$$r_i^{(t)} = \frac{\pi(a_i^{(t)}|\boldsymbol{s}^{(t)})}{\pi_{\text{old}}(a_i^{(t)}|\boldsymbol{s}^{(t)})},$$

$$\widetilde{r}_i^{(t)} = \texttt{clip}(r_i^{(t)}, 1-\varepsilon, 1+\varepsilon),$$

where $\boldsymbol{s}^{(t)}$, $\boldsymbol{a}^{(t)}$, $R^{(t)}$ and $R_{\text{div}}^{(t)}$ denotes the t-th vertex-state vector, action vector, cardinality reward, and diversification reward, respectively. In addition, π_{old} is the policy network with parameters from the previous iteration of updates, T is the maximum number of steps for the MDP, and $\alpha \geq 0$ is the hyperparameter to be tuned. The clipping function $\texttt{clip}(r, r_{\texttt{min}}, r_{\texttt{max}})$ projects the ratio of action probabilities r into an interval $[r_{\texttt{min}}, r_{\texttt{max}}]$ for conservatively updating the agent. Note that the clipping is applied for each vertex, unlike the original framework where clipping is applied once (Schulman et al., 2017).

4. Experiments

In this section, we report experimental results on the proposed learning what to defer (LwD) framework described in Section 3 for solving the maximum independent set (MIS) problem. To this end, we evaluate our framework with and without the local search element described in Section 3.1; we coin the algorithms by LwD† and LwD, respectively. Note that we also include the evaluations of our framework on other locally decomposable combinatorial problems to demonstrate its potential for being applied to a broader domain (See Section 4.3). We perform every experiment using a single GPU (NVIDIA RTX 2080Ti) and a single CPU (Intel Xeon E5-2630 v4). For a more detailed description for the experiments, see Appendix A.

Baselines. For comparison with the deep learning-based methods, we consider the deep reinforcement learning (DRL) framework by Khalil et al. (2017), coined S2V-DQN, and supervised learning (SL) framework by Li et al. (2018), coined TGS. We also consider its variant, coined TGS†, equipped with additional graph reduction and local search algorithms. We remark that other existing deep learning-based schemes for solving combinatorial optimization, e.g., works done by Bello et al. (2016) and Kool et al. (2019), are not comparable since they propose a neural architecture specialized to TSP-like problems.

We additionally consider two conventional MIS solvers as competitors. First, we consider the integer programming solver IBM ILOG CPLEX Optimization Studio V12.9.0 (ILO, 2014), coined CPLEX. We also consider the MIS solver based on the recently developed techniques (Lamm et al., 2015; 2017; Hespe et al., 2019b), coined KaMIS, which won the PACE 2019 challenge at the vertex cover track (Hespe et al., 2019a).

We remark that comparisons among the algorithms should be made carefully by accounting for the scope of applications concerning each algorithm. In particular, KaMIS, TGS† and LwD†, rely on heuristics specialized for the MIS problem, e.g., the local search algorithm, while CPLEX, S2V-DQN, and LwD can be applied even to non-MIS problems. The integer programming solver CPLEX can provide the proof of optimality in addition to its solution. Furthermore, TGS and TGS† are only applicable to problems where solvers are available for obtaining supervised solutions.

Datasets. Experiments were conducted on a broad range of graphs to include both real-world and large-scale graphs. First, we consider random graphs generated from models designed to imitate the characteristics of real-world graphs. Specifically, we consider the models proposed by Erdös-Rényi (ER, Erdős & Rényi, 1960), Barabási-Albert (BA, Albert & Barabási, 2002), Holme and Kim (HK, Holme & Kim, 2002), and Watts-Strogatz (WS, Watts & Strogatz, 1998). For convenience of notation, the synthetic datasets are specified by their type of generative model and size, e.g., ER-$[N_{\texttt{min}}, N_{\texttt{max}}]$ denotes the set of ER graphs generated with the number of vertices uniformly sampled from the interval $[N_{\texttt{min}}, N_{\texttt{max}}]$. Next, we consider real-world graph datasets, namely the SATLIB, PPI, REDDIT, as-Caida, Citation, Amazon, and Coauthor datasets constructed from the SATLIB benchmark (Hoos & Stützle, 2000), protein-protein interactions (Hamilton et al., 2017), social networks (Yanardag & Vishwanathan, 2015), road networks (Leskovec & Sosič, 2016), co-citation network (Yang et al., 2016), co-purchasing network (McAuley et al., 2015), and academic relationship network (Sen et al., 2008), respectively.

