POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Dynamic state and action factorization for distributed reinforcement learning

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Andrea Fondacaro

Advisor: Prof. Marcello Restelli

Co-advisors: Gianvito Losapio Marco Mussi

Academic year: 2024-2025

1. Introduction

Over the last few years, advancements in technology and AI allowed to shift towards automated decision making across many domains. In the case of large scale network infrastructures management, many decision are still taken by humans, but with the growing complexity and evolution of the modern infrastructures this is becoming more and more difficult. In safety-critical operations, like railways or air traffic control, even small mistakes could lead to serious consequences, so it would be better to shift towards automated methods that mitigate the risks of human errors. These systems are so complex, with many variables changing rapidly and small actions that could affect large parts of the system in cascade, that even using newer technologies they are not easy to manage at all.
Reinforcement Learning (RL) is the main technology used in these settings, thanks to its fast decision making capabilities. RL methods learn by interacting with the environment and tying different actions, looking at the effects on the system and understanding which actions lead to the best results in the future. The issue with RL is scaling to big infrastructures: the larger the network is the larger the state and action spaces get (curse of dimensionality) making it harder to learn and reach convergence in a feasible time.
A common approach in these cases is to use Multi Agent Reinforcement Learning (MARL): instead of managing the system with a single centralized controller, several independent agents operate on smaller parts of the system. This helps with learning times, but creates convergence problems, specifically if the system is partitioned in the wrong way. [3] In our setting these agents are cooperative, so they work together sharing a common global reward.
The objective of this thesis is to introduce a new, domain agnostic, clustering method based on Mutual Information (MI) to partition the systems in the best possible way. MI measures statistical dependence between the components: elements that strongly influence each other make sense to be put in the same cluster, while elements that are not strongly related to each other can be managed by different agents without any issues.
Our approach was validated in two different scenarios: a smaller scale power grid environment developed by us, and a larger railway routing problem generated using Flatland simulator. In the power grid case we analyze how our cluster-

ing method performs and if it actually improves training speed while maintaining a good convergence, while on the railway scenario we show that the approach is domain agnostic and can be applied on completely different data.
The project is carried out in the context of AI4REALNET project, founded by the European Union's Europe programme. It promotes AI solutions for complex critical infrastructures management. Our contribution focuses on improving the scalability and reliability of the RL approaches with a data driven network decomposition strategy.

2. Problem formulation

The environments considered in this work are networked infrastructures that could be represented as graphs: nodes are physical components, and edges represents their connections. We formalized our problems as a single Markov Decision Process (MDP), that is factorized by our algorithm into smaller sub-networks to perform learning on reduced state/action spaces. This decomposition brings two main issues: it breaks the Markov property because of the local observations, so the optimality of the solution is no more granted, and from an individual agent's perspective the environment can appear non-stationary since its dynamics are influenced by other agents.
To test and evaluate our approach we defined MDPs of two different case studies: a power grid control task and a large scale railway routing problem.

2.1. Power Grid problem

We created a simplified, scalable, power balancing environment where the network is a chain of N generators connected by $N+1$ lines. Each generator splits its output between the two adjacent lines, which load depends on the decisions of the two adjacent generators. The load of the generators is a random walk and changes at each time step, the observed states are the load on the lines and the state of the generators. The actions can modify the split of a single generator between the adjacent lines. The reward is defined as the negative sum of local imbalance cost, encouraging balanced loads across adjacent lines. We implemented two different versions of the same problem, the first one having discrete state and actions spaces, while the second one presents a continuous split and generator loads. To assess scalability, we conducted experiments on different versions of the network with 2,4 and 8 generators.

2.2. Flatland problem

Flatland [2] is a railway simulator on a two dimensional grid, where trains move in a discrete time taking routing decisions based on the environment. It can be seen as a centralized MDP, but it is typically treated as a multi-agent problem.
In our problem formulation we adopted SwitchFL, a custom extension of Flatland that shifts control from the single trains to the switches, that take decisions every time a train approaches. Each switch receives a local observation of the status of nearby stations. This formulations aligns way better with our approach, since the fixed position of the nodes allows to group them into clusters, while moving trains would have different MI based on the position at that specific time.

