Towards Sample Efficient Agents through Algorithmic Alignment

Mingxuan Li *
Department of Computer Science
Brown University
mingxuan_li@brown.edu

Abstract

Deep reinforcement-learning agents have demonstrated great success on various tasks. However, current methods typically suffer from sample complexity problem when learning in high dimensional observation spaces, which limits the application of deep reinforcement-learning agents to complex, uncertain real world tasks. In this work, we propose and explore Deep Graph Value Network as a promising method to work around this drawback using a message-passing mechanism. The main idea is that the RL agent should be guided by structured non-neural-network algorithms like dynamic programming. According to recent advances in algorithmic alignment, neural networks with structured computation procedures can be trained efficiently. We demonstrate the potential of graph neural network in supporting sample efficient learning by showing that Deep Graph Value Network can outperform unstructured baselines by a large margin with low sample complexity.

1 Introduction

Deep reinforcement-learning algorithms have produced breakthroughs in recent years in various domains. They have defeated human experts and world champions in classic board games and card games [Li et al., 2020, Silver et al., 2016, Brown and Sandholm, 2017, Schaeffer et al., 1992], Atari video games [Badia et al., 2020, Schrittwieser et al., 2019] and even StarCraft II [Vinyals et al., 2019]. However, agents with powerful non-linear function approximators also require large amounts of experience to learn. And, while humans are no longer the best Go players, they can still generalize well and outperform a single agent on general environments with uncertainty. Thus, agents need the right problem formulation to effectively learn in complex environment and generalize their knowledge to new domains[Konidaris, 2019].

An object-oriented representation is a natural way to represent the environment and encapsulate essential dynamics into interactions among objects. Previously, under the standard Markov Decision Process (MDP) formalization, Diuk *et al.* [Diuk et al., 2008] proposed Object-Oriented MDPs, which enable efficient learning using predefined representations. Similarly, relational MDPs [Guestrin et al., 2003] provide a probabilistic relational framework for planning. The obstacle of applying these methods to complex environments is that a lot of human effort is required to craft a suitable representation before learning and planning can be carried out within these frameworks.

Fortunately, deep learning provides us tools to learn a suitable representation automatically. Note that the relational representations of environment dynamics can be viewed as a graph. Therefore, graph neural networks (GNNs) are a natural choice for incorporating deep learning into the relational reinforcement-learning framework [Van Otterlo, 2005]. GNNs have an intrinsic inductive bias towards

*A preliminary version is accepted by DLRLSS 2020 Poster Session

Project Report. Ongoing work.

relational data and object representations, which can improve the learning process [Battaglia et al., 2018].

There has been some preliminary work on applying GNNs to learn representations for agents [Vinyals et al., 2019, Bapst et al., 2019]. Yet, applying these approaches requires, not only a good input (representation), but also a proper solver (computation procedure). Traditional algorithms without learned parameters can trivially strongly generalize with guaranteed correctness, which our superhuman agents cannot achieve. Because current advances in designing neural network architectures do not give explicit guidance on how to structure the problem solving, performance has lagged. For example, neural networks are used as the critic in SAC [Dy and Krause, 2018]. The theoretical analysis shows promising performance of soft value iteration and soft policy improvement. In implementation, most of the benefits are gone as they directly use a neural network to estimate values diverging from how Bellman backup operator utilizes rewards and next state information.

The question we try to answer is how to get the best of both neural networks and non-neural network algorithms (e.g. Dynamic Programming) to learn effectively and generalize robustly?

One solution is to execute those algorithms with neural networks. Different datasets favor different network structures, which is the same situation as learning those non-neural network algorithms. Recent work has shown that better algorithmic alignment improves sample complexity and generalization [Xu et al., 2019]. Given the fact that GNNs align well with dynamic programming (DP) [Bellman, 1954], we presume value iteration, as a probabilistic version of Bellman-ford algorithm, should also be solvable with GNNs.

In this work, we propose Deep Graph Value Network (DeepGV), a message-passing framework, to robustly solve the given MDP. We empirically verify its effectiveness as a general MDP solver and its potential towards building general structured agents.

