Learning to Coordinate with Coordination Graphs in Repeated Single-Stage Multi-Agent Decision Problems

Eugenio Bargiacchi [1] **Timothy Verstraeten** [1] **Diederik M. Roijers** [1 2] **Ann Nowé** [1] **Hado van Hasselt** [3]

Abstract

Learning to coordinate between multiple agents is an important problem in many reinforcement learning problems. Key to learning to coordinate is exploiting loose couplings, i.e., conditional independences between agents. In this paper we study learning in repeated fully cooperative games, *multi-agent multi-armed bandits (MAMABs)*, in which the expected rewards can be expressed as a coordination graph. We propose *multi-agent upper confidence exploration (MAUCE)*, a new algorithm for MAMABs that exploits loose couplings, which enables us to prove a regret bound that is logarithmic in the number of arm pulls and only linear in the number of agents. We empirically compare MAUCE to sparse cooperative Q-learning, and a state-of-the-art combinatorial bandit approach, and show that it performs much better on a variety of settings, including learning control policies for wind farms.

1. Introduction

Many decision problems can be phrased as coordination problems of many artificial intelligent agents (Boutilier, 1996). Examples include robot soccer (Kok et al., 2003), warehouse commissioning (Claes et al., 2017), and traffic light control (Wiering, 2000). We consider the cooperative case, where there is a single goal to be optimised. A naive approach could be to consider a super agent that decides on the actions of all agents involved, which could easily result in an action space which is prohibitively large. However, many coordination tasks have *loose couplings*. This means that the total reward to optimise can be decomposed into a sum of *local rewards* that only depend on (possibly overlapping) subsets of agents. Then, each agent's action can only directly affects the rewards of few small subsets of agents. Key to making coordination efficient is exploiting such loose couplings.

For an example of such a coordination task, consider an autonomously controlled wind farm in which each agent represents a wind turbine that is able to adjust the alignment of its blades to the wind (see Section 6.3). Each turbine can maximize its own power output by aligning the blades exactly perpendicular to the wind, but doing so may hinder turbines that are behind it due to turbulence (Van Dijk et al., 2016). It should be possible to do better through coordination. However, considering the full joint action over all turbines leads to a high-dimensional action space, which would be hard to optimise. Instead, we can see that this problem is loosely coupled, by noting that the power output of each turbine only directly depends on a small subset of other turbines — the turbines upwind within a certain distance. This means that the total output can be phrased as a sum of local rewards that depend on small subsets of agents.

In this paper, we formalize multi-agent multi-armed bandits (MAMABs) and investigate how to balance exploration and exploitation in the *joint* action taken by the agents, such that the loss due to taking suboptimal joint actions during learning is bounded. Building on the *upper confidence bound (UCB)* framework (Auer et al., 2002) for single-agent multi-armed bandits, we formulate a new algorithm that we call *multi-agent upper confidence exploration (MAUCE)* (Section 4). MAUCE balances exploitation and exploration using local estimates and local upper confidence bounds.

We prove in Section 5 that MAUCE achieves a regret bound that depends on the *harmonic mean* of the local upper confidence bounds, rather than their sum, as we would get by applying the combinatorial bandit framework (Cesa-Bianchi & Lugosi, 2012; Chen et al., 2013). This leads to a regret logarithmic in the number of arm-pulls and linear in the number of agents. In contrast, the naive approach of considering the full joint action is exponential in the number of agents. In Section 6 we compare empirically the performance of MAUCE to other approaches from the literature, and show that it achieves much less regret in various settings, including wind farm control.

[1]Dept. of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium [2]Fac. of Science, Vrije Universiteit Amsterdam, The Netherlands [3]Google DeepMind, London, UK. Correspondence to: Eugenio Bargiacchi <svalorzen@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

2. Related Work

Multi-agent reinforcement learning and planning with loose couplings has mainly been studied in sequential problems (Guestrin et al., 2002; Kok & Vlassis, 2006; De Hauwere et al., 2010; Scharpff et al., 2016). In such sequential settings however, the value function does not permit an exact factorization. Therefore, only in the planning setting (Scharpff et al., 2016), some guarantees can be provided. For learning (Kok & Vlassis, 2006), the focus has been on empirical performance. In this paper, we focus on MAMAB, which permit an exact factorization of the value function.

This work is related to combinatorial bandits (Bubeck & Cesa-Bianchi, 2012; Cesa-Bianchi & Lugosi, 2012; Gai et al., 2012; Chen et al., 2013), in which sets of arms can be pulled simultaneously. In our setting, these variables correspond to the different agents, and similarly to the combinatorial bandit framework, the action space grows exponentially with the size of the sets of rewards. We consider a specific variant, called the *semi-bandit* problem (Audibert et al., 2011), in which local components of the global reward are observable. Chen et al. (2013) considered this variant and constructed an algorithm. However, that algorithm assumes access to an (α, β)-oracle that provides a joint action that outputs an α fraction of the optimal expected reward with a certain probability β. Instead, we assume the availability of a coordination graph, which is often a more reasonable assumption in multi-agent settings.

3. Background

Before introducing our new algorithm, we first need to define our learning problem. This problem, the multi-agent multi-armed bandit, is a repeated fully cooperative multi-agent game. We first define the single-agent version of our setting, and then add the multi-agent elements. The single-agent version of our setting is commonly known as the *multi-armed bandit (MAB)*:

Definition 1. *A single-agent multi-armed bandit (MAB) (Thompson, 1933) is a tuple $\langle \mathcal{A}, F \rangle$ where*

- *$\mathcal{A}$ is a set of actions or arms, and*
- *$F(a)$, called the reward function, is a random function taking an arm, $a \in \mathcal{A}$, as input. Specifically, for each $a \in \mathcal{A}$, $F(a)$ is a random variable associated with a probability distribution $P_a : \mathbb{R} \to [0, 1]$ over real-valued rewards r.*

We refer to the mean reward of an arm as $\mu_a = \mathbb{E}_{P_a}[r] = \int_{-\infty}^{\infty} r P_a(r) dr$, and to the optimal reward as the mean reward of the best arm $\mu^* = \max_a \mu_a$.

The goal of an agent interacting with a MAB is to minimize the expected regret.

