A. Experimental Details

A.1. Implementation of LwD

In this section, we provide additional details for our implementation of the experiments.

Hyperparameter. Every hyperparameter was optimized on a per graph type basis and used across all sizes within each graph type. Throughout every experiment, the policy and the value networks were parameterized by graph convolutional network with 4 layers and 128 hidden dimensions. Every instance of the model was trained for 20 000 updates of proximal policy optimization (Schulman et al., 2017), based on the Adam optimizer with a learning rate of 0.0001. The validation dataset was used for choosing the best performing model while using 10 samples for evaluating the performance. Reward was not decayed throughout the episodes of the Markov decision process. Gradient norms were clipped by a value of 0.5. Both the cardinality reward (defined in Section 3.1) and the solution diversification reward (defined in Section 3.2) were normalized by maximum number of vertices in the corresponding dataset. We further provide details specific to each type of datasets in Table 5. For the compared baselines, we used the default hyperparameters provided in the respective codes.

Table 5. Choice of hyperparameters for the experiments on performance evaluation. The REDDIT column indicates hyperparameters used for the REDDIT-B, REDDIT-M-5K, and REDDIT-M-12K datasets.

Parameters	ER, BA, HK, WS	SATLIB	PPI	REDDIT	as-Caida
Maximum iterations per episode	32	128	128	64	128
Number of unrolling iteration	32	128	128	64	128
Number of environments per batch (graph instances)	32	32	10	64	1
Batch size for gradient step	16	8	8	16	8
Number of gradient steps per update	4	8	8	16	8
Solution diversity reward coefficient	0.1	0.01	0.1	0.1	0.1
Maximum entropy coefficient	0.1	0.01	0.001	0.0	0.1

A.2. Implementation of Baselines

S2V-DQN. We implement the S2V-DQN algorithm based on the code (written in C++) provided by the authors.[5] For the synthetic graphs generated from ER, BA, HK, and SW models, S2V-DQN is unstable to be trained on graphs of size $(100, 200)$ and $(400, 500)$ without pre-training. Hence, we perform fine-tuning as mentioned in the original paper (Khalil et al., 2017). For instance, when we train S2V-DQN on the ER-$(100, 200)$ datasets, we fine-tune the model trained on ER-$(50, 100)$. Next, for the ER-$(400, 500)$, we perform "curriculum learning"; we first train S2V-DQN on the ER-$(50, 100)$ dataset, then fine-tune on the ER-$(100, 200)$, ER-$(200, 300)$, ER-$(300, 400)$ and ER-$(400, 500)$ in sequence. Finally, for training S2V-DQN on graphs with size larger than 500, we were unable to train it on the raw graph under the available computational budget. Hence we train S2V-DQN on subgraphs sampled from the training graphs. To this end, we sample edges from the model uniformly at random without replacement, until the number of vertices reach 300. Then we used the subgraph induced from the sampled vertices.

TGS. We use the official implementation and models provided by the authors.[6] Unfortunately, the provided code runs out of memory for larger graphs since that they keep all of the intermediate solutions in the breadth-first search queue. For such cases, we modify the algorithm by discarding the oldest graph in the queue whenever the queue reaches its maximum size, i.e., ten times the number of required solutions for the problem.

CPLEX. We use the CPLEX (ILO, 2014) provided in the official homepage.[7] In order to optimize its performance under limited time, we set its emphasis parameter, i.e., `MIPEmphasisFeasibility`, to prefer higher objective over proof of optimality.

KaMIS. We use KaMIS (Hespe et al., 2019a) from its official hompage without modification.[8]

[5]https://github.com/Hanjun-Dai/graph_comb_opt
[6]https://github.com/intel-isl/NPHard
[7]https://www.ibm.com/products/ilog-cplex-optimization-studio
[8]http://algo2.iti.kit.edu/kamis/

A.3. Dataset Details

In this section, we provide additional details on the datasets used for the experiments.

Synthetic datasets. For the ER, BA, HK, and WS datasets, we train on graphs randomly generated on the fly and perform validation and evaluation on a fixed set of 1000 graphs.

SATLIB dataset. The SATLIB dataset is a popular benchmark for evaluating SAT algorithms. We specifically use the synthetic problem instances from the category of random 3-SAT instances with controlled backbone size (Singer et al., 2000). Next, we describe the procedure for reducing the SAT instances to MIS instances. To this end, a vertex is added to the graph for each literal of the SAT instance. Then edges are added for each pair of vertices satisfying the following conditions: (a) that are in the same clause or (b) they correspond to the same literals with different signs. Consequently, the MIS in the resulting graph corresponds to the truth assignment to the optimal assignments of the SAT problem (Dasgupta et al., 2008).

PPI dataset. The PPI dataset is the protein-protein-interaction dataset with vertices representing proteins and edges representing interactions between them.

