Multi-agent sequential learning via gradient ascent credit assignment

Oussama Sabri, Luc Lehéricy, Alexandre Muzy

MULTI-AGENT SEQUENTIAL LEARNING VIA GRADIENT ASCENT CREDIT ASSIGNMENT

Oussama Sabri
University Medical Center
Hamburg-Eppendorf (UKE)
Hamburg, Germany
o.sabri@uke.de

Luc Lehéricy
Laboratoire JAD, CNRS
Université Côte d'Azur (UCA)
Nice, France
luc.leherichy@univ-cotedazur.fr

Alexandre Muzy
Laboratoire I3S, CNRS
Université Côte d'Azur (UCA)
Sophia Antipolis, France
alexandre.muzy@univ-cotedazur.fr

ABSTRACT

We consider a multi-agent learning problem with two layers: first, split up the tasks among the agents, and then have them solve their allocated task. A set of algorithms is proposed based on a policy gradient method with different reward functions and models of policy parameterizations. The algorithms are proved to converge. Their performances are assessed and compared on several synthetic data sets. The proposed algorithms prove to be more efficient than a naive multi-armed bandit algorithm.

***K*eywords** Credit assignment · Multi-agent sequential learning · Policy gradient algorithm

1 Introduction

The job assignment problem involves assigning n machines to m agents such that each machine is assigned to exactly one agent, subject to certain constraints. This problem, as discussed in works like [1], is a classic optimization problem with no randomness and known constraints, making it inherently combinatorial in nature. The combinatorial nature of the problem arises from the exponential number of possible allocations of machines to agents, which increases rapidly with the number of agents and machines.

Similar to how Markov decision processes serve as stochastic generalizations of combinatorial problems, the problem examined in this article represents a stochastic variant of the assignment problem. In our scenario, agents lack access to the reward function they aim to optimize. Instead, they must estimate it by sampling combinations of agents and machines and observing the resulting, potentially random, rewards.

Our formalization introduces an additional complexity layer to the assignment problem by requiring agents to select one action among several available actions on the machines post-allocation. A practical example is when a manager assigns projects to teams, where the success depends on both the allocation and subsequent decisions made by the teams [2, 3].

Thus, the problem we consider unfolds in two steps:

1. **Centralized Allocation:** Each agent is allocated a machine, ensuring a one-to-one assignment.
2. **Decentralized Action Selection:** Each agent chooses an action from several options available on their allocated machine, where the number of possible actions is machine-specific.

Upon completing both steps, agents observe a reward, which is the same for all agents and can be used to update their decision policies. This problem can also be conceptualized as a sequential decision process, where a single agent moves

through a sequence of machines, selects an action at each machine, and receives a reward after interacting with all the machines.

Our approach does not make assumptions about how the reward is calculated from machine allocations and action choices. An example and the simplest way to formalize the action choices is to use Multi-Armed Bandits (MABs) [4, 5] on each machine, and observe the sum of the rewards of all pulled arms.

Since a combination of multiple decisions results in a single reward, a significant challenge is identifying which decisions are beneficial. This issue relates to the credit assignment problem in Reinforcement Learning (RL), where assigning credit or blame for individual decisions in a sequence is challenging when only a global reward is available [6, 7].

Our problem is also close to Multi-Agent Reinforcement Learning (MARL) [8], where multiple agents make decentralized decisions to maximize their own objectives. However, our scenario is distinct because all agents share the same objective function, and there is a centralized step in the decision process (machine allocation), which can be viewed as selecting the state of their environment.

Although there is increasing interest in experimentally studying sequential learning in MARL [9], convergence studies of such approaches are scarce. In the Decentralized Partially Observable Markov Decision Processes (Dec-POMDP) framework [10, 11, 12], which our setting can be seen as a special case of, convergence studies typically address parallel learning where agents make decisions independently.

We present and compare various models based on action sampling, permutation sampling, or a combination of both. For each model, we introduce a policy gradient ascent algorithm [13] and demonstrate that the algorithms converge to a zero of the gradient of the expected reward. Our proposed method operates with polynomial complexity, making it computationally feasible for larger problem instances. Additionally, we compare our method to a naive multi-armed bandit approach, which treats each agent-machine allocation independently. Our results show that our method outperforms the naive approach.

Our simulations on synthetic data sets, using several reward functions, provide insights into the limits of each model and the conditions under which they might converge to a sub-optimal zero of the gradient of the expected reward instead of the optimal decision.

2 Problem formulation

2.1 Notation and definitions

Let $\mathbb{N}^*$ be the set of positive integers. For any $n \in \mathbb{N}^*$, let $[n]$ be the set $\{1, 2, \dots, n\}$.

Let $n \in \mathbb{N}^*$ be a number of machines and agents, and write $\mathfrak{S}(n)$ the set of permutations of $[n]$. Permutations are used to specify what machines the agents are allocated to (or, when seeing the problem as a single agent sequentially acting on each machine, in what order the machines are picked). We write $\sigma = (\sigma_t)_{t\in[n]}$ the random variable taking values in $\mathfrak{S}(n)$ that describes the allocation of the machines: for any $t \in [n]$, σ_t is the machine picked by agent t (or the t-th machine picked by the agent in the sequential formulation).

Given a permutation $\tau \in \mathfrak{S}(n)$, we define

- $\tau_{k:l} := \{\tau_t\}_{k\leqslant t\leqslant l} \subset [n]$, for all $(k, l) \in [n]^2$, as the machines allocated to the agents numbered between k and l when their allocation is given by τ. If $l < k$, take $\tau_{k:l} = \varnothing$.
- $\tau^{-1}(i) \in [n]$, for all $i \in [n]$, as the agent allocated to machine i when their allocation is given by τ (or, in the sequential formulation, the time step at which machine i is acted upon).