Definition 2. *The expected cumulative regret of pulling a sequence of arms for timestep $t = 1$ to the horizon T (following the definition of Agrawal & Goyal, 2012), is*

$$\mathbb{E}\left[\sum_{t=1}^{T} \mu^* - \mu_{a(t)}\right],$$

where $a(t)$ is the arm pulled at time t, and $n_a(t)$ is the number of times arm a is pulled until timestep t.

In a *multi-agent multi-armed bandit (MAMAB)* there are multiple agents, and the rewards are factored:

Definition 3. *A multi-agent multi-armed bandit (MAB) is a tuple $\langle \mathcal{A}, \mathcal{D}, F \rangle$ where*

- *$\mathcal{D}$ is the set of m enumerated agents,*
- *$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$ is a set of joint actions, which is the Cartesian product of the sets of individual actions, $\mathcal{A}_i$, for each of the m agents in $\mathcal{D}$, and*
- *$F(\mathbf{a})$, called the global reward function, is a random function taking a joint action, $\mathbf{a} \in \mathcal{A}$, as input, but with added structure. Specifically, there are ρ possibly overlapping subsets of agents, and the global reward is decomposed into ρ local noisy reward functions: $F(\mathbf{a}) = \sum_{e=1}^{\rho} f^e(\mathbf{a}^e)$ where $f^e(\mathbf{a}^e) \in [0, r^e_{\max}]$. A local function f^e only depends on the joint action $\mathbf{a}^e$ of the subset $\mathcal{D}^e$ of agents.*

We refer to the mean reward of a joint action as $\mu_{\mathbf{a}}$, which in turn is factorized into the same local reward components as $F(\mathbf{a})$: $\mu_{\mathbf{a}} = \sum_{e=1}^{\rho} \mu(\mathbf{a}^e)$. For simplicity, we refer to an agent i by its index.

$\mu_{\mathbf{a}}$ thus maps joint actions to real-valued expected rewards via real-valued local expected rewards, i.e., it is a *coordination graph (CoG)* (Guestrin et al., 2002; Kok & Vlassis, 2006). When $\mu_{\mathbf{a}}$ and all its components are known, it can be used to extract the optimal reward μ^*. A naive way to do so would be to *'flatten'* the CoG, i.e., enumerate all joint actions, compute their associated mean reward, and then maximize. However, this is typically infeasible, as the number of joint actions, $A \equiv |\mathcal{A}|$, is exponential in the number of agents. For instance, if each agent has two actions, then $A = 2^m$. Therefore, extracting the optimal reward and associated actions is typically done via algorithms like *variable elimination (VE)*. In VE, agents are eliminated from the CoG sequentially, thus solving the maximization problem as a series of *local subproblems*: one per agent. When an agent is eliminated, VE computes its best responses to all possible actions of its neighbors, i.e., the agents with which it shares a local reward function. The local values of these best responses are then used to create a new local mean reward, replacing those to which the eliminated agent was connected. This exploits the graphical structure resulting from the factorization, and the size of the local subproblems depends only on the *induced width*, i.e., how many agents

Algorithm 1 MAUCE

Input: An MAMAB with a factorized reward function, $F(\mathbf{a}) = \sum_{e=1}^{\rho} f^e(\mathbf{a}^e)$, a time horizon T
Initialize $\hat{\mu}^e(\mathbf{a}^e)$ and $n^e(\mathbf{a}^e)$ to zero.
for $i = 1$ **to** T **do**
  $\mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{\mu}_t(\mathbf{a}) + c_t(\mathbf{a})$ where,
    $\hat{\mu}_t^e(\mathbf{a}) = \sum_{e=1}^{\rho} \hat{\mu}_t(\mathbf{a}^e)$ and,
    $c_t(\mathbf{a}) = \sqrt{\frac{1}{2}\left(\sum_{e=1}^{\rho} n_t^e(\mathbf{a}^e)^{-1}(r_{\max}^e)^2\right)\log(tA)}$
  $r_t = \sum_{e=1}^{\rho} r_t^e(\mathbf{a}^e)$ (execute $\mathbf{a}$, obtain local rewards)
  Update $\hat{\mu}_t^e(\mathbf{a}^e)$ using $r_t^e(\mathbf{a}^e)$ for all $\mathbf{a}^e \subset \mathbf{a}_t$
  Increment $n_t^e(\mathbf{a}^e)$ by 1 for all $\mathbf{a}^e \subset \mathbf{a}_t$
end for

the eliminated agent shares a local reward function with at the time of its elimination. When the coordination graph is sparse, i.e., agents are only involved in a small number of local reward functions, the induced width is typically much smaller than the size of the joint action space, making the maximization problem tractable.

When we are not simply maximizing over the joint actions to extract the optimal reward, but also need to explore to learn what the values of the mean rewards are, the situation becomes more complex. Again, we could 'flatten' the MAMAB by treating each joint action as a separate arm in a single-agent MAB, but this quickly leads to too many arms to be able to learn effectively with popular algorithms such as UCB (Auer et al., 2002) of which the regret bounds depend on the number of arms. Furthermore, just adding the standard exploration bonuses to each of the local mean rewards leads to over-exploration, as we will show experimentally in Section 6. Instead, we propose to treat exploration and exploitation as separate objectives during a VE-like scheme, and taking inspiration from the multi-objective literature (Roijers et al., 2015), define a new VE-like subroutine, that allows us to define a MAUCE (Section 4) for which we can prove a much tighter regret-bound.

4. Multi-Agent Upper Confidence Exploration

In this section we propose our new algorithm for MAMABs: *Multi-Agent Upper Confidence Exploration (MAUCE)* (Algorithm 1).

