# Thesis abstract: Dynamic Weights in Multi-Objective Deep Reinforcement Learning

Axel Abels[1], Diederik M. Roijers[2], and Tom Lenaerts[1]

[1] Université Libre de Bruxelles, Brussels, Belgium
[2] Vrije Universiteit Brussel, Brussels, Belgium

In this thesis [1], we study the possibilities of Deep RL in the dynamic weights setting. In this setting the relative importance of objectives changes over time, as recognized by [3] who proposed a tabular Reinforcement Learning algorithm to deal with this problem. However, this earlier work is not feasible for reinforcement learning settings in which the input is high-dimensional, necessitating the use of function approximators, such as neural networks.

Existing Deep (MO)RL algorithms are insufficient in the Dynamic Weights setting because they build a complete set of policies in advance or spend a long time adapting to weight changes. We propose two approaches to allow an agent to quickly perform well and immediately adapt to changes in the weight vector.

In line with existing work on dynamic weights [3] and multi-objective deep RL for different settings [2], we first propose a *Multi-Network (MN)* algorithm that gradually builds a set of policies represented by Q-networks, $\Pi$. In MN, a policy is trained for the active weight vector $\mathbf{w}$ following *scalarized deep Q-learning*[2]. When the active weights change, the trained policy is saved if it is optimal for at least one encountered weight vector. To limit memory usage and ensure fast retrieval by keeping $\Pi$ small, all old policies made redundant by the new policy are removed from $\Pi$. In addition, instead of starting from scratch, we fully (MN) or partially (MN PAR) copy the best past policy for each new weight vector.

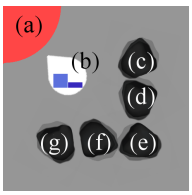


**Fig. 1.** Minecart environment with 5 mines ((c) to (g)) containing varying amounts of 2 ores. The 2 bars on the minecart (b) indicate how much of each ore is present in the cart. Ores are collected on mines and sold on the base (a).

The multi-network approach performs well if policies can converge to accurate Q-values before being saved. This can be impossible when intervals between weight changes are short or when feature extraction does not generalize across policies. We therefore propose *Conditioned Network (CN)*, in which a single network is trained to output Q-value-vectors conditioned on an input weight vector by feeding the weights into the Q-value heads. To promote convergence for the new weight vector's policy and to maintain previously learned policies, mini-batches are trained w.r.t. the current weight vector and a random previously encountered weight vector.

By themselves, MN and CN bias the replay buffer to recent weight vectors. To prevent this, we propose *diverse experience replay (DER)*, a framework to maintain a diverse replay buffer from which relevant experiences can be sampled

**Table 1.** Average episodic regret (**Mean$\Delta$**, i.e., the average distance to the optimal policy, lower is better) and performance relative to MO with Standard ER baseline (>**baseline**, negative % indicate a lower regret than the baseline, i.e., better performance) for both weight change scenarios. We distinguish **overall** performance and **final** performance (respectively averaged over the whole run and over the last 250k steps) with and without Diverse Experience Replay (**DER**). CN with DER outperforms other algorithms for both weight change scenarios.

| | | Overall | | | | Final | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Standard ER | | DER | | Standard ER | | DER | |
| | Algorithm | Mean$\Delta$ | >baseline | Mean$\Delta$ | >baseline | Mean$\Delta$ | >baseline | Mean$\Delta$ | >baseline |
| Sparse Weight Changes | MO | 0.332 | – | 0.291 | -12.257% | 0.268 | – | 0.257 | -3.974% |
| | MN | *0.255* | *-23.23%* | 0.211 | -36.54% | *0.14* | *-47.745%* | ***0.063*** | ***-76.395%*** |
| | MN PAR | 0.47 | +41.599% | 0.413 | +24.488% | 0.381 | +42.259% | 0.286 | +6.769% |
| | CN | 0.259 | -22.081% | ***0.18*** | ***-45.73%*** | 0.169 | -36.903% | 0.068 | -74.672% |
| | CNA | 0.349 | +5.061% | 0.212 | -36.193% | 0.318 | +18.512% | 0.088 | -67.22% |
| | CNC | 0.3 | -9.725% | 0.221 | -33.388% | 0.197 | -26.463% | 0.102 | -61.757% |
| Regular Weight Changes | MO | 0.399 | – | 0.429 | +7.482% | 0.258 | – | 0.319 | +23.503% |
| | MN | 0.719 | +80.223% | 0.748 | +87.482% | 0.67 | +159.854% | 0.709 | +174.899% |
| | MN PAR | 0.694 | +73.852% | 0.642 | +60.931% | 0.656 | +154.09% | 0.642 | +148.965% |
| | CN | 0.219 | -45.03% | ***0.215*** | ***-46.048%*** | 0.069 | -73.069% | ***0.064*** | ***-75.141%*** |
| | CNA | 0.275 | -31.14% | 0.284 | -28.782% | 0.149 | -42.102% | 0.149 | -42.235% |
| | CNC | *0.218* | *-45.431%* | 0.237 | -40.559% | *0.065* | *-74.798%* | 0.071 | -72.593% |

for weight vectors whose policies have not been applied recently. DER replaces the circular model of standard replay buffers by diversity-based memorization. In this thesis we diversify over the space of possible trajectory returns.

We also propose an original multi-objective benchmark, the *Minecart problem* which models the challenges of resource collection, has a continuous state space, stochastic transitions and delayed rewards. In Minecart, an agent must quickly adapt to changes in resource values to efficiently mine different ores.

We evaluate the performance when weight changes are sparse, as in [3], in which case an agent's policy and replay buffer could overfit to the active weights. Then, we look at how our algorithms perform when weights change regularly, in which case it can be tempting to learn a single sub-optimal policy. We compare CN and MN against Scalarized Deep Q-Learning on the current weight vector (MO) as a baseline. We find that CN performs best overall. Furthermore, training the conditioned network on current and past weight vectors (CN) performs better than training only on the current weight vector (CNC) or on randomly sampled past weight vectors (CNA). MN only performs well when given enough training time to learn accurate Q-values. Moreover, DER significantly improves performance when diversity cannot be expected to occur automatically.

## References


1. Abels, A.: Dynamic Preferences in Deep Multi-Objective Reinforcement Learning. Master's thesis, Université Libre de Bruxelles (2018)
2. Mossalam, H., Assael, Y.M., Roijers, D.M., Whiteson, S.: Multi-objective deep reinforcement learning. CoRR **abs/1610.02707** (2016)
3. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: Proceedings of the 22nd International Conference on Machine Learning. pp. 601–608. ICML '05, ACM, New York, NY, USA (2005)