Let $k_1, \dots, k_n \in \mathbb{N}^*$. For each $i \in [n]$, k_i is the cardinal of the set of actions available when on machine i, denoted as $[k_i]$. Let $\mathcal{A} = \prod_{i\in[n]}[k_i]$ be the set of vectors of actions.

Let $A = (A_i)_{t\in[n]} \in \mathcal{A}$ be the random variables that describes the vector of actions chosen by the agent on each machines, *i.e.*, $A_i \in [k_i]$ is the action taken on machine i (by agent, or at time, $\sigma^{-1}(i)$).

An *ordered trajectory* (or simply, a *trajectory*) is a vector $(\tau, \mathbf{a}) = ((\tau_t)_{t\in[n]}, (a_i)_{i\in[n]}) \in \mathfrak{S}(n) \times \mathcal{A}$, where $\tau_t \in [n]$ is the machine allocated to agent t and $a_i \in [k_i]$ is the action selected on machine i. Thus, (σ, A) is the ordered trajectory selected by the agents. We call its distribution the *policy*.

The experiment proceeds as follows: the ordered trajectory (σ, A) is sampled according to its policy, and the environment returns a reward R. We call this procedure an *episode*. Then, the policy is updated based on the trajectory sampled, the reward, and previous episodes.

For each episode $e \geqslant 1$, let $(\sigma^{(e)}, A^{(e)})$ be the random variable taking values in $\mathfrak{S}(n) \times \mathcal{A}$ that describes the trajectory sampled during the e^{th} episode, and let $R^{(e)}$ be the return of the environment at episode e.

Example 1
When the reward is of the form

$$R = \frac{1}{n} \sum_{i \in [n]} R_{i,\sigma^{-1}(i),A_i}$$

where the variables $(R_{i,t,a})_{i,t,a}$ are independent ($R_{i,t,a}$ is the reward generated by machine i, when allocated to agent t and when action a is chosen), our problem becomes a multi-agent multi-arm bandits [14, 15], where one and only one agent can act on each MAB and the reward from each MAB depends not only from the arm and MAB, but also on which agents pulls the arm.

Remark 1
Despite the appellation "machine", this problem is not, in general, a multi-agent multi-armed bandit, since these models assume that the reward has a specific structure.

2.2 Optimization problem

At the end of each episode $e \geqslant 1$, after choosing the trajectory $\left(\sigma^{(e)}, A^{(e)}\right)$, the agents receive a reward $R^{(e)} \in \mathbb{R}$ whose distribution depends only on the trajectory chosen. The goal of the agents is to maximize the expected value of $R^{(e)}$, that is to maximize

$$\pi \longmapsto \mathbb{E}_\pi\left[R^{(1)}\right] = \sum_{\substack{\tau \in \mathfrak{S}(n) \\ \mathbf{a} \in \mathcal{A}}} \pi(\sigma^{(1)} = \tau, A^{(1)} = \mathbf{a})\ \lambda_{\tau,\mathbf{a}}, \tag{1}$$

where $\lambda_{\tau,\mathbf{a}} = \mathbb{E}\left[R^{(1)} \,|\, \sigma^{(1)} = \tau, A^{(1)} = \mathbf{a}\right]$. Note that this conditional expectation does not depend on the policy π: it is a function of the environment and the trajectory, not of the agents' policy. In the following section, we present several classes of policies over which this objective function can be optimized.

3 Policies

In this section, we introduce four classes of policies to sample ordered trajectories. All of them are split into two parts: first, allocate the machine (*i.e.*, sample the permutation σ), and then, sample the actions. For each of these parts, we propose two options.

Fig. 1 represents the interactions of the multi-agent's entities in an experiment. A leader agent matches agents and machines (this is the sampling of the permutation σ). Then, each agent makes its own decision independently of the other agents. In this sense, this is a combination of a centralized policy –to allocate machines to agents– and a decentralized policy [16] –to choose the actions.

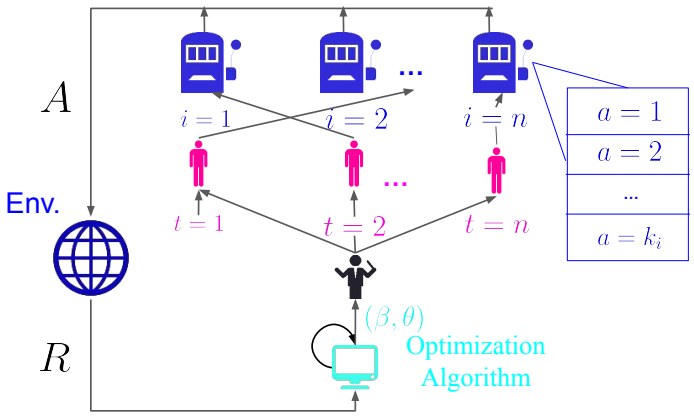

Figure 1: Interactions of the multi-agent entities with the environment and optimization algorithm.

We use the softmax function, denote as Φ, for specifying the policy probabilities. Let I be a finite set and $J \subset I$. Given a vector $\vartheta = (\vartheta_i)_{i\in I} \in \mathbb{R}^I$ of real numbers and an element $i \in I$, let

$$\Phi_J(\vartheta, i) = \mathbb{1}_{i\in J} \frac{e^{\vartheta_i}}{\sum_{j\in J} e^{\vartheta_i}},$$

be the i^{th} component of the softmax function evaluated on ϑ, restricted on the subset J. It defines a probability vector on I which gives mass 1 to J, from a vector of weights ϑ. When $J = I$, we simply write $\Phi(\vartheta, i)$.

3.1 Permutation sampling

In this section, we propose two policies π^1_β and π^2_β to sample the permutation describing the allocation of the machines.