4.1. Performance Evaluation

We first show the performance of our algorithm along with other baselines for the MIS problem.

Moderately sized graphs. First, we provide the experimental results for the datasets with the number of vertices up to 30 000.In Table 1, we observe that LwD† consistently achieves the best objective except for the SATLIB dataset.

Table 1. Objectives achieved from the MIS solvers on moderately size dataset, where the best objectives are marked in bold. Running times (in seconds) of the deep-learning based algorithms are provided in brackets. When running CPLEX and KaMIS, time limits were set by 5 and 30 seconds for {ER, BA, HK, WS} and {SATLIB, PPI, REDDIT, as-Caida} datasets, respectively, so that they spend more (or comparable) time than LwD and LwD† to find their solutions. The minimum and the maximum number of vertices in each dataset are denoted by $N_{\texttt{min}}$ and $N_{\texttt{max}}$, respectively.

			Classic		SL-based		RL-based		
Type	$N_{\texttt{min}}$	$N_{\texttt{max}}$	CPLEX	KaMIS	TGS	TGS†	S2V-DQN	LwD	LwD†
ER	50	100	**21.11**	**21.11**	19.90 (0.32)	**21.11** (0.65)	20.61 (0.03)	21.04 (0.01)	**21.11** (0.15)
	100	200	27.87	**27.95**	24.94 (1.46)	**27.95** (1.54)	26.27 (0.08)	27.67 (0.03)	**27.95** (0.61)
	400	500	31.73	39.61	33.46 (0.93)	39.43 (12.37)	35.05 (0.63)	38.29 (0.16)	**39.81** (5.91)
BA	50	100	**32.07**	**32.07**	31.77 (0.24)	**32.07** (0.25)	31.96 (0.02)	**32.07** (0.01)	**32.07** (0.11)
	100	200	**66.07**	**66.07**	65.25 (0.33)	**66.07** (0.52)	65.52 (0.05)	66.05 (0.01)	**66.07** (0.22)
	400	500	**204.1**	**204.1**	201.2 (0.72)	**204.1** (7.86)	202.9 (0.18)	204.0 (0.02)	**204.1** (0.87)
HK	50	100	**23.95**	**23.95**	23.39 (0.29)	**23.95** (0.60)	23.77 (0.03)	**23.95** (0.02)	**23.95** (0.15)
	100	200	**50.15**	**50.15**	48.74 (0.43)	**50.15** (2.00)	49.64 (0.05)	50.12 (0.04)	**50.15** (0.34)
	400	500	156.8	**157.0**	152.1 (0.92)	**157.0** (7.63)	152.8 (0.22)	156.8 (0.14)	**157.0** (1.63)
WS	50	100	**23.08**	**23.08**	21.90 (0.34)	**23.08** (0.71)	22.64 (0.03)	23.07 (0.03)	**23.08** (0.11)
	100	200	47.17	**47.18**	44.55 (0.49)	**47.18** (1.89)	45.39 (0.06)	47.11 (0.06)	**47.18** (0.23)
	400	500	138.3	**143.3**	134.8 (1.15)	143.2 (6.08)	132.2 (0.23)	142.1 (0.17)	**143.3** (0.90)
SATLIB	1209	1347	426.8	**426.9**	418.1 (19.6)	426.7 (63.0)	413.8 (2.3)	424.8 (1.8)	426.7 (7.1)
PPI	591	3480	**1148**	**1148**	1128 (20.9)	**1148** (568.9)	893 (6.3)	1147 (1.8)	**1148** (30.8)
REDDIT-M-5K	22	3648	**370.6**	**370.6**	367.1 (0.88)	**370.6** (1.7)	370.1 (0.1)	**370.6** (0.6)	**370.6** (2.0)
REDDIT-M-12K	2	3782	**303.5**	**303.5**	300.5 (0.75)	**303.5** (22.1)	302.8 (1.9)	292.6 (0.1)	**303.5** (2.0)
REDDIT-B	6	3782	**329.3**	**329.3**	327.6 (0.7)	**329.3** (2.5)	328.6 (0.1)	**329.3** (0.2)	**329.3** (3.0)
as-Caida	8020	26 475	**20 049**	**20 049**	19 921 (65.85)	**20 049** (601.4)	324 (34.8)	**20 049** (6.1)	**20 049** (34.6)

