Multi-agent RMax for Multi-Agent Multi-Armed Bandits

Eugenio Bargiacchi
Artificial Intelligence Lab
Vrije Universiteit Brussel
Belgium
svalorzen@gmail.com

Raphael Avalos
Artificial Intelligence Lab
Vrije Universiteit Brussel
Belgium
raphael.avalos@vub.be

Timothy Verstraeten
Artificial Intelligence Lab
Vrije Universiteit Brussel
Belgium
timothy.verstraeten@vub.be

Pieter Libin
Artificial Intelligence Lab
Vrije Universiteit Brussel
Belgium
pieter.libin@vub.be

Ann Nowé
Artificial Intelligence Lab
Vrije Universiteit Brussel
Belgium
ann.nowe@vub.be

Diederik M. Roijers
Vrije Universiteit Brussel
HU Univ. of Appl. Sci. Utrecht
Belgium / The Netherlands
diederik.roijers@vub.be

ABSTRACT

In this paper, we provide PAC bounds for best-arm identification in multi-agent multi-armed bandits (MAMABs), via an algorithm we call multi-agent RMax (MARMax). In a MAMAB, the reward structure is expressed as a coordination graph, i.e., the total team reward is a sum over local reward functions that are conditioned on the actions of a subset of the agents. Assuming that these local rewards functions are bounded, we derive a PAC(δ, ε) learning algorithm that achieves an accuracy of ε relative to the maximum team reward, with a number of required samples at most linear in the size of the largest reward reward function with probability $1-\delta$. Furthermore, we show that in practice MARMax performs much better than this theoretical upper bound on the sample complexity by an order of magnitude. We further improve on the empirical result by tempering the optimism used in MARMax, resulting in a new algorithm that we call MAVMax. We show empirically that MAVMax further cuts down the number of required samples by around 30% w.r.t. MARMax.

KEYWORDS

PAC bounds, multi-agent systems, multi-armed bandits, best-arm identification, RMax

1 INTRODUCTION

Imagine a wind-farm with multiple turbines [17]. The orientation with respect to the incoming wind direction – the yaw – determines the power output of a turbine. At the same time, a turbine's orientation also affects the intensity of its wake, which reduces the power produced by the wind turbines downwind of it. We thus need to find the right *joint* policy for the entire wind-farm. This is an example of a multi-agent coordination problem. Furthermore, because the dynamics of the environment are initially unknown, we need to learn the optimal joint policy through (multi-agent) reinforcement learning.

The example given above can be modelled with a *multi-agent multi-armed bandit (MAMAB)* [2]. In these settings, agents (e.g., turbines) can influence local rewards (e.g., the power output of a single turbine), with their local actions. However, the actions of an agent only impact a subset of the local rewards; the setting is loosely coupled [6]. Exploiting this property of being loosely coupled is key to keeping such problems tractable. Both for selecting a joint action [6], and, as we show in this paper, for learning.

Previous approaches for learning in MAMABs have focused on minimising regret [2, 18]. That is, the learning happens in operation, always continues, and we aim to minimise the amount of reward missed due to exploring. However, this is not always how learning can be applied in practice. Many real-world problems require the learning phase to be limited in time, until a level of confidence is reached. When this confidence level is reached, the agents should stop learning, and the policy can be deployed for long-term execution in practice. This is called the *best-arm identification setting* in the bandit literature.

In this paper, we contribute the *multi-agent RMax (MARMax)* and *multi-agent VMAX (MAVMax)* algorithms for best (joint) arm identification in multi-agent multi-armed bandits. We show that to reach a confidence level of $1-\varepsilon$, with a probability of $1-\delta$, i.e., a $PAC(\varepsilon, \delta)$ guarantee, we can bound the number of joint action executions. When the local reward functions are all bounded within the same interval, this number of joint action executions does not depend on the number of local reward functions, nor the size of the full joint action space, but only on the maximum number of *local* joint actions for a local reward function. This renders the exploration process in MAMABs much more efficient than by flattening the problem to single-agent bandits. To the best of our knowledge these are the first algorithms that provide PAC-guarantees for best-arm identification in MAMABs.