	Model 1	Model 2
Notation	$\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_i \\ \vdots \\ \beta_n \end{pmatrix} \in \mathbb{R}^n$	$\beta = \begin{pmatrix} \beta_{1,1} & \cdots & \beta_{1,i} & \cdots & \beta_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{t,1} & \cdots & \beta_{t,i} & \cdots & \beta_{t,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{n,1} & \cdots & \beta_{n,i} & \cdots & \beta_{n,n} \end{pmatrix} \in \mathbb{R}^{n\times n}$

3.1.1 Model 1

For each $\beta = (\beta_1, \dots, \beta_n) \in \mathbb{R}^n$, let π^1_β be the distribution such that

$$\forall (t,i) \in [n]^2 \ , \ \pi^1_\beta(\sigma_t = i \,|\, \sigma_{1:(t-1)}) = \mathbb{1}_{i\in\sigma_{t:n}} \frac{e^{\beta_i}}{\sum_{j\in\sigma_{t:n}} e^{\beta_j}} = \Phi_{\sigma_{t:n}}(\beta, i).$$

Hence, for all $\tau \in \mathfrak{S}(n)$

$$\pi^1_\beta(\sigma = \tau) = \prod_{t\in[n]} \pi^1_\beta\big(\sigma_t = \tau_t \,|\, \sigma_{1:(t-1)} = \tau_{1:(t-1)}\big) = \prod_{t\in[n]} \Phi_{\tau_{t:n}}(\beta, \tau_t).$$

In this model, the same permutation credits β are used throughout the assignment of the agents. In particular, if a machine with a high probability to be chosen is not allocated to an agent, it will still have a high probability to be allocated to the next agent.

3.1.2 Model 2

For each $\beta \in \mathcal{M}_n(\mathbb{R})$ let π^2_β be the distribution such that

$$\forall t \in [n] \ , \ \forall i \in [n] \ , \ \pi^2_\beta(\sigma_t = i \,|\, \sigma_{1:(t-1)}) = \mathbb{1}_{i\in\sigma_{t:n}} \frac{e^{\beta_{t,i}}}{\sum_{j\in\sigma_{t:n}} e^{\beta_{t,j}}} = \Phi_{\sigma_{t:n}}(\beta_t, i).$$

Hence, for all $\tau \in \mathfrak{S}(n)$,

$$\pi^2_\beta(\sigma = \tau) = \prod_{t\in[n]} \pi^2_\beta\big(\sigma_t = \tau_t \,|\, \sigma_{1:(t-1)} = \tau_{1:(t-1)}\big) = \prod_{t\in[n]} \Phi_{\tau_{t:n}}(\beta_t, \tau_t).$$

In this model, each step of the assignment process uses a different credit vector β_t. In particular, the machine with the highest probability to be allocated may be different depending on the agent. This also means that the optimization of the permutation credits are decoupled between agents, making the optimization more flexible than in model 1, where the dependency between allocations interferes with the convergence of the algorithm, as shown in Fig. 2, and may even mislead it, see Fig. 5.

3.2 Action sampling

In this section, we propose two policies π_θ^A and π_θ^B for sampling the agents' actions. These policies are conditional distributions, conditioned on the allocation permutation, and such that each agent makes its decision independently of the others (conditionally to the permutation).

In the first policy, the policy of the agent depends only on the machine it has been allocated, not on who the agent is. In the second policy, the policy of the agent depends on the machine it has been allocated as well as who the agent is.

	Model A	Model B
Notation	$\theta_i = \begin{pmatrix} \theta_{i,1} \\ \vdots \\ \theta_{i,a} \\ \vdots \\ \theta_{i,k_i} \end{pmatrix} \in \mathbb{R}^{k_i}$	$\theta_i = \begin{pmatrix} \theta_{i,1,1} & \cdots & \theta_{i,1,a} & \cdots & \theta_{i,1,k_i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_{i,t,1} & \cdots & \theta_{i,t,a} & \cdots & \theta_{i,t,k_i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \theta_{i,n,1} & \cdots & \theta_{i,n,a} & \cdots & \theta_{i,n,k_i} \end{pmatrix} \in \mathcal{M}_{n,k_i}$

3.2.1 Model A

For each $\theta = (\theta_1, \dots, \theta_n) \in \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_n}$, let π_θ^A be the conditional distribution such that $\forall i \in [n], \forall a \in [k_i], \forall \tau \in \mathfrak{S}(n)$

$$\pi_\theta^A(A_i = a \,|\, \sigma = \tau) = \frac{e^{\theta_{i,a}}}{\sum_{b \in [k_i]} e^{\theta_{i,b}}} = \Phi(\theta_i, a),$$

and the actions $A_1, \dots, A_n$ are independent conditionally to σ. Note that in this model, the actions taken are independent of the allocation permutation σ.
Hence, for all $(\tau, \mathbf{a}) \in \mathfrak{S}(n) \times \mathcal{A}$, $u \in \{1, 2\}$, β and θ,

$$\begin{aligned} \pi_{\beta,\theta}^{u,A}((\sigma, A) = (\tau, \mathbf{a})) &= \pi_\beta^u(\sigma = \tau) \prod_{i \in [n]} \pi_\theta^A\big(A_i = a_i \,|\, \sigma = \tau\big), \\ &= \pi_\beta^u(\sigma = \tau) \prod_{i \in [n]} \Phi(\theta_i, a_i). \end{aligned}$$