Surprisingly, it even outperforms CPLEX and KaMIS on some datasets, e.g., LwD† achieves a strictly better objective than every other baseline on the ER-$[400, 500]$ dataset. Next, LwD significantly outperforms its DL competitors, i.e., S2V-DQN and TGS, for datasets except for the REDDIT-M-12K dataset. The gap grows for the large-scale dataset, e.g., S2V-DQN underperforms significantly for the as-Caida dataset.

Million-scale graphs. Next, we highlight the scalability of our framework by evaluating the algorithms under the million-scale graphs. To this end, we generate synthetic graphs from the BA, SW, and WS models with half, one, and two million vertices.[3] We use the networks trained on the {BA, HK, WS}-$[400, 500]$ datasets for evaluation on graphs from the same generative model. We run CPLEX and KaMIS with a time limit of 1000 seconds so that they spend more time than our methods to find the solutions.[4] Furthermore, we exclude S2V-DQN from comparison for being computationally infeasible to be evaluated on such large-scale graphs.

[3]The ER model requires a computationally prohibitive amount of memory for the large-scale graph generation.

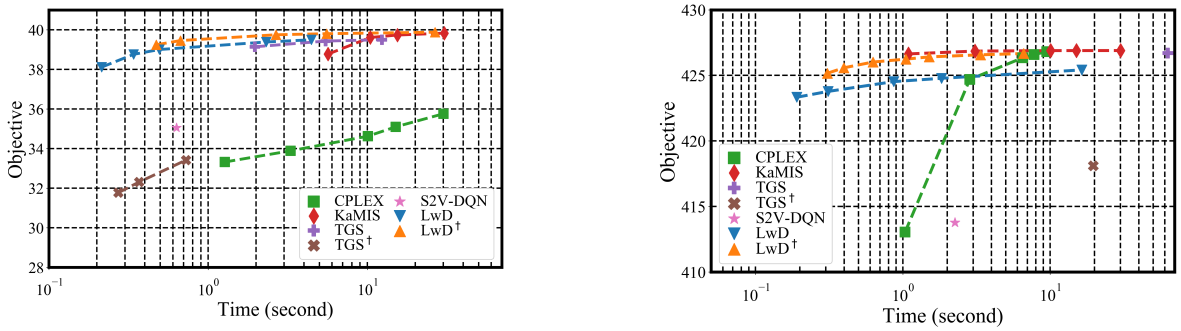
[4]However, the solvers consistently violate the time limit due to their expensive pre-solving process.

In Table 2, we observe that LwD and LwD† mostly outperform other algorithms, i.e., they achieve a better objective in a shorter amount of time. In particular, LwD achieves a higher objective than KaMIS in BA graph with two million vertices along with approximately $\times 13$ speedup. The only exception is the WS graph with five hundred thousand vertices, where LwD and LwD† achieve smaller objective than KaMIS, while still being much faster. Such a result highlights the scalability of our algorithm. It is somewhat surprising to observe that LwD achieves objectives similar to LwD†, while TGS underperforms significantly compared to TGS†. This validates that our method is not sensitive to using the existing heuristics for achieving high performance, unlike the previous method.