MAUCE executes a joint action at every timestep that maximizes the estimated mean reward for a given factorization of the reward function, $\hat{\mu}(\mathbf{a})$, plus an exploration bonus, $c_t(\mathbf{a})$, that is computed using the same factorization. To do so, it keeps mean estimates of local rewards $\hat{\mu}^e(\mathbf{a}^e)$, and local counts $n_t^e(\mathbf{a}^e)$ for each subset of agents. These local estimates depend only on the subset of actions $\mathbf{a}^e \subset \mathbf{a}$ for this group of agents $\mathcal{D}^e \subset \mathcal{D}$. Not all joint actions have to be selected often, or even at all. Note that the counts $n_t^e(\mathbf{a}^e)$ used to compute the bonus for an action $\mathbf{a}$ can change over time, even if the joint action $\mathbf{a}$ has never been selected, because MAUCE observes and uses the local rewards, $r_t^e(\mathbf{a}^e)$. This enables the algorithm to exploit the graphical structure to compute tighter exploration bonuses while guaranteeing a tight regret bound. Despite not guaranteeing to explore all joint actions, the algorithm achieves guaranteed logarithmic regret. The proof for this regret bound is given in Section 5.

Besides the local counts, the exploration bonus also depends on the maximum value of the local rewards $r_{\max}^e$, the time index t, and A. We note that A is exponential in the number of agents. Contrary to single-agent MABs, it is not trivial to maximize over $\hat{\mu}(\mathbf{a}) + c_t(\mathbf{a})$, as we need to maximize over a A efficiently, and $c_t(\mathbf{a})$ is a non-linear function in the local counts $n_t^e(\mathbf{a}^e)$. Hence, MAUCE requires a special algorithm to perform this maximization.

4.1. Maximizing $\hat{\mu}(\mathbf{a}) + c(\mathbf{a})$

We observe that we can express the estimated mean as the sum of local estimated means, and that $c_t(\mathbf{a})$ can be expressed as a function over the inverse counts: $y(\sum_{e=1}^{\rho} n_t^e(\mathbf{a}^e)^{-1}(r_{\max}^e)^2)$. Hence, when we write down the local estimates as two-element vectors: an estimated mean component and a weighted inverse counts component,

$$\mathbf{v}^e(\mathbf{a}^e) = (\hat{\mu}^e(\mathbf{a}^e), n_t^e(\mathbf{a}^e)^{-1}(r_{\max}^e)^2), \tag{1}$$

we can express the mean plus exploration bonus as a function applied to the sum of these vectors:

$$z_t(\mathbf{v}(\mathbf{a})) = \hat{\mu}(\mathbf{a}) + c_t(\mathbf{a}) = \mathbf{v}[1] + \sqrt{\frac{1}{2}\mathbf{v}[2]\log(tA)}, \tag{2}$$

where

$$\mathbf{v}(\mathbf{a}) = \sum_{e=1}^{\rho} \mathbf{v}^e(\mathbf{a}^e). \tag{3}$$

Vector formulations of reward, as those of Equations 1–3, are often used in the multi-objective decision making literature (Roijers & Whiteson, 2017). Consider *multi-objective variable elimination (MOVE)* (Rollón & Larrosa, 2006; Roijers et al., 2015), a multi-objective framework based on variable elimination that is able to handle vectors. Instead of single best responses for eliminated agents, MOVE produces sets of vectors that are possibly optimal as intermediate results. At each agent elimination, MOVE computes all possible (local) value vectors for the subproblem of eliminating the agent i, and *prunes* away those that are locally dominated. After MOVE eliminates the last agent it outputs a possibly very large set of possibly optimal vectors, e.g., a Pareto front or convex coverage set.

In contrast to MOVE, we only want to output a single vector, i.e., the one that maximizes z_t (Equation 2). To do this we

tighten MOVE's simple domination pruning by introducing lower and upper bounds on the exploration part of the vector. This results in an algorithm in which the number of vectors in the intermediate solution sets steeply decreases in the last agent eliminations (in contrast to MOVE, in which the intermediate sets typically continue to grow in size). We call this algorithm *upper confidence variable elimination (UCVE)*.

First, we define the input of UCVE. Specifically, to be able to work with *sets* of vectors as intermediate results, we first reformulate the problem of finding the optimal joint action in these terms. Specifically, we define the input to UCVE as a set $\mathcal{F}$ of *local upper confidence vector set functions (UCVSFs)*. For each f^e of $F(\mathbf{a})$, $\mathcal{F}$ contains an identically scoped UCVSF u^e. Each u^e initially contains a singleton set, $u^e(\mathbf{a}^e) = \{\mathbf{v}^e(\mathbf{a}^e)\}$, where $\mathbf{v}^e(\mathbf{a}^e)$ is defined as in Equation 1. Eliminating an agent i, is performed by replacing all $u^e(\mathbf{a}^e)$ which have i in scope, i.e., $i \in \mathcal{D}^e$, by a new function that incorporates the possibly optimal responses of i. These possibly optimal responses are again vectors in the form of Equation 1.

Algorithm 2 UCVE($\mathcal{F}$)

Input : A set of local upper confidence vector set functions $\mathcal{F}$ and an elimination order q (a queue with all agents)
Output : An optimal joint action, $\mathbf{a}^*$

1: **while** q is not empty **do**
2: $i \leftarrow$ q.dequeue()
3: $\mathcal{F}_i \leftarrow$ the subset of UCVSFs in $\mathcal{F}$ that have i in scope
4: $x_u,\ x_l \leftarrow$ compute upper and lower bounds on the exploration part of the vectors for the remaining factors in $\mathcal{F} \setminus \mathcal{F}_i$
5: $u^{new}(\cdot) \leftarrow$ a new UCVSF
6: **for all** $\mathbf{a}^e_{-i} \in \mathcal{A}_{D^e \setminus \{i\}}$ **do**
7: $\mathcal{V} \leftarrow \bigcup_{a_i \in \mathcal{A}_i} \bigoplus_{u^e \in \mathcal{F}_i} u^e(\mathbf{a}^e_{-i} \times \{a_i\})$
8: $u^{new}(\mathbf{a}^e_{-i}) \leftarrow$ prune$(\mathcal{V}, x_u, x_l)$
9: **end for**
10: $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{F}_i \cup \{u^{new}\}$
11: **end while**
12: $u \leftarrow$ retrieve final factor from $\mathcal{F}$
13: **return** the optimal joint action from u

UCVE is provided in Algorithm 2. Note that we only describe what is traditionally known as the *forward* pass of the variable elimination scheme. This is because to retrieve the optimal joint action, we make use of the tagging scheme of MOVE (Roijers et al., 2015), where vectors are tagged with the appropriate action of an agent during its elimination.