3.2.2 Model B

For each $\theta = (\theta_1, \dots, \theta_n) \in \mathcal{M}_{n,k_1} \times \cdots \times \mathcal{M}_{n,k_n}$, let π_θ^B be the conditional distribution such that $\forall i \in [n], \forall t \in [n], \forall a \in [k_i], \forall \tau \in \mathfrak{S}(n)$

$$\begin{aligned} \pi_\theta^B(A_i = a \,|\, \sigma = \tau) &= \sum_{t \in [n]} \mathbb{1}_{\tau_t = i} \frac{e^{\theta_{i,t,a}}}{\sum_{b \in [k_i]} e^{\theta_{i,t,b}}}, \\ &= \sum_{t \in [n]} \mathbb{1}_{\tau_t = i} \, \Phi(\theta_{i,t}, a), \end{aligned}$$

and the actions $A_1, \dots, A_n$ are independent conditionally to σ.
Hence, for all $(\tau, \mathbf{a}) \in \mathfrak{S}(n) \times \mathcal{A}$, $u \in \{1, 2\}$, β and θ,

$$\begin{aligned} \pi_{\beta,\theta}^{u,B}\big((\sigma, A) = (\tau, \mathbf{a})\big) &= \pi_\beta^u(\sigma = \tau) \prod_{i \in [n]} \pi_\theta^B\big(A_i = a_i \,|\, \sigma = \tau\big), \\ &= \pi_\beta^u(\sigma = \tau) \prod_{t \in [n]} \Phi(\theta_{\tau_t, t}, a_{\tau_t}), \\ &= \pi_\beta^u(\sigma = \tau) \prod_{i \in [n]} \Phi(\theta_{i, \tau^{-1}(i)}, a_i). \end{aligned}$$

3.3 Policy gradients

The gradient of the expected value of the reward for each model, which will be used in the Algorithms in Section 4, is as follows.

Theorem 1
Let $B^{(1)}$ be a random variable that is independent of $\sigma^{(1)}$. For all $(i,t) \in [n]^2$ and $v \in \{A,B\}$,

$$\frac{\partial \mathbb{E}^{1,v}_{\beta,\theta}\left[R^{(1)}\right]}{\partial \beta_i} = \mathbb{E}^{1,v}_{\beta,\theta}\left[\left(1 - \sum_{t=1}^{\sigma^{(1)^{-1}}(i)} \Phi_{\sigma^{(1)}_{t:n}}(\beta, i)\right)\left(R^{(1)} - B^{(1)}\right)\right]. \tag{2}$$

$$\frac{\partial \mathbb{E}^{2,v}_{\beta,\theta}\left[R^{(1)}\right]}{\partial \beta_{t,i}} = \mathbb{E}^{2,v}_{\beta,\theta}\left[\left(\mathbb{1}_{i=\sigma^{(1)}_t} - \Phi_{\sigma^{(1)}_{t:n}}(\beta_t, i)\right)\left(R^{(1)} - B^{(1)}\right)\right]. \tag{3}$$

Proof 1
Proof in supplementary materials Sec. 2.1.

Theorem 2
Let $B^{(1)}$ be a random variable that is independent of $(\sigma^{(1)}, A^{(1)})$. For all $i,t \in [n]$, $a \in [k_i]$ and $u \in \{1,2\}$,

$$\frac{\partial \mathbb{E}^{u,A}_{\beta,\theta}\left[R^{(1)}\right]}{\partial \theta_{i,a}} = \mathbb{E}^{u,A}_{\beta,\theta}\left[\left(\mathbb{1}_{A^{(1)}_i = a} - \Phi(\theta_i, a)\right)\left(R^{(1)} - B^{(1)}\right)\right]. \tag{4}$$

$$\frac{\partial \mathbb{E}^{u,B}_{\beta,\theta}\left[R^{(1)}\right]}{\partial \theta_{i,t,a}} = \mathbb{E}^{u,B}_{\beta,\theta}\left[\mathbb{1}_{\sigma^{(1)}_t = i}\left(\mathbb{1}_{A^{(1)}_i = a} - \Phi(\theta_{i,t}, a)\right)\left(R^{(1)} - B^{(1)}\right)\right]. \tag{5}$$

Proof 2
Proof in supplementary materials Sec. 2.2.

4 Policy search algorithm

Algorithm 1 is a gradient ascent algorithm based on the gradient from Theorems 1 and 2. Let us define the empirical gradient functions $\mathrm{Grad}^{u,v} = (\mathrm{Grad}^u_1, \mathrm{Grad}^v_2)$, for $u \in \{1,2\}$ and $v \in \{A,B\}$, by

$$\begin{cases} \left(\mathrm{Grad}^1_1(\sigma, A, \beta, \theta, X)\right)_i = \left(1 - \sum_{t=1}^{\sigma^{-1}(i)} \Phi_{\sigma_{t:n}}(\beta, i)\right) X, \\ \left(\mathrm{Grad}^2_1(\sigma, A, \beta, \theta, X)\right)_{t,i} = \left(\mathbb{1}_{i=\sigma_t} - \Phi_{\sigma_{t:n}}(\beta_t, i)\right) X, \\ \left(\mathrm{Grad}^A_2(\sigma, A, \beta, \theta, X)\right)_{i,a} = \left(\mathbb{1}_{A_i=a} - \Phi(\theta_i, a)\right) X, \\ \left(\mathrm{Grad}^B_2(\sigma, A, \beta, \theta, X)\right)_{i,t,a} = \mathbb{1}_{\sigma_t=i}\left(\mathbb{1}_{A_i=a} - \Phi(\theta_{i,t}, a)\right) X. \end{cases}$$

Note that these gradients means that updating the policy parameters of sampling the actions of machine i only relies on the global reward, the action on this machine, the policy parameters of this machine, and (in Model B) on the agent the machine has been allocated to. In that sense, both sampling the action on a machine and updating the policy of that machine (given the allocation permutation) rely only on local information, not on what happens on the other machines.