Trade-offs between objective and time. We further investigate the trade-offs between objective and time for the considered algorithms. To this end, we evaluate algorithms on ER-$[400, 500]$ and SATLIB datasets with varying numbers of samples or time limits. In Figure 4, it is remarkable to observe that both LwD and LwD† achieves a better objective than the CPLEX solver on both datasets under reasonably limited time. Furthermore, LwD and LwD† outperforms KaMIS for running time smaller than 10 and 20 seconds, respectively.

Table 2. Objectives achieved from the MIS solvers on large-scale datasets, where the best objectives are marked in bold. Running times (in seconds) are provided in brackets, and the number of vertices is denoted by N. Out of budget (OB) indicates the runs violating the time budget (15 000 seconds) or the memory budgets (128GB RAM for CPU and 12GB VRAM for GPU).

		Classic		SL-based		RL-based	
Type	N	CPLEX	KaMIS	TGS	TGS†	LwD	LwD†
BA	500 000	137 821 (1129)	228 123 (1002)	227 701 (344)	228 733 (6211)	228 803 (45)	**228 829** (340)
	1 000 000	275 633 (1098)	457 541 (2502)	455 354 (620)	457 073 (10 484)	457 698 (117)	**457 752** (651)
	2 000 000	551 767 (1152)	909 988 (3637)	910 856 (1016)	OB	915 887 (276)	**915 968** (1296)
HK	500 000	90 012 (1428)	175 153 (1477)	173 852 (465)	176 253 (777)	176 850 (96)	**177 143** (519)
	1 000 000	179 326 (1172)	347 350 (4463)	350 819 (887)	353 244 (10 415)	353 504 (248)	**353 723** (1151)
	2 000 000	OB	695 544 (10 870)	OB	OB	706 975 (648)	**707 422** (1601)
WS	500 000	135 217 (1424)	**157 298** (1002)	153 190 (354)	155 230 (10 172)	155 086 (62)	155 574 (403)
	1 000 000	270 526 (1093)	303 810 (1699)	308 065 (1009)	OB	310 308 (144)	**311 041** (725)
	2 000 000	540 664 (1159)	603 502 (4252)	611 057 (1628)	OB	620 615 (345)	**621 687** (1388)



(a) ER-$(400, 500)$ dataset

(b) SATLIB dataset

Figure 4. Evaluation of trade-off between time and objective for the ER-$(400, 500)$ dataset (upper-left side is of better trade-off).

Generalization capability. Finally, we examine the potential of the deep learning-based solvers as generic solvers, i.e., whether the solvers generalize well to graph types unseen during training. To this end, we evaluate LwD, LwD†, TGS, TGS†, and S2V-DQN on Citation, Amazon and Coauthor graph datasets. In Table 3, we observe that LwD† achieves near-optimal objective for all of the considered graphs, despite being trained on graphs with different type and a smaller number of vertices. On the other side, S2V-DQN underperforms significantly compared to LwD, i.e., it achieves worse approximation ratios while being slower. We also observe that TGS achieves similar approximation ratios compared to LwD† but takes a longer time in our experimental setups.