3. Algorithm Proposal

To decompose the original MDP into approximately independent subproblems, we introduce a data driven clustering method based on Mutual Information. Starting from a transition dataset (S_t, A_t, S_{t+1}), we build an input vector

$$Z_t = (S_t, A_t) \in R^{n_z}. \tag{1}$$

and compute a transition matrix whose values measure how much each state/action at time t influences each state at time $t+1$:

$$M_{ij} \;=\; I\Big(Z_t^{(i)};\, S_{t+1}^{(j)}\Big)\,. \tag{2}$$

In the fully discrete setting, the values are strongly affected by cardinalities: discrete actions with few values often have really low MI values even when they are relevant. To make these scores comparable between variables, we normalize the matrix using the relation

$$I(X;Y) \le \min\{H(X), H(Y)\}. \tag{3}$$

dividing each value for its upper value based on entropy. In the continuous version this normalization can't be applied, since entropy values are

not directly available. To compute MI we use the Kraskov-Stögbauer-Grassberger (KSG) estimator. [1]
The clustering itself follows a simple principle: each agent should control a subset of actions and observe a subset of the state variables that are significative to decide which of those actions it should take. We initialize one cluster per action and include in it all the components that present MI values higher than the given threshold.
There could be actions that influence the same components, so the clusters created previously are not guaranteed to be independent from each other. We then apply a greedy merging phase: at each step we consider pairs of clusters and evaluate if they are coupled enough to be merged. This decision is based on a score that considers two factors: it penalizes strong dependencies that would be left over if the two clusters remain separated and the weakly related variables that would be grouped together if the clusters are merged. This evaluation leverages a fixed parameter *alpha* that could be changed to penalize more one of the two options.
After this merging part, we deal with the leftovers variables that are not highly related to any action, so they have not been considered in the first initialization. Since they can still be highly coupled with other states and provide useful information to be observed, we evaluate if it is beneficial to include them to one or more of the existing clusters, adding it if so.
It is also important to say that we allow states to be placed into different clusters: this creates a big problem since we are practically giving up local Markov property, so we are not sure to reach the optimal solution anymore. This is a big tradeoff, but enforcing merges between clusters that had cross dependencies led to very few huge clusters and did not bring any advantage, making it very hard for larger scale problems to converge and so giving up one of the main objectives of our method.

3.1. Computational complexity

Let N be the number of transitions in the dataset, n_s the state dimension, n_a the action dimension, and $n_z = n_s + n_a$ the dimension of the joint input vector $Z_t = (S_t, A_t)$. Constructing the MI matrix requires estimating $n_z\, n_s$ mutual information terms, one for each pair $\left(Z_t^{(i)}, S_{t+1}^{(j)}\right)$.
In the discrete version of the algorithm, each MI value can be computed in $O(N)$, so the total cost of the matrix is $O(n_z\, n_s\, N)$.
In the continuous setting the matrix is estimated using KSG, which requires a k-nearest-neighbor search and therefore a higher complexity. With a small k, like in our case, the complexity is due to the neighbor search. In the best case, using efficient indexing, the complexity is $O(n_z\, n_s\, N \log N)$, while in the worst case is $O(n_z\, n_s\, N^2)$.
After the MI computation, the complexity of the next part is highly dependent on the number of actions in the system. To every action corresponds a cluster that has to be compared to every other cluster, so given a number of actions equal to n_a the merge evaluation complexity will be $O(n_a^2)$.
This means that in a case with small action set the MI estimation will take more time, while in rare cases with a huge action space (like the railway one we used to test) the cluster merging part is going to be the bottleneck.

4. Experimental results

We tested our algorithm on both the power grid environment and Flatland, comparing a centralized PPO baseline [4] with our approach. We report both the quality of the factorization and, in the power grid application, also its impact on training time and stability. For compactness, we will include only the most significative graphs.

4.1. Discrete Power Grid results

We visualize the clustering as a graph where square nodes represent states and circular nodes represent actions, while colored boundaries denote the clusters returned by our algorithm.

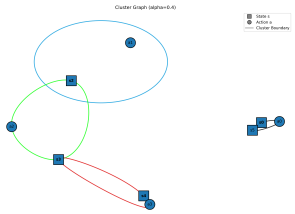

Figure 1: Example of clustering graph for the discrete Power Grid problem with 4 generators.

To assess the effect of factorization, we then compare a single agent PPO trained on the full

state/action space to our clustered approach. We did this experiment for all the three scenarios we created: for $N = 2$, clustering brings no benefits and makes the training very unstable, since the problem is already very small and factorizing it only creates convergence issues. For $N = 4$ the situation gets already better, the convergence is slightly faster and more stable, but the real difference is with $N = 8$: the factorized version converges way faster, reaching very quickly a stable policy.