UCVE eliminates all agents in a predetermined order, q, in the main loop (line 1–11). On line 2 the next agent i is popped off the queue, and on line 3 the factors that have i in scope, $\mathcal{F}_i$ are collected. The functions in $\mathcal{F}_i$ will be replaced in $\mathcal{F}$ by a new UCVSF, u^{new}, incorporating the possible best responses to every possible local joint action of the neighbors of i. This new UCVSF has all the neighboring agents $\mathcal{D}^e \setminus \{i\}$ of agent i in scope.

First, all possible vectors $\mathcal{V}$ that can be made with the UCVSFs in $\mathcal{F}_i$ are computed (on line 7), across all actions of i, for a given $\mathbf{a}^e_{-i}$:

$$\mathcal{V} = \bigcup_{a_i \in \mathcal{A}_i} \bigoplus_{u^e \in \mathcal{F}_i} u^e(\mathbf{a}^e_{-i} \times \{a_i\}),$$

where $\mathcal{A}_i$ is the action space of agent i, and the cross-sum operator $A \oplus B$ is defined as $A \oplus B = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A \wedge \mathbf{b} \in B\}$. Note that the resulting actions always include the appropriate actions a_i (which is under the union) and the appropriate actions from $\mathbf{a}^e_{-i}$. After $\mathcal{V}$ is computed, the vectors in $\mathcal{V}$ that cannot lead to an optimal joint action need to be pruned.

Each vector in $\mathcal{V}$ consists of an estimated mean and a weighted inverse counts part that will lead to the exploration bonus. Because the weighted inverse counts cannot be linearly added to the estimated mean, we cannot a priori tell whether a vector $\mathbf{v} \in \mathcal{V}$ is better than another vector $\mathbf{v}' \in \mathcal{V}$ when $\mathbf{v}[1] > \mathbf{v}'[1]$ but $\mathbf{v}[2] < \mathbf{v}'[2]$. We thus define a pruning operator that satisfies the two conditions necessary for correctness in a multi-objective variable elimination-scheme (Roijers, 2016), i.e., (1) no excess values are kept, and (2) no unnecessary values are returned after the last agent elimination. We compute an upper and a lower bound on the exploration bonus for the remaining functions in $\mathcal{F} \setminus \mathcal{F}_i$, using the sums of the maximum, resp. minimum, values for the exploration part, $x_u = \sum_{u^e \in \mathcal{F} \setminus \mathcal{F}_i} \max_{\mathbf{a}^e} \max_{\mathbf{v} \in u^e(\mathbf{a}^e)} \mathbf{v}[2]$ and $x_l = \sum_{u^e \in \mathcal{F} \setminus \mathcal{F}_i} \min_{\mathbf{a}^e} \min_{\mathbf{v} \in u^e(\mathbf{a}^e)} \mathbf{v}[2]$. Specifically, a vector $\mathbf{v} \in \mathcal{V}$ cannot contribute to the optimal value if there is another vector $\mathbf{v}' \in \mathcal{V}$ such that

$$\mathbf{v}[1] + \sqrt{\frac{1}{2}(\mathbf{v}[2] + x_u)\log(tA)} < \mathbf{v}'[1] + \sqrt{\frac{1}{2}(\mathbf{v}'[2] + x_l)\log(tA)}$$

Hence, prune removes those candidate local upper confidence vectors that cannot contribute to finding the maximal mean plus exploration bonus. This immediately satisfies correctness condition (1), as it follows from the definition of upper and lower bounds.

After all agents have been eliminated, there is only one UCVSF left, containing a single local upper confidence vector. UCVE retrieves the optimal vector— which maximizes the $\hat{\mu}(\mathbf{a}) + c_t(\mathbf{a})$ —and the associated joint action, $\mathbf{a}_t$, from the final UCVSF, satisfying correctness condition 2. We thus have defined an efficient algorithm that correctly outputs the joint action that maximizes $\hat{\mu}(\mathbf{a}) + c_t(\mathbf{a})$, and can therefore be used to select joint actions inside of MAUCE.

5. Linear Regret Bound for Collaborative Multi-Agent Settings

The efficiency of our method is achieved by exploiting localized structures within the global reward. If there are no such structures, then a worst-case regret of $O(A \log T)$ can be achieved by employing an upper-confidence bound (UCB) algorithm (Auer et al., 2002; Auer & Ortner, 2010). As A grows exponentially with the number of agents, the global action space is simply too large to make this bound of practical use. However, we show that when the global reward can be decomposed into local reward functions over subsets of agents, the regret obtained when using our method becomes much smaller. In fact, when the local rewards all have the same range, the regret of our method becomes *linear* in the number of agents.

Assume there are ρ subsets of agents, called *groups*, and that there is a decomposition of the global reward $F(\mathbf{a}) = \sum_{e=1}^{\rho} f^e(\mathbf{a}^e)$ where $f^e(\mathbf{a}^e) \in [0, r^e_{\max}]$. W.l.o.g., let us also consider non-identical groups and that $\sum_{e=1}^{\rho} r^e_{\max} = 1$. The local function f^e only depends on the actions of a group $\mathcal{D}^e$. We maintain the sample mean reward $\hat{\mu}^e_t(\mathbf{a}^e)$ and number of pulls $n^e_t(\mathbf{a}^e)$ for each local joint action $\mathbf{a}^e$ taken by group $\mathcal{D}^e$ at time t. Finally, we define the gap between the true expected rewards of the optimal action $\mathbf{a}_*$ and action $\mathbf{a}$ to be $\Delta(\mathbf{a}) = \mathbb{E}[F(\mathbf{a}_*)] - \mathbb{E}[F(\mathbf{a})]$.