We evaluate MARMax and MAVMax on a set of benchmarks from the MAMAB literature, and show that while our guarantees already provide a probability bound on the number of samples until the desired confidence level is reached, in practice our algorithms require significantly fewer samples. Especially in MAVMax, which uses more tempered optimism in the face of uncertainty compared to MARMax, we are able to significantly decrease the number of samples, i.e., the number of joint action executions, that is necessary to reach the desired confidence level compared to the theoretical upper bound on this number of joint action executions.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

2 BACKGROUND

We model our problem as a multi-agent multi-armed bandit, which is a repeated fully cooperative multi-agent game with a factored team reward function:

Definition 2.1. A multi-agent multi-armed bandit (MAMAB) is a tuple $\langle \mathcal{A}, \mathcal{D}, F \rangle$ where

- $\mathcal{D}$ is the set of N enumerated agents,
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ is a set of joint actions, which is the Cartesian product of the sets of individual actions, $\mathcal{A}_i$, for each agent in $\mathcal{D}$,
- $F(\mathbf{a})$, 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 $f^e(\mathbf{a}^e) \in \left[0, r^e_{\max}\right]$. A local function f^e only depends on the local joint action $\mathbf{a}^e$ of the subset $\mathcal{D}^e$ of agents.

In this work we tackle the problem of *best arm identification*. In particular, we focus on the goal of recommending a joint arm that, with probability $1-\delta$, has an expected mean within a factor ε of the optimal one. 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 $F(\mathbf{a})$: $\mu_{\mathbf{a}} = \sum_{e=1}^{\rho} \mu^e(\mathbf{a}^e)$. For simplicity, we refer to an agent i by its index. We note that this is a fully cooperative setting, and that the agents can coordinate to decide on a joint action.

One important feature of MAMABs is that the factorization of $F(\mathbf{a})$ allows us to model the problem as a *coordination graph (CoG)* [6, 7]. Thus, we can efficiently extract the optimal joint action once we have learned the means of all local actions $\mu^e(\mathbf{a}^e)$. A naive way to do this would be to *'flatten'* the CoG, i.e., enumerate all joint actions, compute their associated mean reward, and then maximize. However, this is typically computationally 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 $A = 2^N$. Instead, we can leverage the properties of the CoG, and extract the optimal reward and associated actions using algorithms like *variable elimination (VE)* [6].

In VE, agents are eliminated from the CoG sequentially, one by one, solving the maximization problem as a series of *local subproblems*. 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 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.

Learning the values of the mean joint rewards incurs similar problems. 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. Instead, we wish to devise a strategy that allows us to gather the maximum amount of information with each individual joint arm pull of the MAMAB.

Algorithm 1 MARMax

Input: An MAMAB with a factorized reward function, $F(\mathbf{a}) = \sum_{e=1}^{\rho} f^e(\mathbf{a}^e)$, number of samples per local reward estimator, m.
Output: The estimated best joint action, $\mathbf{a}_t$.
Initialize $\hat{\mu}^e(\mathbf{a}^e)$ to r^e_{max}
Initialize $q^e(\mathbf{a}^e)$ and $n^e(\mathbf{a}^e)$ to 0.
Initialize $v^e(\mathbf{a}^e)$ to 1.
$\mathbf{a}_t \leftarrow$ a random joint action
while $\sum_{e=1}^{\rho} v^e(\mathbf{a}^e_t) > 0$ **do**
  Pull $\mathbf{a}_t$ and receive local rewards $r^e_t(\mathbf{a}^e_t)$
  for all $r^e_t(\mathbf{a}^e_t)$ **do**
    $q^e(\mathbf{a}^e_t) \leftarrow q^e(\mathbf{a}^e_t) + r^e_t(\mathbf{a}^e_t)$
    $n^e(\mathbf{a}^e_t) \leftarrow n^e(\mathbf{a}^e_t) + 1$
    if $n^e(\mathbf{a}^e_t) \geq m$ **then**
      $\hat{\mu}^e(\mathbf{a}^e_t) \leftarrow \frac{q^e(\mathbf{a}^e_t)}{n^e(\mathbf{a}^e_t)}$
      $v^e(\mathbf{a}^e_t) \leftarrow 0$
    end if
  end for
  $\mathbf{a}_t = \arg\max_{\mathbf{a}} \sum_{e=1}^{\rho} \hat{\mu}^e(\mathbf{a}^e)$ //Using Variable Elimination
end while
return $\mathbf{a}_t$

3 MULTI-AGENT RMAX

In a multi-agent multi-armed bandit (MOMAB), every timestep ρ local reward functions are sampled. To determine a PAC-bound, it is key to exploit that these local rewards are independently distributed and drawn.