REDDIT datasets. The REDDIT-B, REDDIT-M-5K, and REDDIT-M-12K datasets are constructed from online discussion threads in reddit[9] where vertices represent users and edges mean at least one of two users responded to the other user's comment.

Autonomous system dataset. The as-Caida dataset is a set of autonomous system graphs derived from a set of RouteViews BGP table snapshots (Leskovec et al., 2005).

Citation dataset. The Cora and the Citeseer are networks constructed by vertices and edges representing documentation and citation links between them, respectively (Sen et al., 2008).

Amazon dataset. The Computers and Photo graphs are segmented from the Amazon co-purchase graph (McAuley et al., 2015), where vertices correspond to goods and edges represent goods which are frequently purchased together.

Coauthor dataset. The CS and Physics graphs represent authors and the corresponding co-authorships by vertices and edges, respectively. It was collected from Microsoft Academic Graph from the KDD Cup 2016 challenge3.[10]

We also provide the statistics of the datasets used in experiments corresponding to Table 1 and 3 in Table 6 and 7, respectively.

Table 6. Number of nodes, edges and graphs for SATLIB, PPI, REDDIT, and as-Caida datasets used in Table 1. Number of graphs is expressed as a tuple of the numbers of training, validation and test graphs, respectively.

Dataset	Number of nodes	Number of edges	Number of graphs
SATLIB	(1209, 1347)	(4696, 6065)	(38 000, 1000, 1000)
PPI	(591, 3480)	(3854, 53 377)	(20, 2, 2)
REDDIT (BINARY)	(6, 3782)	(4, 4071)	(1600, 200, 200)
REDDIT (MULTI-5K)	(22, 3648)	(21, 4783)	(4001, 499, 499)
REDDIT (MULTI-12K)	(2, 3782)	(1, 5171)	(9545, 1192, 1192)
as-Caida	(8020, 26 475)	(36 406, 106 762)	(108, 12, 12)

Table 7. Number of nodes and edges for each dataset used in the Table 3.

Dataset	Number of nodes	Number of edges
Citeseer	3327	3668
Cora	2708	5069
Coauthor CS	18 333	81 894
Coauthor Physics	34 493	247 962
Amazon Computers	13 381	245 778
Amazon Photo	7487	119 043

[9] https://www.reddit.com/
[10] https://kddcup2016.azurewebsites.net/

B. Details of Other Combinatorial Optimizations

B.1. Maximum Weighted Indpendent Set Problem

First, we describe the maximum weighted independent set (MWIS) problem (Balas & Yu, 1986). Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ associated with positive weight function $w : \mathcal{V} \rightarrow \mathbb{R}^{+}$. The goal of the MWIS problem is to find the independent set $\mathcal{I} \subseteq \mathcal{V}$ where the total sum of weight $\sum_{i \in \mathcal{I}} w(i)$ is maximum. In order to apply the LwD framework to the MWIS problem, we include the weight of each vertex as its feature to the policy network and modify the reward function by the increase in weight of included vertices, i.e., $R(\boldsymbol{s}, \boldsymbol{s}') = \sum_{i \in \mathcal{V}_* \setminus \mathcal{V}_*'} s_i' w(i)$. We sample the weights of each vertices from a normal distribution with mean and standard deviation fixed to 1.0 and 0.1, respectively.

B.2. Prize Collecting Maximum Independent Set Problem

The prize collecting maximum independent set (PCMIS) problem is an instance of the generalized minimum vertex cover problem (Hassin & Levin, 2006). To describe this problem, one may consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a subset of vertices $\mathcal{I} \subseteq \mathcal{V}$. The PCMIS problem is associated with the following the “prize” function f to maximize:

$$f(\mathcal{I}) := |\mathcal{I}| - \lambda |\{\{i,j\} : i,j \in \mathcal{I}, i \neq j\}|,$$

where $\lambda > 0$ is the penalty function for including two adjacent vertices. We set $\lambda = 0.5$ in the experiments. Such a problem could be interpreted as relaxing the hard constraints on independent set to a penalty function in the MIS problem. Especially, one can examine that optimal solution of the PCMIS problem becomes the maximum independent set when $\lambda > 1$. For applying the LwD framework on the PCMIS problem, we remove the clean-up phase in the transition function of MDP and modify the reward function $R(\boldsymbol{s}, \boldsymbol{s}')$ as the increase in prize function at each iteration.

B.3. Maximum Cut Problem

Next, we introduce the maximum cut (MAXCUT) problem (Garey & Johnson, 1979). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, goal of the MAXCUT problem is on finding a subset of vertices $\mathcal{I}$ that maximize the following objective:

$$f(\mathcal{I}) := |\{i,j\} : i \in \mathcal{I}, j \in \mathcal{V} \setminus \mathcal{I}|,$$

which corresponds to the number of edges that form a “cut” between $\mathcal{I}$ and $\mathcal{V} \setminus \mathcal{I}$. To apply LwD, we remove the clean-up phase in the transition function of our MDP and set the reward function $R(\boldsymbol{s}, \boldsymbol{s}')$ as the increase in the objective function at each iteration.

