Multi-objective Reinforcement Learning in Factored MDPs with Graph Neural Networks

Extended Abstract

Marc Vincent
Thales Land and Air Systems
Paris, France
LIP6, Sorbonne Université, CNRS
Paris, France
marc.vincent@thalesgroup.com

Amal El Fallah Seghrouchni
LIP6, Sorbonne Université, CNRS
Paris, France
Ai Movement - International Artificial Intelligence Center of Morocco,
Mohammed VI Polytechnic University
Rabat, Morocco
amal.elfallah@lip6.fr

Vincent Corruble
LIP6, Sorbonne Université, CNRS
Paris, France
vincent.corruble@lip6.fr

Narayan Bernardin
Thales Land and Air Systems
Paris, France
narayan.bernardin@thalesgroup.com

Rami Kassab
Thales Land and Air Systems
Paris, France
rami.kassab@thalesgroup.com

Frédéric Barbaresco
Thales Land and Air Systems
Paris, France
frederic.barbaresco@thalesgroup.com

ABSTRACT

Many potential applications of reinforcement learning involve complex, structured environments. Some of these problems can be analyzed as factored MDPs, where the dynamics are decomposed into locally independent state transitions and the reward is rewritten as the sum of local rewards. However, in some scenarios, these rewards may represent conflicting objectives, so that the problem is better interpreted as a multi-objective one, with a weight associated to each reward. To deal with such *multi-objective factored MDPs*, we propose a method which combines the use of graph neural networks, to process structured representations, and vector-valued Q-learning. We show that our approach empirically outperforms methods that directly learn from the scalarized reward and demonstrate its ability to generalize to different weights and number of entities.

KEYWORDS

Multi-objective; Reinforcement Learning; Factored MDP; Relational Learning; Graph Neural Networks; Deep Learning

ACM Reference Format:
Marc Vincent, Amal El Fallah Seghrouchni, Vincent Corruble, Narayan Bernardin, Rami Kassab, and Frédéric Barbaresco. 2023. Multi-objective Reinforcement Learning in Factored MDPs with Graph Neural Networks: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

Most research in reinforcement learning (RL) focuses on end-to-end learning, where the agent starts out with no prior on the task. However, for complex problems, we often have information about the structure of the environment at our disposal. This information can be used to inject priors into the learning process. Before the advent of deep RL, researchers tackled large Markov decision processes (MDP) by decomposing the states and rewards into locally conditioned elements to obtain factored MDPs (FMDP) [3]. Even though in recent times function approximation by deep neural networks has alleviated these issues, accounting for the structure of the state space still helps solving complex environments, for example using relational learning via graph neural networks (GNN) [2, 11, 12].

In FMDPs, rewards typically have an additive structure: they represent the sum of locally-scoped rewards. However, some tasks are better described as striking a compromise between several possibly contradicting objectives. Multi-objective RL (MORL) treats such problems where multiple reward functions are used, each associated with a different objective [9]. The overall goal is given by a utility function that depends on these rewards and on their associated weights, which reflect their level of priority. This setting allows easier definitions of the desired compromises between the competing objectives; it also makes it possible to train an agent that adapts to utility functions that may change over time, e.g. when using a linear scalarization whose weights are not constant. Many promising applications of RL benefit from a multi-objective specification, like self-driving cars [5].

In this paper, we study the case where the problem combines a structured environment with a multi-objective specification. Although significant work has been carried out for FMDPs and MORL, to our knowledge their intersection has not received scrutiny yet. To do so, we propose a multi-objective version of DQN [6] that can be applied to *multi-objective factored MDPs* by making use of recent advances in graph neural networks proposed in [4]. We compare our method, factored multi-objective DQN (FMODQN), to the single-objective algorithm from [4] on two novel benchmarks and show that it outperforms this baseline, can generalize to different reward weight vectors and number of entities, and is able to deal with hundreds of objectives.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 ALGORITHM