Inspired by RMax [3, 15], we first propose *multi-agent RMax (MARMax)* as described in Algorithm 1. Following an RMax-inspired strategy, the estimator for every possible local reward, $\hat{\mu}^e(\mathbf{a}^e)$, is optimistically initialised with r^e_{max}, the upper bound of the associated random variable. Until the local joint action is not sampled at least m times, we consider that its reward is still unknown and therefore keep it to its maximum value r^e_{max}. After MARMax has performed a local joint action at least m times, its reward becomes *known*, and its estimator is replaced by the maximum likelihood estimator with respect to the $\geq m$ samples. In Algorithm 1 the known status is maintained by $v^e(\mathbf{a}^e)$ which is equal to 1 if it is unknown and 0 otherwise. The number of samples m is a parameter of MARMax that we will determine to obtain theoretical guarantees.

MARMax calculates the best joint action $\mathbf{a}_t$ on the basis of the estimators μ^e using variable elimination [8]. When all the components of this joint action $\mathbf{a}_t, a^e_t$, are known, i.e., $\sum_{e=1}^{\rho} v^e(\mathbf{a}^e_t) = 0$, MARMax terminates and recommends $\mathbf{a}_t$ as its output. We note that when this happens, the reward of all the other joint actions, $\mathbf{a}_{alt}$ consist of either all known components as well, or are an upper bound on the m-sample maximum likelihood estimate for $\hat{\mu}(\mathbf{a}_{alt}) = \sum_{e=1}^{\rho} \hat{\mu}^e(\mathbf{a}^e_{alt})$, that would be computed if MARMax would continue to run until all components become known. Therefore, we can base our PAC

bound off the confidence interval for the recommended joint action, $\mathbf{a}_t$.

We aim for a PAC-bound where we reach at least a factor $1-\varepsilon$ of the global upper bound on the team reward, with a probability of at least $1-\delta$. Recall that each local joint arm is a random variable X^e bounded the interval $\left[0, r^e_{max}\right]$. After any given set of full joint action pulls, for each group $e \in \rho$ we obtain an ensemble of m^e samples, $X^e_{[1,m^e]}$, to estimate $\hat{\mu}^e(\mathbf{a}_t)$. Then, considering the Hoeffding bound for the scaled random variables $Z^e = X^e/m^e$ we obtain the following:

$$P\left(\left|\sum_{e=1}^{\rho}\sum_{i=1}^{m^e} Z^e_i - \sum_{e=1}^{\rho}\sum_{i=1}^{m^e}\frac{\mu^e}{m^e}\right| > t\right) \leq 2exp\left(\frac{-2t^2}{\sum_{e=1}^{\rho}\sum_{i=1}^{m^e}(r^e_{max}/m^e)^2}\right) \tag{1}$$

Given that, with $m = \min_e m^e$, we have:

$$\sum_{e=1}^{\rho}\sum_{i=1}^{m^e} Z^e_i - \sum_{e=1}^{\rho}\sum_{i=1}^{m^e}\frac{\mu^e}{m^e} = \hat{\mu}(\mathbf{a}_t) - \mu(\mathbf{a}_t) \tag{2}$$

$$\sum_{e=1}^{\rho}\sum_{i=1}^{m^e}\left(\frac{r^e_{max}}{m^e}\right)^2 = \sum_{e=1}^{\rho}\frac{(r^e_{max})^2}{m^e} \leq \frac{\sum_{e=1}^{\rho}(r^e_{max})^2}{m}, \tag{3}$$

and that we would like to express our bound in terms of the maximum possible reward $\sum_{e=1}^{\rho} r^e_{max}$, we can state the bound as:

$$P\left(|\hat{\mu}(\mathbf{a}_t) - \mu(\mathbf{a}_t)| > \varepsilon\sum_{e=1}^{\rho} r^e_{max}\right) \leq 2exp\left(-2\frac{\varepsilon^2(\sum_{e=1}^{\rho} r^e_{max})^2 m}{\sum_{e=1}^{\rho}(r^e_{max})^2}\right). \tag{4}$$