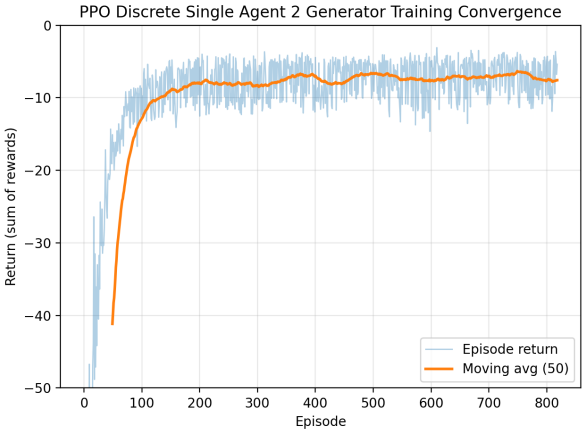

Figure 2: Single agent, $N = 2$.

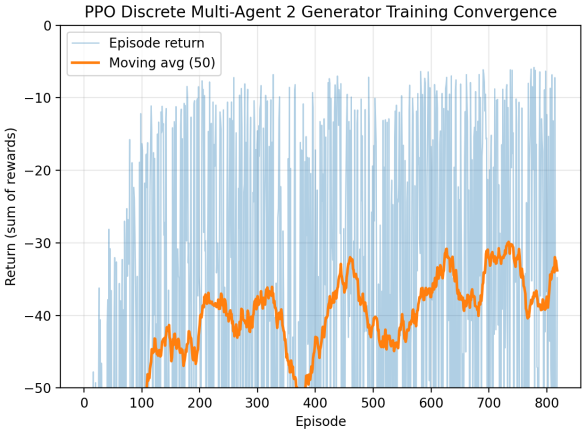

Figure 3: Multi-agent, $N = 2$.

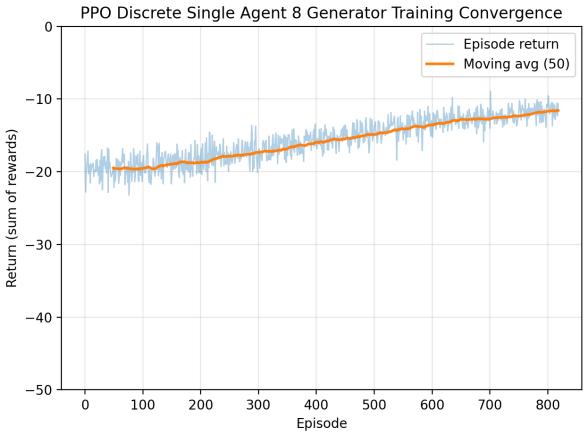

Figure 4: Single agent, $N = 8$.

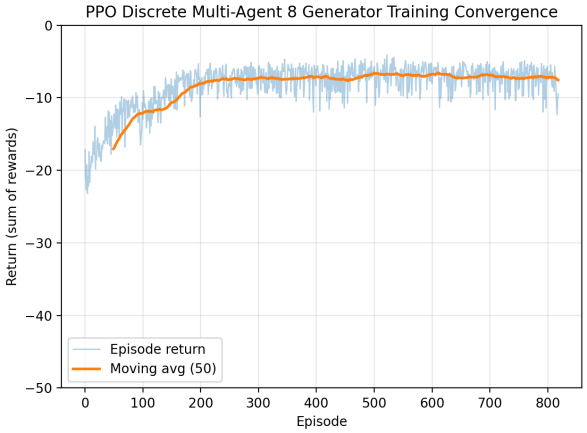

Figure 5: Multi-agent, $N = 8$.

4.2. Continuous Power Grid results

In the continuous variant, MI is estimated using KSG instead that computed exactly. Despite this added difficulty, the clusters remain coherent: actions are grouped with the most affected state variable and the overall factorization make sense with the structure of the problem.

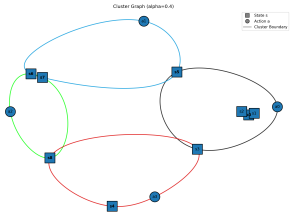

Figure 6: Example of clustering graph for the continuous Power Grid problem with 4 generators.