Theorem 1. *If at each time t we choose $\mathbf{a}_t$ such that*

$$\begin{aligned}\mathbf{a}_t &= \arg\max_{\mathbf{a}} w_t(\mathbf{a}) \\ &= \arg\max_{\mathbf{a}} \hat{\mu}_t(\mathbf{a}) + c_t(\mathbf{a}),\end{aligned}$$

with

$$\hat{\mu}_t(\mathbf{a}) = \sum_{e=1}^{\rho} \hat{\mu}^e_t(\mathbf{a}^e),$$

$$c_t(\mathbf{a}) = \sqrt{\frac{1}{2}\left(\sum_{e=1}^{\rho} n^e_t(\mathbf{a}^e)^{-1}(r^e_{\max})^2\right)\log(tA)},$$

then the expected global regret is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T}\Delta(\mathbf{a}_t)\right] \leq \frac{2N\sum_{e=1}^{\rho}(r^e_{\max})^2\log(TA)}{\min_{\mathbf{a}}\Delta(\mathbf{a})^2} + \log T + 1,$$

where $N \equiv \sum_{e=1}^{\rho}\prod_{i\in\mathcal{D}^e} A_i$ is the total number of local joint actions and $A_i = |\mathcal{A}_i|$.

Proof. Let $C_t(\mathbf{a})$ be the event that $\Delta(\mathbf{a}) > 2c_t(\mathbf{a})$ holds and $\overline{C}_t(\mathbf{a})$ its negation. By the law of the excluded middle, we can then write

$$\begin{aligned}\mathbb{E}[\Delta(\mathbf{a}_t)] = \mathbb{E}[\Delta(\mathbf{a}_t) \mid C_t(\mathbf{a}_t)]\, P(C_t(\mathbf{a}_t)) \\ + \mathbb{E}\left[\Delta(\mathbf{a}_t) \mid \overline{C}_t(\mathbf{a}_t)\right] P(\overline{C}_t(\mathbf{a}_t))\end{aligned}$$

which implies

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}\Delta(\mathbf{a}_t)\right] \leq \sum_{t=1}^{T} P(C_t(\mathbf{a}_t)) \\ + \mathbb{E}\left[\Delta(\mathbf{a}_t) \mid \overline{C}_t(\mathbf{a}_t)\right]\mathbb{E}\left[\mathcal{I}\{\overline{C}_t(\mathbf{a}_t)\}\right]\end{aligned}$$

where $\mathcal{I}\{\cdot\}$ is the indicator function. We first look at all time steps on which $C_t(\mathbf{a}_t)$ holds. Specifically, we bound the probability that this event occurs. Using the law of total probability and chain rule, we can derive

$$P(C_t(\mathbf{a}_t)) \leq \sum_{a\in\mathcal{A}} P(\mathbf{a} = \mathbf{a}_t \mid C_t(\mathbf{a})) \tag{4}$$

By definition, action $\mathbf{a}_t$ maximizes the upper bound $w_t(\cdot)$. Therefore,

$$\begin{aligned}&P(\mathbf{a} = \mathbf{a}_t \mid C_t(\mathbf{a})) \\ &\quad= P(w_t(\mathbf{a}) = w_t(\mathbf{a}_t) \mid C_t(\mathbf{a})) \\ &\quad\leq P(w_t(\mathbf{a}) \geq w_t(\mathbf{a}_*) \mid C_t(\mathbf{a})) \\ &\quad= P(\hat{\mu}_t(\mathbf{a}) - \hat{\mu}_t(\mathbf{a}_*) \geq c_t(\mathbf{a}_*) - c_t(\mathbf{a}) \mid C_t(\mathbf{a})) \\ &\quad\leq \exp\left(-\frac{2(\Delta(\mathbf{a}) + c_t(\mathbf{a}_*) - c_t(\mathbf{a}))^2}{\sum_{e=1}^{\rho}(r^e_{\max})^2\,(n^e_t(\mathbf{a}^e)^{-1} + n^e_t(\mathbf{a}^e_*)^{-1})}\right)\end{aligned}$$

In the last step, we used Hoeffding's inequality. This is possible, as $\hat{\mu}_t(\mathbf{a})$ is a sum of i.i.d. random variables bounded within the interval $\left[0, \frac{r^e_{\max}}{n^e_t(\mathbf{a}^e)}\right]$. We now apply condition $C_t(\mathbf{a})$ such that $\Delta(\mathbf{a}) > 2c_t(\mathbf{a})$ and derive

$$\begin{aligned}&P(\mathbf{a} = \mathbf{a}_t \mid C_t(\mathbf{a})) \\ &\quad\leq \exp\left(-\frac{2(c_t(\mathbf{a}) + c_t(\mathbf{a}_*))^2}{\sum_{e=1}^{\rho}(r^e_{\max})^2\,(n^e_t(\mathbf{a}^e)^{-1} + n^e_t(\mathbf{a}^e_*)^{-1})}\right) \\ &\quad\leq \exp\left(-\frac{2c_t(\mathbf{a})^2 + 2c_t(\mathbf{a}_*)^2}{\sum_{e=1}^{\rho}(r^e_{\max})^2\,(n^e_t(\mathbf{a}^e)^{-1} + n^e_t(\mathbf{a}^e_*)^{-1})}\right) \\ &\quad= \exp(-\log(tA)) \\ &\quad\leq (tA)^{-1}\end{aligned}$$

Using (4), we can conclude

$$\sum_{t=1}^{T} P(C_t(\mathbf{a}_t)) \leq \sum_{t=1}^{T}(tA)^{-1}A \leq \log T + 1 \tag{5}$$

where for the last step we used $\sum_{t=1}^{T} t^{-1} < \log T + \gamma + \frac{3}{6T+2} < \log T + 1$ (Chen & Qi, 2003), where γ is Euler's constant.