This leads, for each component, to a required number of samples per arm of:

$$m \geq \frac{\sum_{e=1}^{\rho} {r^e_{max}}^2}{2\varepsilon^2(\sum_{e=1}^{\rho} r^e_{max})^2}\ln(2/\delta), \tag{5}$$

In order to reach m samples for each local payoff estimate $\hat{\mu}^e(\mathbf{a}^e)$, we need to consider the worst case number of full joint action arm pulls, n, to obtain these samples. Assuming we can only pull 1 component that is still unknown at every time-step, we would need ρA^c times m pulls in total. This would lead to:

$$n = m\rho A^c \geq \frac{\sum_{e=1}^{\rho} {r^e_{max}}^2}{2\varepsilon^2(\sum_{e=1}^{\rho} r^e_{max})^2}A^c\rho\ln(2/\delta), \tag{6}$$

where A is the maximal size of the action space of an agent $A = max_i|\mathcal{A}_i|$, and c is the maximal number of agents in scope for a single local reward function, $f^e(\mathbf{a}^e)$. This leads us to the following theorem:

Theorem 3.1. *MARMax is a PAC(δ,ε)-learning algorithm with for a MAMAB with a maximum number of A^c fields per local reward function and ρ local reward functions in total, when parameterised with*

$$m = \left\lceil \frac{\sum_{e=1}^{\rho} {r^e_{max}}^2}{2\varepsilon^2(\sum_{e=1}^{\rho} r^e_{max})^2}\ln(2/\delta)\right\rceil.$$

The number of required full joint action executions to reach the bound is:

$$n = O\left(\frac{\sum_{e=1}^{\rho} {r^e_{max}}^2}{2\varepsilon^2(\sum_{e=1}^{\rho} r^e_{max})^2}A^c\rho\ln(2/\delta)\right).$$

When all the r^e_{max} have the same value, r_{max}, Equation 4 simplifies to:

$$P(|\hat{\mu}(\mathbf{a}_t) - \mu(\mathbf{a}_t)| > \varepsilon\rho r_{max}) \leq 2exp\left(-2\varepsilon^2\rho m\right), \tag{7}$$

which in turn leads to:

$$m \geq \frac{\ln 2/\delta}{2\rho\varepsilon^2}, \tag{8}$$

and,

$$n \geq \frac{A^c\ln 2/\delta}{2\varepsilon^2}, \tag{9}$$

i.e., the minimum number of required arm pulls only depends on the number of fields in a local reward function, but not on the number of local reward functions.

Lemma 3.2. *When all local reward functions in a MAMAB, M, are bounded random variables in the same interval $[0, r_{max}]$, with a maximum number of A^c fields per local reward function and ρ local reward functions in total, MARMax is a PAC(δ,ε)-learning algorithm with for M when parameterised with*

$$m = \left\lceil \frac{\ln 2/\delta}{2\rho\varepsilon^2}\right\rceil.$$

The number of required full joint action executions to reach the bound is:

$$n = O\left(\frac{A^c\ln 2/\delta}{2\varepsilon^2}\right).$$

In Algorithm 1, the reward of local actions that was not been sampled at least m times is estimated as r^e_{max}. This ensures that our estimator is an upper bound of the excepted reward, leading to selecting that local action if it could be part of the optimal joint