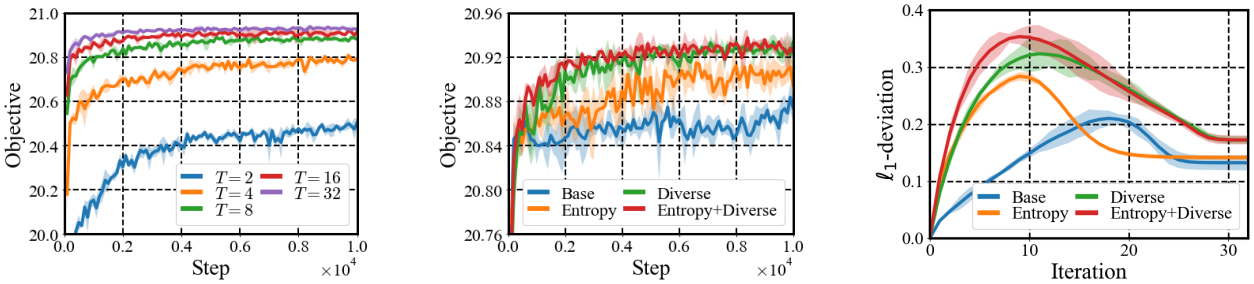
4.2. Ablation Study

We now ablate each component of our algorithm to validate its effectiveness. First, we confirm that stretching the determination process indeed improves the performance of LwD. Then we show that the effectiveness of solution diversification reward.

Deferring the decision. We first show that "deferring" the decision for the MIS problem indeed helps to solve the MIS problem. Specifically, we experiment with varying the maximum number of iterations T in MDP by $T \in \{2, 4, 8, 16, 32\}$ on ER-$(50, 100)$ dataset. Figure 5a reports the corresponding training curves. We observe that the performance of LwD improves whenever the agent receives more time to generate the final solution, which verifies that the deferring the decisions plays a crucial role in solving the MIS problem.

Table 3. Approximation ratios of the deep-learning based MIS solvers on real-world graphs, i.e., Citation-{Cora, Citeseer}, Amazon-{Photo, Computers}, and Coauthor-{CS, Physics}, unseen during training. Best approximation ratios are marked in bold. Running times (in seconds) are provided in brackets, and the number of vertices is denoted by N. For computing the approximation ratios, We compute the optimal solutions from running the CPLEX solver without any time limit.

		SL-based		RL-based		
Type	N	TGS	TGS†	S2V-DQN	LwD	LwD†
Citation-Cora	2708	**1.00** (4)	**1.00** (4)	0.96 (3)	**1.00** (3)	**1.00** (3)
Citation-Citeseer	3327	**1.00** (3)	**1.00** (3)	0.99 (3)	**1.00** (2)	**1.00** (4)
Amazon-Photo	7487	0.99 (9)	**1.00** (485)	0.27 (66)	0.99 (4)	**1.00** (33)
Amazon-Computers	13 381	0.99 (8)	**1.00** (823)	0.26 (236)	0.99 (3)	**1.00** (101)
Coauthor-CS	18 333	0.99 (17)	**1.00** (80)	0.88 (197)	**1.00** (3)	**1.00** (78)
Coauthor-Physics	34 493	0.98 (52)	**1.00** (1304)	0.19 (1564)	0.98 (9)	**1.00** (186)



(a) Performance with varying T (b) Contribution of each regularizers (c) Deviation in intermediate stages

Figure 5. Illustration of ablation studies done on ER-$(50, 100)$ dataset. The solid line and shaded regions represent the mean and standard deviation across 3 runs respectively Note that the standard deviation in (c) was enlarged ten times for better visibility.

Table 4. Objectives achieved by LwD and CPLEX for solving various combinatorial optimization problems on the ER datasets, where the best objectives are marked in bold. Running times (in seconds) for LwD are provided in brackets and CPLEX was run under time limit of 5 seconds. The minimum and the maximum number of vertices are denoted by $N_{\texttt{min}}$ and $N_{\texttt{max}}$, respectively.