Now, we look at the time steps where $\overline{C}_t(\mathbf{a}_t)$ holds. Either $\mathbf{a}_t = \mathbf{a}_*$ and $\Delta(\mathbf{a}_t) = 0$, or

$$\begin{aligned}\Delta(\mathbf{a}_t) &\leq 2c_t(\mathbf{a}_t) \\ \Delta(\mathbf{a}_t)^2 &\leq 2\log(tA)\sum_{e=1}^{\rho}(r^e_{\max})^2(n^e_t(\mathbf{a}^e_t))^{-1} \\ 1 &\leq \frac{2\log(tA)}{\min_e n^e_t(\mathbf{a}^e_t)}\frac{\sum_{e=1}^{\rho}(r^e_{\max})^2}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2}\end{aligned}$$

$$\min_e n_t^e(\mathbf{a}_t^e) \leq 2\log(TA)\frac{\sum_{e=1}^{\rho}(r_{\max}^e)^2}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2} \tag{6}$$

Note that as there are at most $N = \sum_{e=1}^{\rho}\prod_{i\in\mathcal{D}^e} A_i$ local joint actions, the left-hand side will increase every at most N time steps. Since the right-hand side is fixed and does not depend on t, (6) can only be true on at most $2N\log(TA)\sum_{e=1}^{\rho}(r_{\max}^e)^2/\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2$ different time steps. This implies that

$$\begin{aligned}\sum_{t=1}^{T}&\mathbb{E}\left[\Delta(\mathbf{a}_t)\mid\overline{C}_t(\mathbf{a}_t)\right]\mathbb{E}\left[\mathcal{I}\{\overline{C}_t(\mathbf{a}_t)\}\right]\\ &\leq\mathbb{E}\left[\sum_{t=1}^{T}\mathcal{I}\{\overline{C}_t(\mathbf{a}_t)\wedge\mathbf{a}_t\neq\mathbf{a}_*\}\right]\\ &\leq\frac{2N\log(TA)\sum_{e=1}^{\rho}(r_{\max}^e)^2}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2}\end{aligned}$$

Together with (4) and (5), this implies

$$\mathbb{E}\left[\sum_{t=1}^{T}\Delta(\mathbf{a}_t)\right]\leq\frac{2N\log(TA)\sum_{e=1}^{\rho}(r_{\max}^e)^2}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2}+\log T+1$$

□

Corollary 1. *If $A_i \leq k$ for all agents i, and if $|\mathcal{D}^e| \leq d$ for all groups $\mathcal{D}^e$, then*

$$\mathbb{E}\left[\sum_{t=1}^{T}\Delta(\mathbf{a}_t)\right]\leq\frac{2\rho k^d\left(\log T+m\log k\right)}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2}+\log T+1\,.$$

Proof. $\sum_{e=1}^{\rho} r_{\max}^e = 1$ implies $\sum_{e=1}^{\rho}(r_{\max}^e)^2 \leq 1$. Additionally, $\log A = \sum_{i=1}^{m}\log A_i \leq m\log k$. Finally, $N = \sum_{e=1}^{\rho}\prod_{i\in\mathcal{D}^e} A_i \leq \rho k^d$. □

The important thing to note is that the given regret bound is *linear* in the number of agents m and in the number of functions ρ, which implies — since $\rho \leq \binom{m}{d} < m^d$ — that it is *polynomial* in m, with degree at most $d+1$. This is a huge improvement over the naive 'flattened' regret bound, which is *exponential* in the number of agents.

Corollary 2. *If, in addition to the assumptions in Corollary 1, each local function has the same range such that $r_{\max}^e = \rho^{-1}$, then*

$$\mathbb{E}\left[\sum_{t=1}^{T}\Delta(\mathbf{a}_t)\right]\leq\frac{2k^d\left(\log T+m\log k\right)}{\min_{\mathbf{a}\neq\mathbf{a}_*}\Delta(\mathbf{a})^2}+\log T+1\,.$$

Proof. If $r_{\max}^e = \rho^{-1}$ for each e, then $\sum_{e=1}^{\rho}(r_{\max}^e)^2 = \rho^{-1}$ and therefore $N\sum_{e=1}^{\rho}(r_{\max}^e)^2 = k^d$. □

Note that this implies that under the assumption that each local function has the same range, 1) the regret no longer depends on ρ and 2) the regret is *linear* in the number of agents.

6. Experiments

In order to test the performance of MAUCE, and compare it to competing approaches, we tested it on three different settings of increasing complexity, which are described below. We compared our results against several baselines: a uniformly random action selector, Sparse Cooperative Q-Learning (SCQL) (Kok & Vlassis, 2006), and Learning with Linear Rewards (LLR) (Gai et al., 2012).

SCQL is a multi-agent Q-Learning based algorithm that can leverage domain knowledge about agents' interdependencies to lower its sample requirements. SCQL was originally proposed in the context of multi-agent MDPs, but does apply to MAMABs. To allow for exploration we use both optimistic initialization and an ε-greedy policy, with the ε parameter linearly decreasing over time: $\varepsilon = 0.05 - 10^{-5}t$.

LLR is a UCB algorithm from the combinatorial bandit literature that applies most to MAMABs, as it assumes that the rewards are a linear combination of what we refer to as local reward functions. Contrary to MAUCE however, it computes upper confidence bounds on the local reward components separately, before summing them, rather than our vector-based formulation of Equations 1–3. LLR is parameterless, aside from the knowledge of which agents depend on each other.

In all experiments the rewards were normalized so that the maximum possible regret per timestep is one. The maximum possible reward was computed by directly solving non-stochastic versions of the problems with Variable Elimination (or brute-force enumeration). Reward normalization enables directly comparing the output results to see how each approach performs across different settings.

We now describe each of our problem settings: the 0101-Chain, which is simple but illustrates the fast learning properties of MAUCE; Gem Mining, which is real-world inspired and adapted from an established benchmark multi-objective coordination graph; and Wind Farm, a real-world coordination problem, in which we connect our learning problem to a state-of-the-art wind farm simulator.

All the code needed to run the experiments can be found at `https://bitbucket.org/Svalorzen/mauce-experiments/src/master/`

6.1. 0101-Chain

The 0101-Chain is a simple MAMAB, with a known optimal action. The problem consists of n agents, and $n-1$ local reward functions. Each local reward function $f^i(a_i, a_{i+1})$ is connected to the agent with the same index, i, and to $i+1$.

The optimal action in the 0101-Chain problem is $a_i = 0$ if i is even, and $a_i = 1$ is i odd. The reward tables for each local group are given in Table 1.

i *is even*	$a_{i+1}=0$	$a_{i+1}=1$
$a_i=0$	$\frac{f(\text{suc};0.75)}{n-1}$	$\frac{1}{n-1}$
$a_i=1$	$\frac{f(\text{suc};0.25)}{n-1}$	$\frac{f(\text{suc};0.9)}{n-1}$

Table 1. The reward table for 0101-Chain. n is the number of agents in the problem. $f(\text{suc};p)$ is a Bernoulli distribution with success probability p, i.e., $f(1;p)=p$ and $f(0;p)=1-p$. The table for odd agents is the same but transposed.