Algorithm 2 MAVMax

Input: An MAMAB with a factorized reward function, $F(\mathbf{a}) = \sum_{e=1}^{\rho} f^e(\mathbf{a}^e)$, number of samples per local reward estimator, m.
Output: The estimated best joint action, $\mathbf{a}_t$.
Initialize $\hat{\mu}^e(\mathbf{a}^e)$ to r^e_{max}
Initialize $q^e(\mathbf{a}^e)$ and $n^e(\mathbf{a}^e)$ to 0.
Initialize $v^e(\mathbf{a}^e)$ to 1.
$\mathbf{a}_t \leftarrow$ a random joint action
while $\sum_{e=1}^{\rho} v^e(\mathbf{a}^e_t) > 0$ **do**
  Pull $\mathbf{a}_t$ and receive local rewards $r^e_t(\mathbf{a}^e_t)$
  for all $r^e_t(\mathbf{a}^e_t)$ **do**
    $q^e(\mathbf{a}^e_t) \leftarrow q^e(\mathbf{a}^e_t) + r^e_t(\mathbf{a}^e_t)$
    $n^e(\mathbf{a}^e_t) \leftarrow n^e(\mathbf{a}^e_t) + 1$
    if $n^e(\mathbf{a}^e_t) < m$ **then**
      {*Here is the difference with MARMAX*}
      $\hat{\mu}^e(\mathbf{a}^e_t) \leftarrow \frac{q^e(\mathbf{a}^e_t)+(m-n^e(\mathbf{a}^e_t))r^e_{max}}{m}$
    else
      $\hat{\mu}^e(\mathbf{a}^e_t) \leftarrow \frac{q^e(\mathbf{a}^e_t)}{n^e(\mathbf{a}^e_t)}$
      $v^e(\mathbf{a}^e_t) \leftarrow 0$
    end if
  end for
  $\mathbf{a}_t = \arg\max_{\mathbf{a}} \sum_{e=1}^{\rho} \hat{\mu}^e(\mathbf{a}^e)$ //Using VE
end while
return $\mathbf{a}_t$

i is even	$a_{i+1} = 0$	$a_{i+1} = 1$
$a_i = 0$	$f(\text{suc}; 0.75)$	1.0
$a_i = 1$	$f(\text{suc}; 0.25)$	$f(\text{suc}; 0.9)$

Table 1: The reward table for 0101-Chain. $f(\text{suc}; p)$ 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.

action. This upper-bound is however naive as it does not take into account the $k < m$ local rewards already sampled. Since for all $k \in [1, m-1]$ and with the random variables X_i bounded by r_{max}

$$\sum_{i=1}^{m} X_{e,i} \leq \sum_{i=1}^{k} X_{e,i} + (n-k) r^e_{max} \tag{10}$$

Based on this observation we extend Algorithm 1 by updating progressively as in VMAX [10]. Such tempered optimism does not affect the proof in any way, as long as the estimators, $\hat{\mu}^e(\mathbf{a}^e)$, for the m-sample-based averages remain an upper bound. This leads to Algorithm 2, which we call *multi-agent VMax (MAVMax)*. Updating the estimators progressively leads to a tighter upper bound which can be expected to speed-up convergence as sub-optimal actions could be discarded earlier.

4 EXPERIMENTS

In this section we empirically evaluate our proposed methods, MARMax and MAVMax, on a set of MAMAB environments.

4.1 0101-Chain

The 0101-Chain environment [2] consists of a number of local reward functions that solely depend on successive pairs of agents, i.e. agents 1 and 2, 2 and 3, 3 and 4, and so on. All agents have binary actions.

Local rewards in this environment are drawn from independent Bernoulli distributions, with a probability that depends on the local joint action. The distributions corresponding to each local action are shown in Table 1. These rewards result in a problem where the optimal action can be trivially determined even for large number of agents: even-indexed agents need to take action 0, while agents with odd indices must take action 1.

Due to how our implementation of MARMax selects the joint action to pull, however, this very structured pre-determined optimal action was always discovered after $2 \cdot m$ timesteps. To make the problem somewhat harder, we additionally tested on a randomized version of the Chain. In particular, when generating the environment, we sample a full joint action uniformly and set it as the optimal action. Then, for each pair of agents, the rewards table shown in Table 1 is either used as-is, or transposed, or its columns swapped, or its rows swapped, so that the 1.0 best entry corresponds to the pre-determined optimal local joint action of those agents. For example, if agent 1 and agent 2 optimal actions are $(0, 0)$, then the reward table for their local reward is the same as Table 1 but with the columns swapped. Note that here the information on whether an agent's index is even or odd is not used.

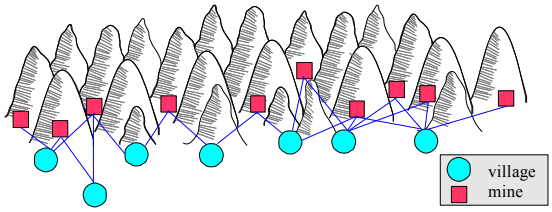

Figure 1: Gem Mining example. Each village represents an agent, while the mines represent the local reward functions.

4.2 Gem Mining

The Gem Mining environment [2] is adapted from the Mining Day problem from [11], which is a multi-objective coordination graph benchmark problem.