Algorithm 1 GAtACA-Series $((n, \mathcal{A}), u \in \{1,2\}, v \in \{A,B\}, \alpha > 0, \varepsilon > 0, \gamma > 0)$

initialize: $e = 1$, $B^{(1)} = 0$
repeat
  sample $(\sigma^{(e)}, A^{(e)}) \sim \pi^{u,v}_{\beta^{(e)},\theta^{(e)}}$
  get $R^{(e)}$
  $(\beta^{(e)}, \theta^{(e)}) \leftarrow (\beta^{(e)}, \theta^{(e)}) + \alpha \mathrm{Grad}^{u,v}\left(\sigma^{(e)}, A^{(e)}, \beta^{(e)}, \theta^{(e)}, R^{(e)} - B^{(e)}\right)$
  $B^{(e+1)} \leftarrow \frac{1}{1-\gamma^e}\left(R^{(e)} + \gamma(1-\gamma^{e-1})B^{(e)}\right)$
  $e \leftarrow e + 1$
until $|R^{(e-1)} - B^{(e)}| < \varepsilon$
return $\left(\beta^{(e-1)}, \theta^{(e-1)}\right)$

Theorem 3 (Convergence)
Let $u \in \{1,2\}$ and $v \in \{A,B\}$. Assume that there exists $r > 0$ such that $|R^{(e)}| \leqslant r$ and $|B^{(e)}| \leqslant r$ for all $e \geqslant 1$. Let $(\alpha_e)_{e\geqslant 1}$ be a sequence of nonnegative real numbers satisfying the Robbins-Monro conditions

$$\sum_{e=1}^{+\infty} \alpha_e = +\infty \quad \text{and} \quad \sum_{e=1}^{+\infty} \alpha_e^2 < +\infty.$$

For all $t \in [n]$ and $i \in [n]$, update $((\beta^{(e)}, \theta^{(e)}))_{e\geqslant 1}$ according to the rule

$$\left(\beta^{(e)}, \theta^{(e)}\right) \leftarrow \left(\beta^{(e)}, \theta^{(e)}\right) + \alpha_e \operatorname{Grad}^{u,v}\left(\sigma^{(e)}, A^{(e)}, \beta^{(e)}, \theta^{(e)}, R^{(e)} - B^{(e)}\right).$$

Then the expected reward $\left(\mathbb{E}^{u,v}_{\beta^{(e)},\theta^{(e)}}[R^{(1)}]\right)_{e\geqslant 1}$ converges. Moreover, if $u = 2$, for each e, t, i, a, let $\tilde{\theta}^{(e)}_{i,(t,)a} = \theta^{(e)}_{i,(t,)a} - \max_{a\in[k_i]} \theta^{(e)}_{i,(t,)a}$, and $\tilde{\beta}^{(e)}_{t,i} = \beta^{(e)}_{t,i} - \max_{j\in[n]} \beta^{(e)}_{t,j}$. Let $\mathcal{B}$ be the (compact) set of parameters $(\tilde{\beta}_{t,i})_{t,i}$ such that $\tilde{\beta}_{t,i} \in [-\infty, 0]$ and $\max_{j\in[n]} \tilde{\beta}_{t,j} = 0$ for all t, i, and likewise, let Θ be the (compact) set of parameters $(\tilde{\theta}_{i,t,a})_{i,t,a}$ (when $v = B$, and $(\tilde{\theta}_{i,a})_{i,a}$ when $v = A$) such that $\tilde{\theta}_{i,(t,)a} \in [-\infty, 0]$ and $\max_{a\in[k_i]} \tilde{\theta}_{i,(t,)a} = 0$ for all t, i, a.

Then the limit points of $(\tilde{\beta}^{(e)}, \tilde{\theta}^{(e)})_{e\geqslant 1}$ are all in the same connected component of the set of zeroes of the gradient of $(\tilde{\beta}, \tilde{\theta}) \in \mathcal{B} \times \Theta \longmapsto \mathbb{E}^{2,v}_{\tilde{\beta},\tilde{\theta}}[R^{(1)}]$.

Proof 3
The proof is similar to the proof of Theorem 2 of [15]: by Proposition 3 of [17], the expectation of the reward converges and its gradient with respect to β and θ converges to zero. The fact that the limits of the sequence belong to the same connected component is due to the fact that $\|(\tilde{\beta}^{(e+1)}, \tilde{\theta}^{(e+1)}) - (\tilde{\beta}^{(e)}, \tilde{\theta}^{(e)})\| \longrightarrow 0$.

Changing (β, θ) into $(\tilde{\beta}, \tilde{\theta})$ ensures that the parameter space is compact and clears up potential problems with the convergence of the parameters. For instance, consider the case where $(\beta^{(e)})_e$ is such that the policy of σ converges towards the policy π defined by $\pi(\sigma_1 = 1) = p$ and $\pi(\sigma_1 = 2) = 1 - p$, with $1/2 < p < 1$. To converge towards this limit, $(\beta^{(e)}_{1,\cdot})_e$ must converge to a β such that $\beta_1 = \beta_2 = +\infty$ and $\beta_j = -\infty$ for $j > 2$ (since $\sum_j \beta^{(e)}_{t,j} = 0$ for all e, t). This does not properly describe the distribution π. In contrast, the sequence $(\tilde{\beta}^{(e)}_{1,\cdot})_e$ converges to a $\tilde{\beta}$ such that $\tilde{\beta}_1 = 0$, $\tilde{\beta}_2 = \log(1/p - 1)$ and $\tilde{\beta}_j = -\infty$ for $j > 2$, which does define a proper policy.