			MWIS		PCMIS		MAXCUT		Ising	
Type	$N_{\texttt{min}}$	$N_{\texttt{max}}$	CPLEX	LwD	CPLEX	LwD	CPLEX	LwD	CPLEX	LwD
ER	50	100	**21.46**	**21.46** (0.16)	**22.64**	21.77 (0.01)	**251.6**	240.4 (0.01)	108.8	**151.9** (0.01)
	100	200	27.94	**28.44** (0.39)	27.12	**29.56** (0.11)	222.5	**1029** (0.01)	-479.7	**406.8** (0.02)
	400	500	34.29	**39.21** (0.54)	4.56	**38.39** (0.42)	412.4	**7747** (0.09)	-14180	**1899** (0.12)

Solution diversification reward. Next, we inspect the contribution of the solution diversity reward used in our algorithm. To this end, we trained agents with four options: (a) without any exploration bonus, coined Base, (b) with the conventional entropy bonus (Williams & Peng, 1991), coined Entropy, (c) with the proposed diversification bonus, coined Diverse, and (d) with both of the bonuses, coined Entropy+Diverse. Figure 5b demonstrates the corresponding training curves for validation scores. We observe that the agent trained with the proposed diversification bonus outperforms other agents in terms of validation score, confirming the effectiveness of our proposed reward. In addition, combining both methods, i.e., Entropy+Diverse, achieves the best performance.

Finally, we verify our claim that the maximum entropy regularization fails to capture the diversity of solutions effectively, while the proposed solution diversity reward term does. To this end, we compare the fore-mentioned agents with respect to the ℓ_1-deviations between the coupled inter-

mediate vertex-states $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$, defined as $|\{i : i \in \mathcal{V}, s_i \neq \bar{s}_i\}|/|\mathcal{V}|$. We show the corresponding results in Figure 5c. We observe that the entropy regularization promotes large deviation during the intermediate stages, but converges to solutions with smaller deviation. On the contrary, agents trained on diversification rewards succeed in enlarging the deviation between the final solutions.

4.3. Other Combinatorial Optimizations

Now we evaluate our framework on other locally decomposable combinatorial optimization problems, the maximum weighted independent set problem (MWIS), the prize collecting maximum independent set problem (PCMIS, Hassin & Levin, 2006), the maximum cut problem (MAXCUT, Garey & Johnson, 1979) and the maximum-a-posteriori inference problem for the Ising models (Onsager, 1944). We compare our algorithm to CPLEX, which is also capable of solving the considered problems with its integer programming framework. See Appendix B for detailed descriptions of the problems.

We report the results of the experiments conducted on the ER datasets in Table 4. Surprisingly, we observe that LwD outperforms CPLEX for most problems and graphs under the time limit of 5 seconds. In particular, CPLEX fails to produce reasonable solutions for the PCMIS and the Ising problems for ER-$[100, 200]$ and ER-$[400, 500]$ graphs on the PCMIS, MAXCUT and the Ising problems. This is because the CPLEX solves the PCMIS, MAXCUT and the Ising problems by hard integer quadratic programming (IQP). Such an issue does not exist in the MIS and MWIS problems as CPLEX solves them by integer linear programming (ILP), which is easier to solve. The proposed LwD framework does not rely on such domain-specific knowledge of IQP vs. ILP, and is more robust under various problems in our experiments.

5. Conclusion

In this paper, we propose a new deep reinforcement learning scheme for the maximum independent set problem that is scalable to large graphs. Our main contribution is the framework of learning what to defer, which allows the agent to defer the decisions on vertices for efficient expression of complex structures in the solutions. Through extensive experiments, our algorithm shows performance that is both superior to the existing reinforcement learning baseline and competitive with the conventional solvers.

Acknowledgements

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-01779, A machine learning and statistical inference framework for explainable artificial intelligence). This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)). We thank Hyuntak Cha, Hankook Lee, Kimin Lee, Sangwoo Mo, and Jihoon Tak for providing helpful feedbacks and suggestions in preparing the early version of the manuscript.

References

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Amizadeh, S., Matusevych, S., and Weimer, M. Learning to solve circuit-sat: An unsupervised differentiable approach. 2018.

Andrade, D. V., Resende, M. G., and Werneck, R. F. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547, 2012.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. 2016.

Biere, A., Heule, M., and van Maaren, H. *Handbook of satisfiability*, volume 185. IOS press, 2009.