This environment consists of a set of villages (agents) and mines (local reward functions). At each timestep, each village can send its workers to a mine that is connected to it. Each mine is associated with a productivity value $x \in (0, 1)$, and samples a reward from a Bernoulli distribution with a probability parameter $p = \min(x \cdot 1.03^{w-1}, 1.)$, where w is the total number of workers sent to it. However, if w is 0 then the reward is always 0 as well.

4.3 Wind Farm

The Wind Farm environment [2, 18] uses a state-of-the-art simulator [16] to evaluate the energy production of a wind-farm subject to predetermined fixed wind conditions. In a wind-farm, turbines can be oriented at certain angles to maximize energy production. At the same time, each turbine's orientation directly affects the production of turbines downwind through the wake effect. The goal is to determine the best optimal joint configuration to maximize energy production of the whole wind-farm.

We use the same setup as [2, 18]. Specifically, the simulated wind-farm has 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 always oriented directly against the wind and are not controlled by agents, as they cannot

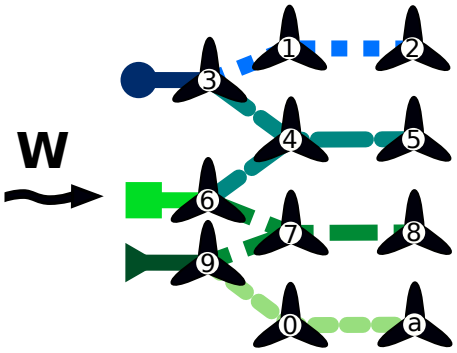

Figure 2: Wind farm setup. The incoming wind is denoted by an arrow. Each local group is denoted by a different color and line type.

N	ρ	A_c	δ	ε	m	n	Avg n	ε-Opt
MARMax								
10	9	4	0.05	0.05	82	2952	328.00	1.00
			0.10	0.10	17	612	68.80	1.00
20	19	4	0.05	0.05	39	2964	156.15	1.00
			0.10	0.10	8	608	33.50	1.00
50	49	4	0.05	0.05	16	3136	65.09	1.00
			0.10	0.10	4	784	21.11	1.00
MAVMax								
10	9	4	0.05	0.05	82	2952	97.71	1.00
			0.10	0.10	17	612	29.90	1.00
20	19	4	0.05	0.05	39	2964	55.00	1.00
			0.10	0.10	8	608	19.81	1.00
50	49	4	0.05	0.05	16	3136	32.35	1.00
			0.10	0.10	4	784	16.91	1.00

Table 2: Results for the 0101-Chain environment, with randomized optimal actions, for different combinations of number of agents, δ and ε.

N	ρ	A_c	δ	ε	m	n	Avg n	ε-Opt
MARMax								
7	10	144	0.05	0.05	74	106560	10656.00	1.00
			0.10	0.10	15	21600	2160.00	1.00
9	12	36	0.05	0.05	62	26784	2232.00	1.00
			0.10	0.10	13	5616	468.00	1.00
13	16	128	0.05	0.05	47	96256	6016.00	1.00
			0.10	0.10	10	20480	1280.00	1.00
MAVMax								
7	10	144	0.05	0.05	74	106560	8598.36	1.00
			0.10	0.10	15	21600	1520.17	1.00
9	12	36	0.05	0.05	62	26784	1855.89	1.00
			0.10	0.10	13	5616	377.12	1.00
13	16	128	0.05	0.05	47	96256	5039.90	1.00
			0.10	0.10	10	20480	898.44	1.00

Table 3: Results for the Mines environment, for different combinations of villages, mines, δ and ε.

generate turbulence that can impact power production. Thus, the problem requires joint cooperation of 7 agents with 3 actions each.

4.4 Discussion

For each environment we show results over a range of different choices of δ and ε parameters, as well as varying the number of agents N and local reward functions. Results are an average over 1000 independent runs for the 0101-Chain and Mines environments, and 100 independent runs for the Wind environment.

For each choice of parameters we show the corresponding value of m, the upper bound value of n, the empirical average value of n (the average number of timesteps needed to recommend an arm), and the percentage of runs which achieved ε-optimality, i.e. where regret was less than $\varepsilon \sum_{e=1}^{\rho} r^e_{max}$.