6.2. Gem Mining

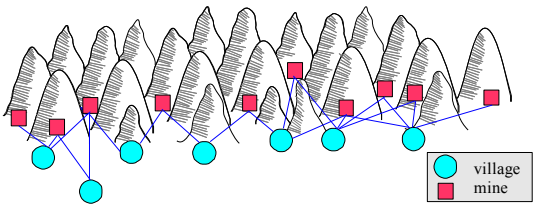


Figure 1. Gem Mining example. Each village represents an agent, while the mines represent the local reward functions.

Our Gem Mining problem is adapted from the Mining Day problem from (Roijers et al., 2015), which is a multi-objective coordination graph benchmark problem.

In Gem Mining, a mining company mines gems from a set of mines (local reward functions) located in the mountains (see Figure 1). The mine workers live in villages at the foot of the mountains. The company has one van in each village (agents) for transporting workers and must determine every morning to which mine each van should go (actions), but vans can only travel to nearby mines (graph connectivity). Workers are more efficient when there are more workers at a mine: the probability of finding a gem in a mine is $x \cdot 1.03^{w-1}$, where x is the base probability of finding a gem in a mine and w is the number of workers at the mine. To generate an instance with v villages (agents), we randomly assign 1-5 workers to each village and connect it to 2–4 mines. Each village is only connected to mines with a greater or equal index, i.e., if village i is connected to m mines, it is connected to mines i to $i+m-1$. The last village is connected to 4 mines and thus the number of mines is $v+3$.

6.3. Wind Farm

In our wind farm experiment, we used a state-of-the-art simulator (Van Dijk et al., 2016) to mimic the energy production of a series of wind turbines when exposed to a global incoming wind vector. In the real world, turbines can often be oriented at certain angles to maximize production. This is a non-trivial control task, as the turbulence caused by a turbine will negatively affect turbines downwind. The direction of this generated turbulence depends on the angle that the turbine has w.r.t. the incoming wind vector.

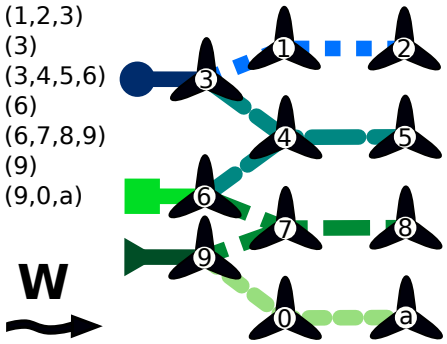


Figure 2. Wind farm setup. The incoming wind is denoted by an arrow. Each local group is denoted by a different color and line type. Groups are listed explicitly on the left. Note the three single-agent groups on the left to handle per-agent rewards.

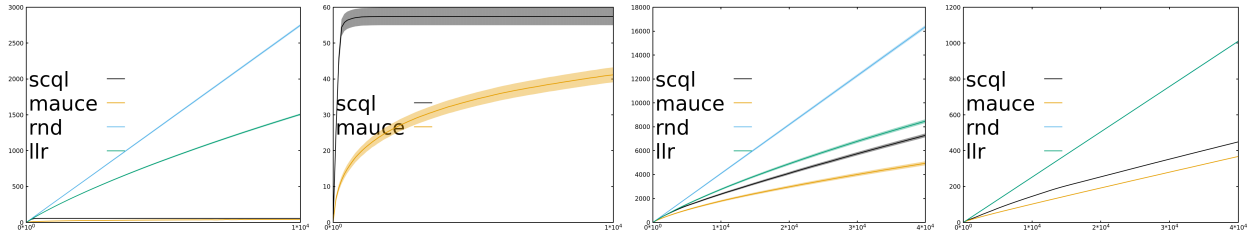
We setup our simulated wind farm using 11 turbines (see Figure 2). Each turbine has a choice between three different actions (angles) that it can turn to. The last 4 turbines downwind (2, 5, 8, and a) are set directly against the wind and are not controlled by agents, as they cannot generate turbulence that can impact power production. However, the remaining 7 turbines do influence the rest of the farm, and so must cooperate to maximize power production.

We vary the wind speed in the simulator at each timestep, following a truncated normal distribution with mean 8.1 m/s. The overall reward is normalized to a $[0,1]$ interval using the maximum possible overall reward at the highest wind strength and the minimum possible reward per turbine at the minimum wind strength. While this makes it impossible to compute the true regret, as choosing the optimal action does not result in a 0 regret in expectation, it avoids having to calculate the true expected reward for all actions in this scenario, which is non-trivial.

Differently from the previous experiments, rewards in this settings are obtained per-agent from the simulator rather than per-group. Thus, we use single-agent local groups to prevent dependencies between the reward functions of each group. The reward for agents in more than one group is given solely to their single-agent group, and none to the others (rather than splitting).

6.4. Results

We tested the performance of MAUCE on the higly structured 0101-Chain problem with 11 agents for 10,000 joint action executions, and compare its performance against random, SCQL and LLR. The results (Figure 3) indicate that both SCQL and MAUCE can learn effectively, far outclassing random joint action selection and LLR.



(a) 0101-Chain, All Algorithms (b) 0101-Ch., MAUCE&SCQL (c) Gem Mining (d) Wind Farm

Figure 3. Cumulative regret for all experiments as a function of the number of actions executed: a) 0101-Chain averaged over 100 runs, b) Same as 3(a), only SCQL and MAUCE, c) Gem Mining, averaged over 5 random setups, 100 runs per setup, and d) Wind Farm, 10 runs.

When comparing MAUCE and SCQL (Figure 3(b)), MAUCE achieves considerably less regret than SCQL. This is because MAUCE's exploration strategy is based on the aggregation of local exploration bounds, while SCQL uses an ε-greedy exploration strategy. On the other hand, SCQL does learn the optimal joint action quickly, thanks to optimistic initialization and this aggressive exploration strategy. We note that after a while, we decreased ε to 0, i.e., only exploit, making the regret graph a flat line from that point onward. We note that the annealing of ε needed to be fine-tuned. We thus conclude that MAUCE is an effective algorithm that can exploit the graphical structure, leading to superior performance for this highly-structured problem.