In FMDPs [3, 8], a state $s \in \mathcal{S}$ is reinterpreted as a set of n state variables $(s_1, s_2, ..., s_n)$. From there, for a given scope $Z \subseteq \{1, ..., n\}$, we can define a scoped variable $s[Z] = (s_i)_{i \in Z}$ as a subset of the state variables. Given n scopes $Z_1, ..., Z_n$, the transition probability function can then be factored into n locally-scoped transition functions in the form $P(s' \mid s, a) = \prod_{i=1}^{n} P_i(s'[i] \mid s[Z_i], a)$. The transition dependencies between state variables can be represented with a directed graph where each node stands for a state variable and an edge going from node i to node j indicates dependency of s_j on s_i, i.e. $i \in Z_j$. Similarly, the reward function can be factored over m scopes $Z_1^r, ..., Z_m^r$, for example as a sum $R(s, a) = \sum_{i=1}^{m} R_i(s[Z_i^r], a)$.

However, in many cases each local reward can be interpreted as a different objective. To account for this, we modify the FMDP formalism by altering the agent's objective to maximize the scalarization of expected returns over a vector of locally-scoped rewards $\mathbf{R}(s_t, a_t) = (R(s_t[Z_i^r], a_t))_{1 \leq i \leq m}$. We restrict ourselves to linear scalarizations with weights w_i; therefore, we express the agent's objective as $\max_{\pi \in \Pi} \sum_{i=1}^{m} w_i \, \mathbb{E}_{\pi,T}[\sum_{t=0}^{\infty} \gamma^t R(S_t[Z_i^r], A_t)]$. The general approach in multi-objective value-based methods is to train a Q-value estimator that returns a matrix $\mathbf{Q}(s, \cdot, \mathbf{w}) \in \mathcal{R}^{|\mathcal{A}| \times m}$ from which we can then retrieve the scalarized action-state value $Q(s, a, \mathbf{w}) = \mathbf{Q}(s, a, \mathbf{w})^\top \cdot \mathbf{w}$. The policy is derived by choosing the action that maximizes this scalarized Q-value [1, 10].

In contrast to other MORL methods, our focus is on combinatorial generalization, that is, on environments where the number of state variables, actions, and local rewards may vary between episodes: as a consequence, we only consider cases where the local transitions and rewards are defined the same for all scopes, with one action and one reward per state variable. The method most closely related to ours in the literature is SR-DRL [4], a single-objective algorithm which combines a message-passing GNN and an auto-regressive policy into a neural network trained with A2C [7] that is applicable to FMDPs with variable state and action spaces. We adapt the GNN architecture from [4] so that each input node represents one state variable and the weight of its associated reward, and each updated output node represents the Q-value associated with this reward. Using this scheme, we can condition the network on the current reward weights and stay invariant to the number of state variables and rewards/weights. Since there is one action per state variable, we augment each input node with an action embedding with value 1 if the action corresponding to this state variable is chosen, 0 otherwise. For a given action a, this augmented graph G_a is passed through the GNN to obtain $\mathbf{Q}(s, a, \mathbf{w})$. This operation is applied over n augmented graphs, one for each action, to compute $\mathbf{Q}(s, \cdot, \mathbf{w})$. The processing can be parallelized by batching the augmented graphs. We call this version of our algorithm Batch-FMODQN.

Unfortunately Batch-FMODQN's time complexity is quadratic in the number of nodes. However, this can be improved depending on the graph topology. In particular, in some environments the state variables may be entirely independent from each other (i.e. there are no edges in the graph). In this case, assuming we have one action a_i and one reward r_i per state variable $s[i]$, we can process each entity separately by applying a position-wise MLP to the list of state variables. This MLP has two outputs for each state variable $s[i]$: the action-state value relative to reward r_i that corresponds to choosing the associated action $Q_i(s[i], a_i, w_i)$, and the same action-state value corresponding to any other action $Q_i(s[i], A \neq a_i, w_i)$. We can then recover the scalarized Q-value of each action via the weighted sum of the local Q-values: $Q(s, a_i, \mathbf{w}) = w_i Q_i(s[i], a_i, w_i) + \sum_{j \neq i} w_j Q_j(s[j], A \neq a_i, w_j)$. This way we do not have to directly compute the whole Q-matrix since it would be mostly redundant; instead, the time complexity of the algorithm is $O(n)$. We call this variant of our algorithm Split-FMODQN.