All environments use $r^e_{max} = 1$ for all local reward functions. For the 0101-Chain and Mines environments this is because rewards are always sampled from Bernoulli distributions. For the Wind Farm environment, we empirically evaluated the individual minimum and maximum production of each turbine across all possible joint actions, so that we could normalize each to a reward between 0 and 1.

In all environments, MARMax and MAVMax always fully respect the theoretical bounds, recommending ε-optimal arms nearly all the time. Even when a ε-optimal arm is not recommended, in expectation the δ bound is never broken.

However, the arms recommended by both MARMax and MAVMax were most of the time not the optimal joint actions. This can be explained as in a factored setting the exponential number of full joint arms have expected rewards that are more tightly packed than in a more traditional flat bandit environment. This is because for each full joint action there are many similar ones that differ from it by only a single local joint action. In the same way, there will be many joint arms with expected rewards close to the optimal one. Thus, uniquely identifying the optimal full joint action can require a significant number of pulls in our setting. As MARMax and MAVMax are optimized to find an ε-optimal action, they do not need to expend additional timesteps trying to determine the true optimal joint action.

N	ρ	A_c	δ	ε	m	n	Avg n	ε-Opt
MARMax								
7	7	27	0.05	0.05	106	20034	1161.42	0.99
			0.10	0.10	22	4158	237.43	1.00
MAVMax								
7	7	27	0.05	0.05	106	20034	424.21	1.00
			0.10	0.10	22	4158	95.39	1.00

Table 4: Results for the Wind environment, for different combinations of δ and ε.

We additionally note that empirically the value of n, i.e. the number of arm pulls before an arm is recommended, is generally in the order of mA_c rather than $m\rho A_c$ as in Equation 6. While in Equation 6 we considered the worst case scenario of being able to pull only a single unknown local arm at a time, in practice each pull of a joint action samples all ρ local actions concurrently (with hopefully most of them unknown). Given that A_c is the size of the largest local reward function, once it is fully explored (after mA_c timesteps), the others will generally be explored as well.

Additionally, MAVMax is able to recommend an arm much faster than MARMax, with *Avg n* between 2/3 and 3/4 of MARMax. In additional experiments we performed with higher δ and ε values (and thus lower m and looser bounds), MAVMAx can sometimes fail to recommend a ε-optimal arm, but still never breaks the theoretical bounds.

5 RELATED WORK

There have recently been advances in multi-agent multi-armed bandits regret minimization. The MAUCE algorithm [2] uses techniques from the multi-objective literature to select the optimal joint arm to take at each timestep using an UCB mechanism to ensure consistent exploration. MATS [18] uses Thompson sampling on

individual local actions, together with variable elimination, to consistently select full joint actions following the posterior probability that they are optimal.

The best-arm identification literature is generally divided into two distinct sub-fields: fixed budget and fixed confidence. Fixed budget algorithms are provided a finite number of timesteps to act in as input, and must provide the best possible recommendation within that time. On the other hand, fixed confidence settings are provided as input a risk probability, and the goal is to ensure that the probability of recommending the wrong arm is lower than this. While sharing similarities, the two settings usually have different theoretical frameworks and bounds proven in one setting do not apply to the other.

Some algorithms can work in an anytime fashion, and can be used in both settings, provided they use appropriate stopping rules. TTTS [13] uses Thompson sampling to recommend the most likely best arm with some probability p, and samples again to recommend the most likely contender with probability $1 - p$. The Adaptive UCB-E [1] uses a UCB mechanism to handle exploration without requiring tuning, does not currently provide any guarantees.

In the fixed budget setting, the SR algorithm [1] uses a fixed budget by splitting the arm pulling process into several phases. After each phase an arm is removed from the considered set and is not pulled afterwards. The last arm to remain is recommended as the best.

For fixed confidence, the RMax algorithm [3] inspired this work. RMax learns in stochastic games by marking interactions as known after a set number of experience points have been collected. The T3C algorithm [14] uses a deterministic scoring function based on KL divergence to avoid the computational costs of TTTS when the posteriors have significantly converged.