5 Simulation results

Consider the following reward function

$$\begin{aligned} R : \mathfrak{S}(n) \times \mathcal{A} &\longrightarrow \mathbb{R} \\ (\tau, \mathbf{a}) &\longmapsto \frac{1}{n} \sum_{t\in[n]} \nu_{t,\tau_t,a_{\tau_t}}, \end{aligned} \tag{6}$$

where $\nu_{t,\tau_t,a_{\tau_t}}$ is the reward of agent t after having been allocated machine τ_t and choosing action $a \in [k_{\tau_t}]$. We test our policy models $\pi^{u,v}_{\beta,\theta}$, where $(u, v) \in \{1,2\} \times \{A, B\}$, for different reward functions for the problem stated in Section 2.2, and show the advantages and limitations of each model.

In section 5.1, we consider the case where the actions set is reduced to a single action for all the agents. That is, for all $i \in [n]$, $k_i = 1$. Therefore, the objective is to find the matching of agents and machines maximizing the total reward R. This situation corresponds to an assignment problem. In this case, the policies comes down to the permutation sampling models presented in Section 3.1.

In section 5.2, we consider the general case where each agent has to choose between one or more actions. The goal in this case is to allocate the machines to the agents and pick one action per machine in a way that maximizes the total reward.

Table 1 describes the different forms of reward functions and the different models considered.

Reward \ Model	1	2	2A	2B
$\frac{1}{n}\sum_{t\in[n]} \mathbb{1}_{\tau_t=\tau_t^\star}$ (Sec. 5.1.1)	✓	✓		
$\frac{1}{n}\sum_{t\in[n]} \nu_{t,\tau_t}$ (Sec. 5.1.2)	✗	✓		
$\frac{1}{n}\sum_{t\in[n]} \nu_{t,\tau_t,a}$ (Sec. 5.2.1) Case 1: best action independent of agent			✓	✓
$\frac{1}{n}\sum_{t\in[n]} \nu_{t,\tau_t,a}$ (Sec. 5.2.2) Case 2: best action dependent of agent			✗	✓

Table 1: Reward functions and models. Green checks indicates cases where the algorithm's limit is optimal. Red crosses indicate cases where the algorithm's limit is sub-optimal. Model 1 is not considered in Section 5.2, where the agents must pick actions in addition to the allocation, since Model 1 is shown to converge either slower than Model 2, or to a sub-optimal limit, in Section 5.1.

5.1 Permutation sampling

For the permutation sampling of models 1 and 2, we consider two cases of reward functions, Eq. (7) and Eq. (9). For both cases, we consider $n = 12$ agents, a constant learning rate $\alpha = 0.01$ and 1.5×10^5 episodes. Each simulation is performed over 30 replications. More details on the simulations can be found in supplementary materials Sec. 1

5.1.1 Case 1

Let $\tau^* \in \mathfrak{S}(n)$ be a target permutation, and consider the following reward function:

$$R : \tau \in \mathfrak{S}(n) \longmapsto R(\tau) = \frac{1}{n}\sum_{t\in[n]} \mathbb{1}_{\tau_t=\tau_t^\star}. \tag{7}$$

This reward function returns the proportion of machines allocated according to the target permutation τ^*.

Fig. 2 shows the average rewards obtained by both models of the policy π_β^1 and π_β^2 respectively. Both model were able to converge to an optimal policy for which the reward is maximized; however, model 1 converges slowly than model 2.

Fig. 3 shows the evolution of the values of the parameter $(\hat{\beta}_i)_{i\in[n]}$ for model 1 and the values of $(\hat{\beta}_{t,i})_{i\in[n]}$ at time step $t = 6$ for model 2 (left and right respectively).

For example, consider the target order $\tau^* = (12, 1, 8, 10, 2, 3, 7, 4, 9, 11, 6, 5)$ represented in the legend of Fig. 3 (right). In model 1, the policy parameters $\hat{\beta}_i$ produced by the algorithm are such that the machine $i = 12$ (in red), followed by $i = 1$ (in blue) and so on, according to the target permutation τ^* (the algorithm converged to the correct limit). In model 2 though, the parameters $\hat{\beta}_{t,\tau_t^*}$ are eventually notably higher than the other parameters $\hat{\beta}_{t,i}$, $i \neq \tau_t^*$, making it almost certain that the machine τ_t^* will be allocated to agent t. In the example above, the agent at time step $t = 6$ is $\tau_t^* = 3$. Its parameter value $\hat{\beta}_{t,i}$ continue to increase while the other agents $\tau_{(t+1):n}$ parameter values decrease. Therefore, the agent $i = 3$ is more probable to be chosen at that time step, if it has not been chosen before).

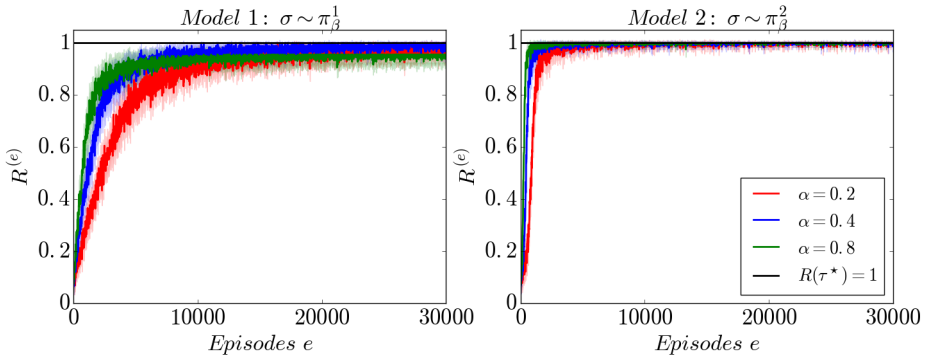

Figure 2: Average reward R on model 1 and on model 2 (left and right respectively) for different values of the learning rates α. The horizontal curve $y = 1$ corresponds to the optimum of the function R. Several learning rates α have been tested to ascertain the best rate of convergence for Model 1, and it seems to be achieved for some α between 0.2 (red curve) and 0.8 (green curve), for which the algorithm no longer converge. For this whole range of α, Model 1 is handily beaten by Model 2. For some learning rates (*e.g.*, $\alpha = 0.2$), Model 1 converges slower than model 2, and sometimes model 1 does not converge to the optimal solution (*e.g.*, $\alpha = 0.8$) while model 2 still converges to the optimal trajectory.