Braunstein, A., Mézard, M., and Zecchina, R. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

Brooks, R. L. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pp. 194–197. Cambridge University Press, 1941.

Chang, L., Li, W., and Zhang, W. Computing a near-maximum independent set in linear time by reducing-peeling. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1181–1196. ACM, 2017.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. Learning heuristics for the tsp by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 170–181. Springer, 2018.

Erdős, P. and Rényi, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Feo, T. A., Resende, M. G., and Smith, S. H. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

Gardiner, E. J., Willett, P., and Artymiuk, P. J. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information and Computer Sciences*, 40 (2):273–279, 2000.

Garey, M. R. and Johnson, D. S. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Harary, F. and Ross, I. C. A procedure for clique detection using the group matrix. *Sociometry*, 20(3):205–215, 1957.

Hassin, R. and Levin, A. The minimum generalized vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 2(1):66–78, 2006.

Hastad, J. Clique is hard to approximate within n/sup 1-/spl epsiv. In *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 627–636. IEEE, 1996.

Hespe, D., Lamm, S., Schulz, C., and Strash, D. Wegotyoucovered: The winning solver from the pace 2019 implementation challenge, vertex cover track. *arXiv preprint arXiv:1908.06795*, 2019a.

Hespe, D., Schulz, C., and Strash, D. Scalable kernelization for maximum independent sets. *Journal of Experimental Algorithmics (JEA)*, 24(1):1–16, 2019b.

Holme, P. and Kim, B. J. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):026107, 2002.

Hoos, H. H. and Stützle, T. Satlib: An online resource for research on sat. *Sat*, 2000:283–292, 2000.

Hopfield, J. J. and Tank, D. W. neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

Cplex optimization studio. ILOG, IBM, 2014. URL http://www.ibm.com/software/commerce/optimization/cplex-optimizer.

Jiang, L. and Walrand, J. A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking (ToN)*, 18(3): 960–972, 2010.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 2017.

Knuth, D. E. *The art of computer programming*, volume 3. Pearson Education, 1997.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Lamm, S., Sanders, P., and Schulz, C. Graph partitioning for independent sets. In *International Symposium on Experimental Algorithms*, pp. 68–81. Springer, 2015.

Lamm, S., Sanders, P., Schulz, C., Strash, D., and Werneck, R. F. Finding near-optimal independent sets at scale. *Journal of Heuristics*, 23(4):207–229, 2017.

Leskovec, J. and Sosič, R. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, 2018.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52. ACM, 2015.

Mezard, M., Mezard, M., and Montanari, A. *Information, physics, and computation*. Oxford University Press, 2009.

Miller, R. E. and Muller, D. E. A problem of maximum consistent subsets. Technical report, IBM Research Report RC-240, JT Watson Research Center, Yorktown Heights, NY, 1960.

Onsager, L. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65 (3-4):117, 1944.

San Segundo, P., Rodríguez-Losada, D., and Jiménez, A. An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2):571–581, 2011.

Sander, P. V., Nehab, D., Chlamtac, E., and Hoppe, H. Efficient traversal of mesh edges using adjacency primitives. *ACM Transactions on Graphics (TOG)*, 27(5):144, 2008.

Schaefer, T. J. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pp. 216–226. ACM, 1978.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. 2017.

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJMC_iA5tm.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Smith, K. A. Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS Journal on Computing*, 11(1):15–34, 1999.

Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., and Wakatsuki, M. A simple and faster branch-and-bound algorithm for finding a maximum clique. In *International Workshop on Algorithms and Computation*, pp. 191–203. Springer, 2010.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Voigt, B. F. Der handlungsreisende, wie er sein soll und was er zu thun hat, um aufträge zu erhalten und eines glücklichen erfolgs in seinen geschäften gewiss zu sein. *Commis-Voageur, Ilmenau*, 1831.

Watts, D. J. and Strogatz, S. H. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440, 1998.

Williams, R. J. and Peng, J. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 40–48, 2016.