6 CONCLUSION

We have presented MARMax and MAVMax, two novel algorithms for best-arm identification in multi-armed multi-agent bandits. The algorithms exploit the structured representation of the joint reward function, which allows them to efficiently learn and identify a ε-optimal joint action. We provide a PAC-bound for MARMax, proving that the sample complexity of the algorithm is linear in the size of the largest local reward function, rather than exponential in the number of agents. We tested both algorithms empirically in a variety of settings taken from the MAMAB literature, and show that the bounds hold in all cases.

In future work, we aim to strengthen the bounds in multi-agent multi-armed bandits. Firstly, as we observe in the paper, the number of joint action executions we actually need to obtain m samples for each local reward function is much smaller than the theoretical upper bound we establish, for MARMax and even more so for MAVMax. As such, we believe there is space to improve the bound for our algorithms. In addition, we aim to adapt other best-arm identification algorithms for single-agent multi-armed bandits [1, 5, 13] to the multi-agent setting, in the hope of establishing a tighter bound. Finally, we aim to extend our results to sequential [4] and multi-objective [9, 12] multi-agent settings.

ACKNOWLEDGMENTS

The first and second authors are supported by the Research Foundation – Flanders (FWO), under grant numbers 1SA2820N & 11F5721N. P.J.K.L. gratefully acknowledges support from the Fonds voor Wetenschappelijk Onderzoek (FWO) via postdoctoral fellowship 1242021N, and the Research council of the Vrije Universiteit Brussel (OZR-VUB) via grant number OZR3863BOF. This research was supported by funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

REFERENCES

[1] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. 2010. Best arm identification in multi-armed bandits.. In *COLT*. Citeseer, 41–53.
[2] Eugenio Bargiacchi, Timothy Verstraeten, Diederik Roijers, Ann Nowé, and Hado Hasselt. 2018. Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems. In *International conference on machine learning*. PMLR, 482–490.
[3] Ronen I Brafman and Moshe Tennenholtz. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3, Oct (2002), 213–231.
[4] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1* (2010), 183–221.
[5] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. 2012. Best arm identification: A unified approach to fixed budget and fixed confidence. *Advances in Neural Information Processing Systems* 25 (2012).
[6] C.E. Guestrin, D. Koller, and R. Parr. 2002. Multiagent Planning with Factored MDPs. In *NIPS 2002: Advances in Neural Information Processing Systems 15*. 1523–1530.
[7] J.R. Kok and N. Vlassis. 2006. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. *Journal of Machine Learning Research* 7 (Dec. 2006), 1789–1828.
[8] Jelle R Kok and Nikos Vlassis. 2006. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup 2005: Robot Soccer World Cup IX*. 1–12.
[9] Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. 2020. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 1–52.
[10] Karun Rao, Shimon Whiteson, et al. 2012. V-MAX: tempered optimism for better PAC reinforcement learning.. In *AAMAS*. 375–382.
[11] D.M. Roijers, S. Whiteson, and F. Oliehoek. 2015. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research* 52 (2015), 399–443.
[12] D. M. Roijers and S. Whiteson. 2017. Multi-objective decision making. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 11, 1 (2017), 1–129.
[13] Daniel Russo. 2016. Simple bayesian algorithms for best arm identification. In *Conference on Learning Theory*. PMLR, 1417–1418.
[14] Xuedong Shang, Rianne Heide, Pierre Menard, Emilie Kaufmann, and Michal Valko. 2020. Fixed-confidence guarantees for Bayesian best-arm identification. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1823–1832.
[15] Alexander L Strehl, Lihong Li, and Michael L Littman. 2009. Reinforcement Learning in Finite MDPs: PAC Analysis. *Journal of Machine Learning Research* 10, 11 (2009).
[16] M. T. Van Dijk, J. W. Wingerden, T. Ashuri, Y. Li, and M. Rotea. 2016. Yaw-Misalignment and its Impact on Wind Turbine Loads and Wind Farm Power Output. *Journal of Physics: Conference Series* 753, 6 (2016).
[17] Timothy Verstraeten. 2021. *A Multi-Agent Reinforcement Learning Approach to Wind Farm Control*. Ph.D. Dissertation. Vrije Universiteit Brussel.
[18] Timothy Verstraeten, Eugenio Bargiacchi, Pieter JK Libin, Jan Helsen, Diederik M Roijers, and Ann Nowé. 2020. Multi-agent Thompson sampling for bandit applications with sparse neighbourhood structures. *Scientific reports* 10, 1 (2020), 1–13.