Note that during episode e, the parameters $\hat{\beta}_{t,j}$ for $j \in \sigma^{(e)}_{1:(t-1)}$ will not change; in particular, when the policy converges to a Dirac in τ^*, the parameters $\hat{\beta}_{t,j}$ for $j \in \tau^*_{1:(t-1)}$ will remain almost stationary, around 0 in Fig. 3.

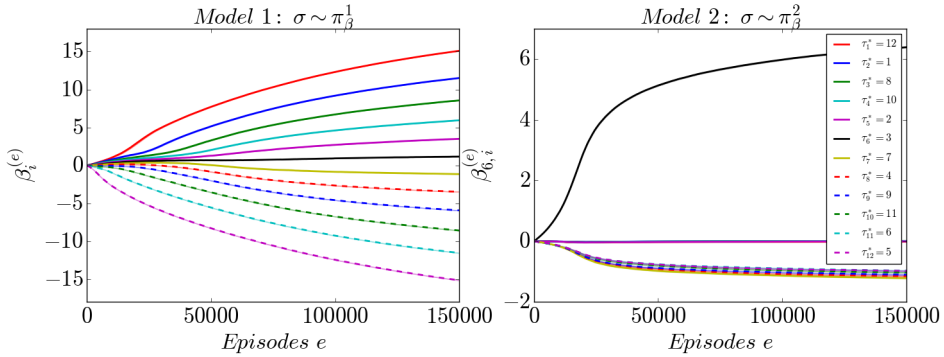

Figure 3: Average $(\hat{\beta})_{i\in[n]}$ on model 1 (left) and model 2 for agent $t = 6$ (right). See Fig. S.1 for full visualization of $\beta_{t,i}$ on model 2. Each color and/or shape corresponds to a different machine. The legends shown in the right are sorted w.r.t. the optimal allocation order τ^*.

The most probable ordered trajectory $\hat{\tau}$ for both models at the last episode is defined as follows

$$\begin{cases} \hat{\tau}_1 = \min\Big(\underset{i\in[n]}{\operatorname{argmax}} (\hat{\beta})_{1,i}\Big), \\ \hat{\tau}_t = \min\Big(\underset{i\in[n]\setminus\hat{\tau}_{1:(t-1)}}{\operatorname{argmax}} (\hat{\beta})_{t,i}\Big), \quad \forall t \geqslant 2. \end{cases} \tag{8}$$

In both case, $\hat{\tau} = \tau^*$. Fig. 4 shows the end policy for each model. That is

$$\forall t \in [n], \forall i \in [n], \quad \pi^u_{\hat{\beta}}(i = \hat{\tau}_t).$$

We sorted the columns with respect to the target order τ^* to facilitate the comparison between the learned order $\hat{\tau}$ and the target order τ^* by looking at the diagonal of the matrix.

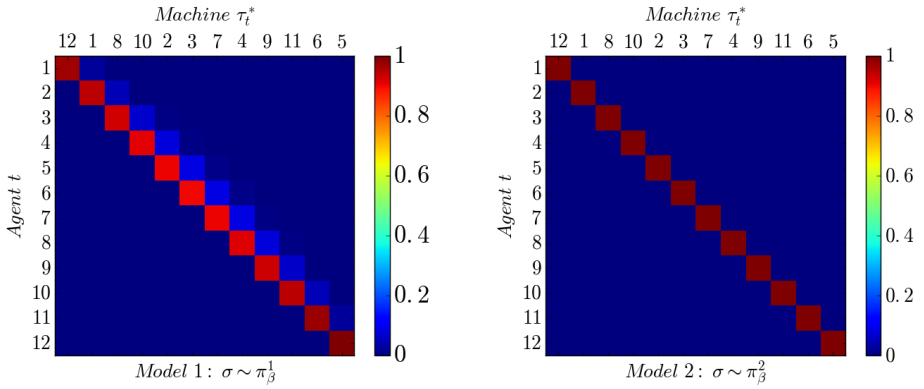

Figure 4: $\pi_{\hat{\beta}}^u(\sigma_t = i)_{(t,i)\in[n]^2}$, $u \in \{1, 2\}$. Lines correspond to the agents $t \in [n]$ and the columns correspond to the machines $i \in [n]$. The columns are sorted w.r.t. the optimal allocation order τ^*. See Fig S.2 for full visualization of the evolution of $\pi_{\beta^{(e)}}^u(\sigma_t = i)_{(t,i)\in[n]^2}$ for all episodes e.

Both models converge to an almost Dirac distribution, where at each time step t, τ_t^* is the most likely to be chosen. As discussed above, model 1 converge notably more slowly than model 2. This can be seen on the evolution of the policy for model 1 in Fig. S.2a in supplementary materials. The policy probabilities converge faster to 0 or 1 for the first and last agents, and take longer to converge for the agents in the middle. In comparison, in model 2, the algorithm reinforces the probability of the "good" machine with the same speed for all agents.