We then tested MAUCE against the other algorithms on randomly generated Gem Mining instances with 5 villages and 8 mines, to compare performances on a more challenging problem. Figure 3(c) represents the average regret over multiple different scenarios. We observe that, while SCQL and LLR are all able to achieve sublinear regret curves, MAUCE handles the exploration-exploitation tradeoffs best, resulting in the lowest regret over time.

Finally, to test the performance of MAUCE on a real-world problem, we run the algorithms on a Wind Farm instance (Figure 3(d)). Due to the high computational costs of running the simulator, we perform only ten runs. As explained before, the measure shown is not an exact form of regret, as the optimal action will not result in a 0 regret in expectation.

The MAUCE algorithm once again performs best, with less cumulative regret than both LLR and SCQL. The LLR algorithm also doesn't seem to achieve any significant learning with respect to the random policy. Note that MAUCE keeps learning and fine tuning this policy over the whole duration of the experiment, which allows it to increasingly achieve lower regret than SCQL. At timestep 10000, the difference between the two is 43.258, while at timestep 40000 it is 81.373. It is important to note that SCQL could probably be made to perform better by finely tuning the initialization values and epsilon updates, but this would take significant human time and repetitive trials. MAUCE can instead directly manage the exploration-exploitation trade-off by using its local bounds for each local joint action.

We thus conclude that MAUCE is an effective algorithm for trading off exploration versus exploitation in MAMABs, and has superior performance w.r.t. the alternative algorithms.

7. Conclusion

In this paper, we proposed the *multi-agent upper confidence exploration (MAUCE)* algorithm for *multi-agent multi-armed bandits (MAMABs)*. While learning, MAUCE leverages the graphical properties of the MAMAB by treating as separate objectives both exploration, expressed as a function of the sum over weighted inverse local counts, and exploitation, i.e., the sum over estimated mean local rewards. Via a subroutine, *upper confidence variable elimination (UCVE)*, that can handle these objectives, MAUCE selects the action that best balances exploration and exploitation according to the joint overall mean reward plus (upper confidence) exploration bound. We have proven a regret bound for MAUCE that is only linear in the number of agents, rather than exponential, as it would be if we were to flatten the MAMAB to a single-agent MAB. Furthermore, the regret bound is logarithmic in the number of arm pulls. We compared MAUCE empirically to state-of-the-art algorithms in multi-agent reinforcement learning and combinatorial bandits, and have shown that MAUCE achieves much lower empirical regret than these approaches.

We note that the range parameters r^e_{max} for MAUCE, which represent the difference between the maximum and minimum possible reward for each local joint action, can be difficult to guess in advance when the problem is not exactly known, as in the Wind Farm experiments. One way to mitigate this, could be to estimate them from the coordination graph of expected mean rewards learnt while running the algorithm, rather than running preliminary experiments as we did for the Wind Farm. We will test this in future work. Furthermore, we aim to build on MAUCE to achieve quality guarantees for reinforcement learning in multi-agent MDPs.

Acknowledgements

The first author was supported by Flanders Innovation & Entrepreneurship (VLAIO), SBO project 140047: Stable MultI-agent LEarnIng for neTworks (SMILE-IT), second author was supported by an FWO PhD grant (Fonds Wetenschappelijk Onderzoek - Vlaanderen), third author was a Postdoctoral Fellow with the FWO (grant #12J0617N).

References

Agrawal, S. and Goyal, N. Analysis of Thompson sampling for the multi-armed bandit problem. In *COLT*, pp. 39–1, 2012.

Audibert, J.-Y., Bubeck, S., and Lugosi, G. Minimax policies for combinatorial prediction games. In *COLT*, volume 19, pp. 107–132, 2011.

Auer, P. and Ortner, R. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Boutilier, C. Planning, learning and coordination in multiagent decision processes. In *TARK 1996: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pp. 195–210, 1996.

Bubeck, S. and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint:1204.5721*, 2012.

Cesa-Bianchi, N. and Lugosi, G. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

Chen, C.-P. and Qi, F. The best lower and upper bounds of harmonic sequence. *RGMIA research report collection*, 6 (2), 2003.

Chen, W., Wang, Y., and Yuan, Y. Combinatorial multi-armed bandit: General framework, results and applications. In *Proceedings of the 30th international conference on machine learning*, pp. 151–159, 2013.

Claes, D., Oliehoek, F., Baier, H., and Tuyls, K. Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 492–500. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

De Hauwere, Y.-M., Vrancx, P., and Nowé, A. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pp. 715–722, 2010.

Gai, Y., Krishnamachari, B., and Jain, R. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5): 1466–1478, 2012.

Guestrin, C., Koller, D., and Parr, R. Multiagent planning with factored MDPs. In *NIPS 2002: Advances in Neural Information Processing Systems 15*, pp. 1523–1530, 2002.

Kok, J. and Vlassis, N. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, December 2006.

Kok, J. R., Spaan, M. T. J., Vlassis, N., et al. Multi-robot decision making using coordination graphs. In *Proceedings of the 11th International Conference on Advanced Robotics, ICAR*, volume 3, pp. 1124–1129, 2003.

Roijers, D., Whiteson, S., and Oliehoek, F. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 2015.

Roijers, D. M. *Multi-Objective Decision-Theoretic Planning*. PhD thesis, University of Amsterdam, 2016.

Roijers, D. M. and Whiteson, S. Multi-objective decision making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 11(1):1–129, 2017.

Rollón, E. and Larrosa, J. Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12: 307–328, 2006.

Scharpff, J., Roijers, D. M., Oliehoek, F. A., Spaan, M., and de Weerdt, M. M. Solving transition-independent multi-agent MDPs with sparse interactions. In *AAAI16: Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Van Dijk, M. T., Wingerden, J. W., Ashuri, T., Li, Y., and Rotea, M. Yaw-misalignment and its impact on wind turbine loads and wind farm power output. *Journal of Physics: Conference Series*, 753(6), 2016.

Wiering, M. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151–1158, 2000.