This project is ongoing. There are a wide range of possibilities in the near future. One immediate direction is extending DeepGV to learn purely from experiences to serve as a better critic or even actor module. Then, from a long-term view, intrinsically equipped with a rich relational representation and interface of hierarchical structures, the message-passing framework is well suited to learning complex environment dynamics from a structured, object-oriented perspective. We may try to incorporate DeepGV with ideas from algorithms that work well in tabular cases with guaranteed performance like feudal framework [Dayan and Hinton, 2000], object-oriented MDP learning [Diuk et al., 2008] and options [Sutton et al., 1999].

2 Preliminaries

In this section, we briefly go through some basics of Graph Neural Networks and the concept of algorithmic alignment.

2.1 Graph Neural Networks (GNNs)

The original GNNs definition was proposed by Scarselli et al. [2008] for processing the data represented in graphs. The goal of GNNs is to learn a node embedding $h_v \in \mathbb{R}^n, v \in G(V, E)$ or an aggregation h of the graph knowledge. As an example, the computation process of a GNN layer is [Battaglia et al., 2018]:

$$h_v^{(k+1)} = \sum_{v \in V} f_\theta(h_v^{(k)}, h_{\mathcal{N}(v)}^{(k)}) \tag{1}$$

$$h = \sum_{v \in V} g_\omega(h_v^{(K)}), \tag{2}$$

where f_θ is a parametric transition function gathering information from node v's neighbours and edges, g_ω is the output function that aggregates all the information in the graph and $h_{\mathcal{N}(v)}$ is a set of neighbours of node v. In each iteration, the graph is updated using the results from last iteration $k, k \in \mathbb{N}$. The final output depends on the result after K iterations when the graph representation converges. In the following discussions, we will use 'state' and 'node' interchangeably for simplicity of explanation.

2.2 Algorithmic Alignment

We adopt the theoretical framework of [Xu et al., 2019]. Our next step is to extend the following results to the reinforcement learning setting via KWIK bounds [Li et al., 2008].

Definition 2.1. (PAC learning and sample complexity). Fix an error parameter $\epsilon > 0$ and failure probability $\delta \in (0, 1)$. Suppose $\{x_i, y_i\}_{i=1}^{M}$ are i.i.d. samples from distribution $\mathcal{D}$, and the data satisfies $y_i = g(x_i)$ for some underlying function g. Let $f = \mathcal{A}\left(\{x_i, y_i\}_{i=1}^{M}\right)$ be the function generated by a learning algorithm $\mathcal{A}$. Then, g is (M, ϵ, δ)-*learnable* with $\mathcal{A}$ if

$$\mathbb{P}_{x\in\mathcal{D}}[||f(x) - g(x)|| \leq \epsilon] \geq 1 - \delta. \tag{3}$$

The *sample complexity* $\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta)$ is

$$\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta) = \min\{M|\mathbb{P}_{x\in\mathcal{D}}[||f(x) - g(x)|| \leq \epsilon] \geq 1 - \delta\}. \tag{4}$$

Definition 2.2. (Algorithmic alignment). Let g be a reasoning function and $\mathcal{F}$ a neural network with n modules f_i. The module functions $g_1, ..., g_n$ generate g for $\mathcal{N}$ if, by replacing f_i with g_i, the network simulates g. Then, $\mathcal{N}(M, \epsilon, \delta)$-algorithmically aligns with g if (1) $g_1, ..., g_n$ generate g and (2) there are learning algorithms $\mathcal{A}_i$ for the f_i such that $n \cdot \max_i \mathcal{C}_{\mathcal{A}_i}(g_i, \epsilon, \delta) \leq M$.

From the definition, it is clear that only if each module of the underlying function aligns well with neural network modules can we reduce M to its minimum.