From a learning perspective, the single agent baseline in this case already converges very

quickly, making improvements harder to notice at small scales. For $N = 2$ and $N = 4$ in our tests the convergence was slightly more unstable, while at $N = 8$ the two versions are already comparable, suggesting that, while the estimator-based clustering is effective, the benefits of factorization are only visible on more challenging problems.

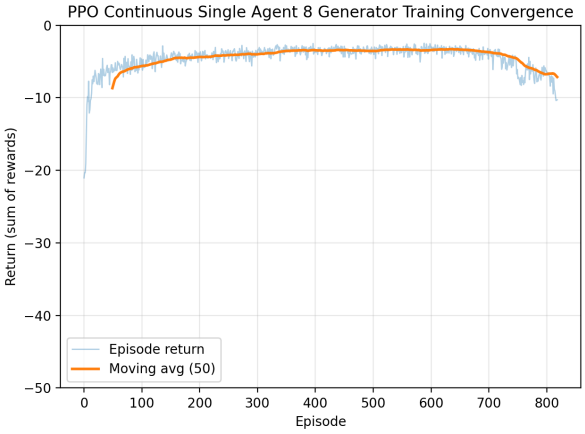

Figure 7: Single agent, $N = 8$.

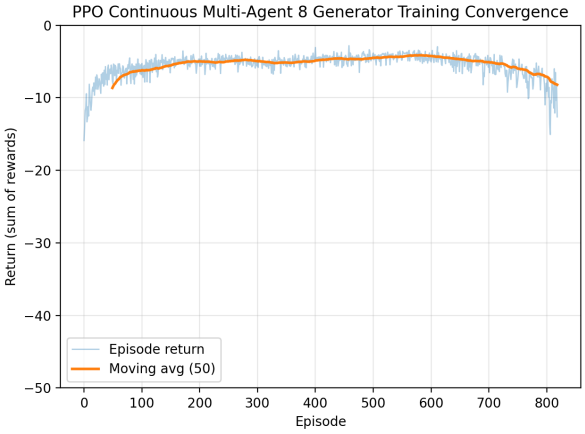

Figure 8: Multi-agent, $N = 8$.

4.3. Flatland results

For Flatland environment we used the SwitchFL version, where actions are taken by switches and transitions occur asynchronously. Since our MI requires transitions based on time instants, we exported observations and actions and reconstructed a dataset grouping elements by timestamp. From the reconstructed sequence, we computed directly the MI matrix, since in this setting all the variables are discrete. Also in this case we applied normalization, but since we didn't know the exact cardinality of some variables we considered the number of unique values found in our simulation.

We then applied our clustering algorithm considering as state components the station delays. This idea is based on the fact that we want to have clusters of switches that are informative on the same subset of stations, which should result in meaningful grouping around traffic bottlenecks. Since the initial number of agents is very large (around 280 switches in our case), we tuned the merging parameter *alpha* to promote aggregation.

The resulting structure consists of one highly interconnected cluster that observes most of the network and contains the most informative switches, along smaller clusters that coordinate switches among smaller subsets of stations. A graph visualization in this case is very impractical due to the huge number of elements that should be represented. This experiment was a very useful test case for our algorithm: having a number of actions way bigger than the number of states, the time taken to clusters goes up from seconds to a few minutes. The reulting factorization reamins feasible, so this should still bring an advantage in terms of time considered the amount of time is should save during the learning process.

5. Conclusions

The Power Grid experiments indicate that our MI based approach produces meaningful factorization and is able to scale with the system dimension. The approach becomes more and more efficient as the system grows, giving the best results at $N = 8$. Even in the continuous scenario, the results we had suggest that advantages may emerge more clearly for larger networks.

The Flatland test instead, supports the generality of the approach, proving its ability to cluster different variables in a completely different domain.

Overall, the results suggest that our MI-based clustering is a promising, domain-agnostic strategy for factorizing network MDPs, improving training efficiency and stability if the problems are sufficiently large and complex.

6. Bibliography

References

[1] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004. Preprint: arXiv:cond-mat/0305641.

[2] Jiaoyang Li, Zhe Chen, Yi Zheng, Shao-Hung Chan, Daniel Harabor, Peter J. Stuckey, Hang Ma, and Sven Koenig. Scalable rail planning and replanning: Winning the 2020 flatland challenge. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS)*, pages 477–485. AAAI Press, 2021.

[3] Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 12:94610–94631, 2024.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.