N	Baseline	SR-DRL	Batch-FMODQN
10	49.71 ± 0.61	54.33 ± 0.65	**58.64 ± 0.58**
20	33.55 ± 0.37	38.95 ± 0.41	**42.82 ± 0.40**
40	25.12 ± 0.21	30.04 ± 0.22	**33.55 ± 0.25**
80	20.58 ± 0.13	23.65 ± 0.13	**27.02 ± 0.41**
160	18.42 ± 0.08	19.94 ± 0.08	**22.14 ± 0.27**

(a) SysAdmin

N	Baseline	SR-DRL	Split-FMODQN
10	-4.85 ± 0.04	-3.52 ± 0.03	**-0.43 ± 0.01**
20	-6.64 ± 0.03	-6.32 ± 0.03	**-3.23 ± 0.02**
40	-7.76 ± 0.02	-7.68 ± 0.02	**-5.58 ± 0.02**
80	-8.37 ± 0.01	-8.37 ± 0.01	**-7.17 ± 0.01**
160	-8.71 ± 0.01	-8.72 ± 0.01	**-8.07 ± 0.01**

(b) Spinning Plates

Table 1: Results of evaluation over 1000 runs in the dynamic weights setting for different problem sizes N. Score is the average cumulative reward with 95% confidence interval.

3 EXPERIMENTS

To validate our framework, we test it on two environments: we apply Batch-FMODQN to a multi-objective version of SysAdmin [3], and Split-FMODQN to Spinning Plates, a similar custom environment where all state variables are independent. We compare our approach to SR-DRL, using the same set of hyperparameters as in [4]; our only modification to their algorithm is the presence of the reward weights in the node attributes, so that the network can condition on them. Batch-FMODQN uses the same GNN architecture as SR-DRL while the network for Split-FMODQN consists of a single MLP with 5 hidden layers of size 32.. The baseline policy for SysAdmin selects the offline computer with highest reward weight to reboot; the baseline for Spinning Plates is a random policy. We test our approach in the dynamic weights setting: a new reward weight vector is generated at the beginning of each episode, both in training and evaluation. The agents are trained with problems of size $N = 10$ and evaluated on other problem sizes. Our results, presented in table 1, demonstrate our algorithms' improved ability to adapt to different reward weights and different problem sizes compared to SR-DRL on these tasks. We also trained our algorithms with fixed weights and $N = 10/20/40$, evaluated on problems of the same size N, and found little difference in performance compared to variable weights and size; this highlights FMODQN's generalization capabilities. Overall, our experiments confirm the efficiency of our approach on this specific type of problems.

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, Long Beach, California, USA, 11–20.

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv* (Oct. 2018). arXiv:1806.01261 [cs, stat]

[3] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. 2003. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research* 19 (Oct. 2003), 399–468. https://doi.org/10.1613/jair.1000

[4] Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. 2021. Symbolic Relational Deep Reinforcement Learning Based on Graph Neural Networks. In *International Conference on Machine Learning (Reinforcement Learning for Real Life (RL4RealLife) Workshop)*. https://doi.org/10.48550/arXiv.2009.12462

[5] Changjian Li and Krzysztof Czarnecki. 2019. Urban Driving with Multi-Objective Deep Reinforcement Learning. *International Conference on Autonomous Agents and Multiagent Systems* (Feb. 2019). https://doi.org/10.48550/arXiv.1811.08586

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236

[7] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *International Conference on Machine Learning* 48 (June 2016), 1928–1937. arXiv:1602.01783

[8] Ian Osband and Benjamin Van Roy. 2014. Near-Optimal Reinforcement Learning in Factored MDPs. In *Advances in Neural Information Processing Systems*. https://doi.org/10.48550/arXiv.1403.3741

[9] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research* 48 (Oct. 2013), 67–113. https://doi.org/10.1613/jair.3987

[10] Tomasz Tajmajer. 2017. Modular Multi-Objective Deep Reinforcement Learning with Decision Values. *Federated Conference on Computer Science and Information Systems* (Feb. 2017). arXiv:1704.06676 [cs]

[11] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* 575, 7782 (Nov. 2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[12] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. 2018. Relational Deep Reinforcement Learning. *arXiv* (June 2018). arXiv:1806.01830 [cs, stat]