Theorem 2.3. (Algorithmic alignment improves sample complexity). Fix ϵ and δ. Suppose $S_i, y_i{}_{i=1}^{M} \in \mathcal{D}$, where $|S_i| < N$, and $y_i = g(S_i)$. Suppose $f_1, ..., f_n$ are network $\mathcal{F}$'s MLP modules in sequential order. Suppose $\mathcal{N}$ and $g(M, \epsilon, \delta)$-algorithmically align via functions $g_1, ..., g_n$. Under the following assumptions, g is $(M, O(\epsilon), O(\delta))$-learnable by $\mathcal{N}$:

- **Algorithm stability.** Let $\mathcal{A}$ be the learning algorithm for $\mathcal{N}_i$. Suppose $f = \mathcal{A}(\{x_i, y_i\}_{i=1}^{M})$, and $\hat{f} = \mathcal{A}(\{\hat{x_i}, y_i\}_{i=1}^{M})$. For any x, $||f(x) - \hat{f}(x)|| \leq L_0 \cdot \max_i ||x_i - \hat{x_i}||$, for some L_0.
- **Sequential learning.** $\mathcal{F}$ is trained sequentially: f_1 has inputs $\{x_i^{(1)}, g_1(x_i^{(1)})\}_{i=1}^{N}$, with $x_i^{(1)}$ obtained from S_i. For $j > 1$, the input $x_i^{(j)}$ for f_j are the outputs from f_{j-1} while labels are generated by the ground truth functions $g_{j-1}, ..., g_1$ on $x_i^{(1)}$.
- **Lipschitzness.** The learned functions f_i satisfy $||f_i(x) - f_i(\hat{x})|| \leq L_1 \cdot ||x - \hat{x}||$, for some L_1.

2.3 Alignment Between Value Iteration and GNNs

In each iteration of message distribution and aggregation, every node first updates its embeddings according to information from its neighbours and edges between them. After some iteration of execution, we aggregate all the information and extract answers from it with another network. As shown in Eq. 5, comparing general GNN updates with value iteration, they share quite similar underlying computational steps, which can be simulated by neural networks easily.

$$h_v^{(k+1)} = f_\psi\left(\sum_{v\in V} f_\theta(h_v^{(k)}, h_{\mathcal{N}(v)}^{(k)})\right) \tag{5}$$

$$V^{(k+1)}(s) = \max_a \sum_{s'\in\mathcal{S}} p(s'|a, s)\left[R(s, a, s') + \gamma V^{(k)}(s')\right] \tag{6}$$

The gap between value iteration and GNNs updates is in how we inject the knowledge of transition probabilities. In an end-to-end learning setting, one way is to use an attention mechanism in f_θ and not explicitly provide the transition probability as supervision. Instead, we provide ground truth state-value information as labels. Thus, the local aggregation function f_ψ only needs to learn a max over neighbourhoods to approximate value iteration. Intuitively, this computation structure should outperform vanilla MLPs by a large margin.

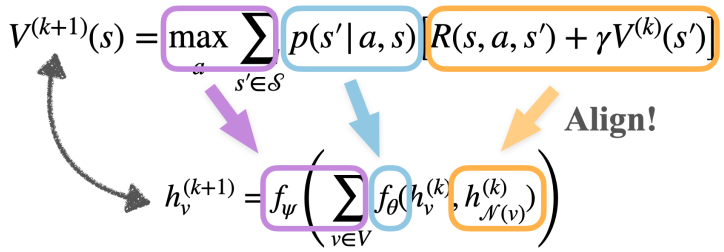
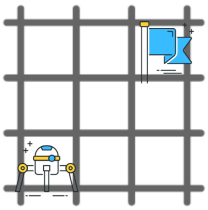
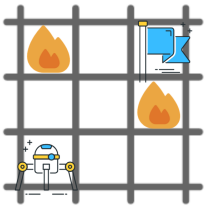


Figure 1: An example of how generally GNNs align with value iteration. Both GNNs and VI share the same loop structure, which does not need to be learned. Furthermore, the inner computation procedures are also alike. Thus, it would be relatively simpler for GNNs than generic MLPs to learn to execute value iteration.



(a) Plain grid world.



(b) Grid world with traps.

Figure 2: Two types of grid world environment.

3 Deep Graph Value Network

In this section, we describe the framework of Deep Graph Value Network. The network takes a complete graph $G(V, E)$ as input with different attributes assigned to nodes and edges. Each node represents a state in the given MDP $\langle S, A, R, \gamma, T \rangle$. Edges between nodes represent actions with rewards binding to them.