B.4. Maximum-a-posteriori Inference Problem on the Ising Model

Finally, we describe the maximum-a-posteriori (MAP) inference problem on the anti-ferromagnetic Ising model (Onsager, 1944). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the probability distribution of the Ising model is described as $p(\boldsymbol{s}) := \frac{1}{Z} \exp\big(\phi(\boldsymbol{s})\big)$, where $\boldsymbol{s}$ is the random variable, Z is the normalization constant, and the function ϕ is the objective to maximize. To be specific, ϕ is defined as follows:

$$\phi(\boldsymbol{s}) : = \gamma \sum_{i \in \mathcal{V}} f_1(i) + \beta \sum_{\{i,j\} \in \mathcal{E}} f_2(i,j),$$

$$f_1(i) = \begin{cases} -1 & \text{if } s_i = 0 \\ 1 & \text{if } s_i = 1 \end{cases}, \qquad f_2(i,j) = \begin{cases} -1 & \text{if } s_i = s_j \\ 1 & \text{if } s_i \neq s_j \end{cases},$$

Here, β and γ are called interaction and magnetic field parameters, respectively. Furthermore, $\mathbf{1}_{\mathcal{A}}$ is an indicator function for set of vertices $\mathcal{A}$ and $\mathcal{A}_k = \{i : i \in \mathcal{V}, s_i = k\}$ for $k = 0, 1$. In order to solve the MAP inference problem on the Ising model, we remove the clean-up phase in the transition function as in the PCMIS problem and modify the reward function $R(\boldsymbol{s}, \boldsymbol{s}')$ as the increase in objective function at each iteration. In our experiments, we set $\beta = 1.0, \gamma = 1.0$ and report $\phi(\boldsymbol{s})$ as the objective to maximize.

C. Graphical Illustration of LwD

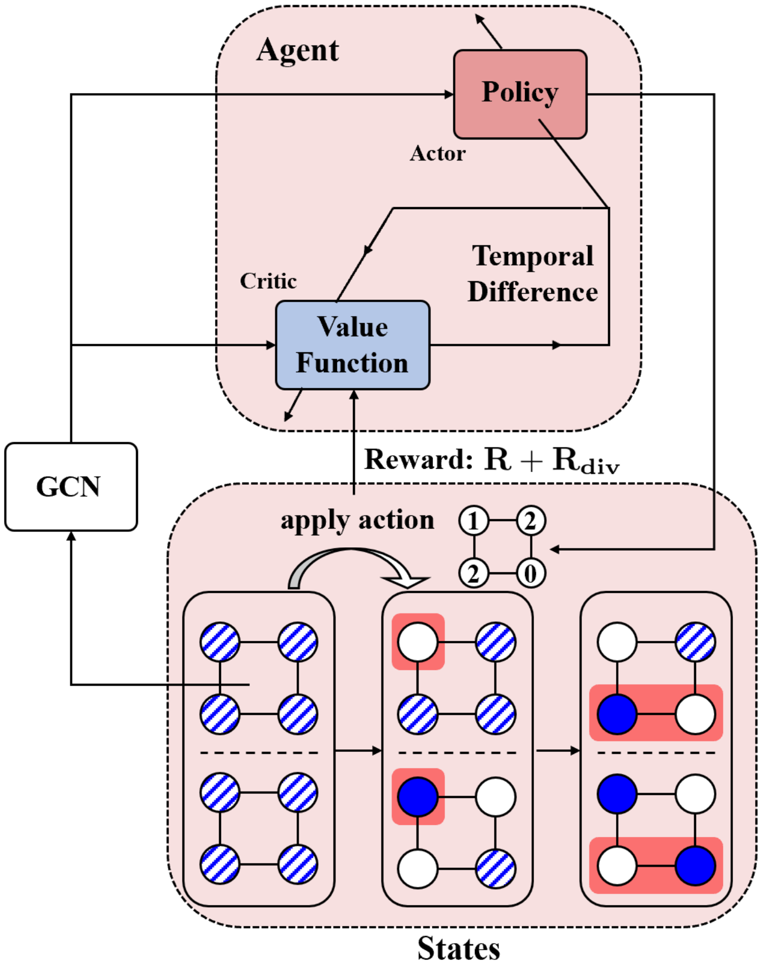

Figure 6. Illustration of the overall learning what to defer (LwD) framework.

References

Balas, E. and Yu, C. S. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068, 1986.

Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. V. *Algorithms*. McGraw-Hill Higher Education, 2008.

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

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*, 2019.

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

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.

Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 177–187. ACM, 2005.

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.

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

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

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.

Singer, J., Gent, I. P., and Smaill, A. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.