In iteration $k+1$, the node embedding h_i is updated as follows,

$$h_i^{(k+1)} = f_\psi \left[\underset{a \in \mathcal{A}}{\text{CAT}} \left(\sum_{j \in \mathcal{N}(i)} A_{i,j}^{(k+1)} \left(r(i,a,j) + \gamma h_j^{(k)} \right) \right) \right], \tag{7}$$

where r is the embedded reward attributes attached to edges, γ is the discount factor and $A_{i,j}$ is the transition probability learned by a self-attention mechanism. The part inside the square brackets calculates the Q-value for state i and the local aggregate function f_γ learns to execute $\max$. The attention mechanism is learned as follows,

$$A_{i,j}^{(k+1)} = \frac{exp(e_{i,j}^{(k+1)})}{\sum_{l \in \mathcal{N}(i)} exp(e_{i,l}^{(k+1)})}, \tag{8}$$

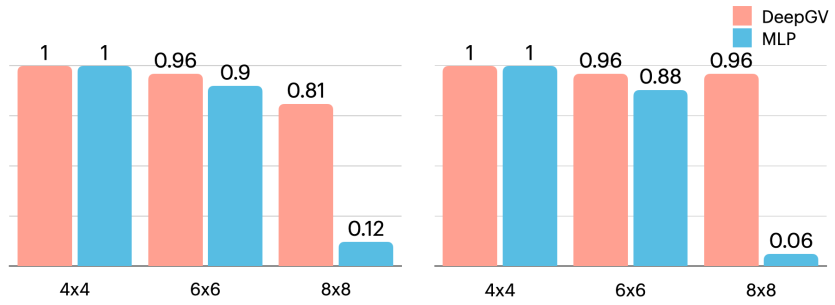
$$a_{i,j}^{(k+1)} = f_\theta(h_i^{(k)}, h_i^{(k)}). \tag{9}$$

After K iterations, we aggregate information from all the nodes and pass them through an MLP to get value-ranking predictions:

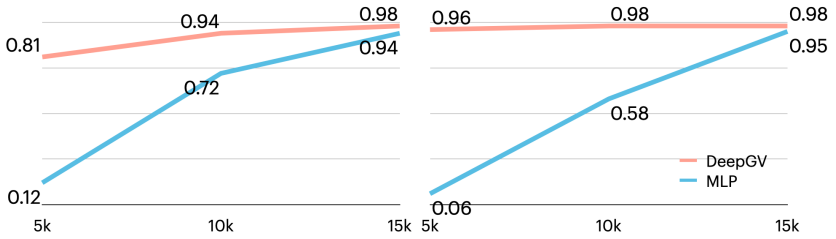
$$\hat{y} = f_\omega \left(\underset{i \in V}{\text{CAT}} h_i^{(K)} \right). \tag{10}$$

4 Experiments and Results

In this section, we apply our framework to solve MDPs and compare its performance with that of vanilla MLPs. From the analysis in previous sections, unlike DeepGV, MLP doesn't align well with



(a) Value ranking accuracy varying grid world size.



(b) Training set size vs. test accuracy.

Figure 3: **Test accuracy and sample complexity on two types of grid world.** In each sub-figure, the LHS result comes from plain grid world while the RHS result comes from grid world with traps. (a) MLP fails on 8×8 world while DeepGV performs well. (b) As the size of training set increases, both DeepGV and MLP can converge to reasonable performance at test time. But, DeepGV achieves good performance with fewer training examples.

dynamic programming, suggesting poor performance for solving MDPs. Thus, we want to verify that DeepGV can easily be trained and outperforms the MLP by a large margin.

4.1 Experiment Setup

We evaluate the networks on two different types of MDPs. The first one is a slippery $N \times N$ grid world without obstacles and fire states. The second type has obstacles and fire states. We generate different instances of these two kinds of MDPs and evaluate two algorithms on each. See fig. 2 for a visualization. To solve a given MDP, the input is randomly initialized state values, state coordinates and the queried state coordinates. The expected output should be the value ranking of the queried state. The corresponding label is a one-hot vector of length N^2 indicating the correct ranking.

4.2 Results and Discussion

Generally speaking, DeepGV can predict value rankings much easier. In fig. 3, when we fix the training set size and increase the task difficulty (the size of grid world), DeepGV can still be trained to a satisfying level while MLP performance drops severely when it faces the 8×8 grid world. To assess the algorithm's sample complexity, we varied the training set size and fixed the size of the grid world at 8×8. As shown in the figure, there is a clear gap between MLP and DeepGV performance. This finding empirically supports Theorem 2.2—better algorithmic alignment induces better sample complexity. Overall, our experiment answers the question we raise in the beginning of this section, showing that DeepGV trains well and outperforms the MLP by a large margin.

5 Conclusion

We presented Deep Graph Value Network (DeepGV), a message-passing-mechanism-based network structure that can efficiently learn to solve MDPs by executing a similar computational procedure to that of value iteration. By reviewing some recent theoretical advances in algorithmic alignment, we

illustrate the importance of utilizing neural networks as structured learning agents instead of merely feature extractors in deep reinforcement learning. Our preliminary experimental results show the performance gap between structured networks and unstructured networks in solving relational tasks suggesting the promising potential of message-passing mechanisms. Our results provide an exciting new avenue for future work on designing high-efficiency agents incorporating GNN-like structures from not only feature extraction but also a structured problem-solver perspective.

6 Acknowledgements

I would like to thank my advisor Michael Littman for insightful discussion that motivated me to think more thoroughly and deeper; Evan Carter for offering an intuitive explanation of concepts and sharing his meeting slot.

References

A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. *CoRR*, abs/2003.13350, 2020. URL `https://arxiv.org/abs/2003.13350`.

V. Bapst, A. Sanchez-Gonzalez, C. Doersch, K. L. Stachenfeld, P. Kohli, P. W. Battaglia, and J. B. Hamrick. Structured agents for physical construction. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 464–474, 2019. URL `http://proceedings.mlr.press/v97/bapst19a.html`.

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

R. Bellman. Some applications of the theory of dynamic programming - A review. *Operations Research*, 2(3):275–288, 1954. doi: 10.1287/opre.2.3.275. URL `https://doi.org/10.1287/opre.2.3.275`.

N. Brown and T. Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359:eaao1733, 12 2017. doi: 10.1126/science.aao1733.

P. Dayan and G. Hinton. Feudal reinforcement learning. *Advances in Neural Information Processing Systems*, 5, 09 2000. doi: 10.1002/0471214426.pas0303.

C. Diuk, A. Cohen, and M. L. Littman. An object-oriented representation for efficient reinforcement learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 240–247, 2008. doi: 10.1145/1390156.1390187. URL `https://doi.org/10.1145/1390156.1390187`.

J. G. Dy and A. Krause, editors. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 2018. PMLR. URL `http://proceedings.mlr.press/v80/`.

C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational mdps. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 1003–1010, 2003. URL `http://ijcai.org/Proceedings/03/Papers/144.pdf`.

G. Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7, 2019.

J. Li, S. Koyamada, Q. Ye, G. Liu, C. Wang, R. Yang, L. Zhao, T. Qin, T. Liu, and H. Hon. Suphx: Mastering mahjong with deep reinforcement learning. *CoRR*, abs/2003.13590, 2020. URL `https://arxiv.org/abs/2003.13590`.

L. Li, M. L. Littman, and T. J. Walsh. Knows what it knows: a framework for self-aware learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 568–575, 2008. doi: 10.1145/1390156.1390228. URL https://doi.org/10.1145/1390156.1390228.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

J. Schaeffer, J. C. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artif. Intell.*, 53(2-3):273–289, 1992. doi: 10.1016/0004-3702(92)90074-8. URL https://doi.org/10.1016/0004-3702(92)90074-8.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265, 2019. URL http://arxiv.org/abs/1911.08265.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

R. S. Sutton, D. Precup, and S. P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

M. Van Otterlo. A survey of reinforcement learning in relational domains. *Centre for Telematics and Information Technology (CTIT) University of Twente, Tech. Rep*, 2005.

O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019. doi: 10.1038/s41586-019-1724-z.

K. Xu, J. Li, M. Zhang, S. S. Du, K. Kawarabayashi, and S. Jegelka. What can neural networks reason about? *CoRR*, abs/1905.13211, 2019. URL http://arxiv.org/abs/1905.13211.