5.1.2 Case 2

Let $(\nu_{t,i})_{(t,i)\in[n]^2} \in \mathbb{R}^{n\times n}$ and consider the following reward function

$$R : \tau \in \mathfrak{S}(n) \longmapsto R(\tau) = \frac{1}{n}\sum_{t\in[n]} \nu_{t,\tau_t}, \tag{9}$$

where $\nu_{t,i}$ is the reward of allocating machine i at agent t. The reward function is this case returns the average reward of the agents. The goal is to find the order τ^* that maximizes the expected reward R.

Fig. 5 shows the average rewards obtained by both models of the policy π_β^1 and π_β^2 respectively. For any learning rate considered here, only model 2 was able to converge to an optimal policy for which the reward is maximized, whereas model 1 converged to a non optimal policy where the reward is not maximized.

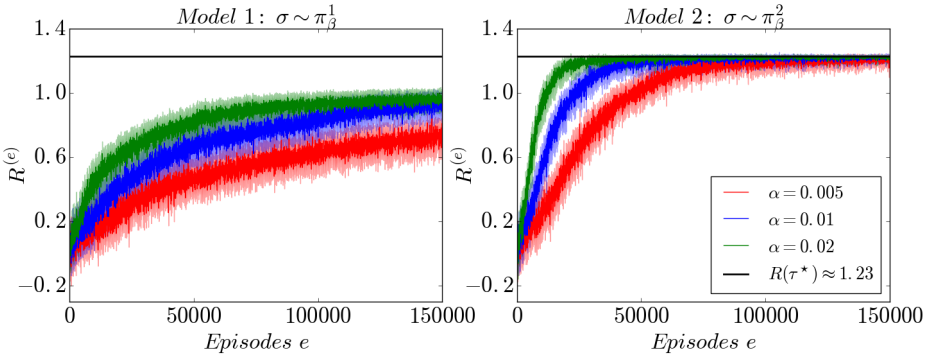

Figure 5: Average reward R on model 1 (left) and on model 2 (right) for different values of the learning rates α. The horizontal curve $y \approx 1.23$ corresponds to the maximum of the function R.

Fig. 6 shows the evolution of the averaged values of the parameter $(\beta_i)_{i\in[n]}$ for model 1 and the values of $(\beta_{t,i})_{i\in[n]}$ for agent $t = 6$ and model 2 (left and right respectively). In model 1, the algorithm did not manage to discriminate the agents.In model 2, for each agent t, the algorithm manages to clearly identify the best machine, in the sense that the

parameter β_{t,τ_t^*} grows significantly larger than the others, where τ^* is the permutation that achieves the optimal reward. For instance, at time step $t = 6$, the "good" agent is $i = 9$.

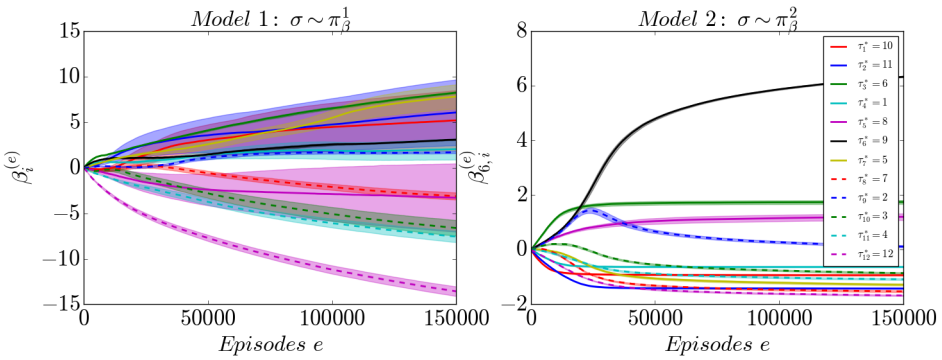

Figure 6: Evolution of $(\beta_i)_{i\in[n]}$ on model 1 (left) and $(\beta_{t,i})_{i\in[n]}$ for agent $t = 6$ on model 2 (See Fig S.3 for full visualisation of $\beta_{t,i}$). Each color and/or shape corresponds to a different machine. The legends shown in the right are sorted w.r.t. the optimal allocation order τ^*.

The most probable allocation permutation for models 1 and 2 ($\hat{\tau}^1$ and $\hat{\tau}^2$ respectively) are computed as in the previous section. We show the end policy for each model in Fig. 7. As stated before, model 1 did not converge to an optimal policy (that is $\hat{\tau}^1 \neq \tau^*$), whereas model 2 has converged to an almost Dirac measure where $\hat{\tau}^2 = \tau^*$.

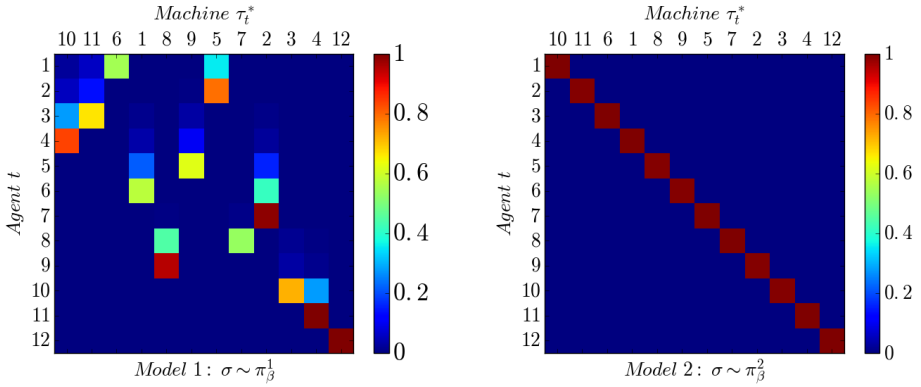

Figure 7: The end policy $\pi_\beta^u(\sigma_t = i)_{(t,i)\in[n]^2}$, $u \in \{1, 2\}$. Lines correspond to the agents $t \in [n]$ and the columns correspond to the machines $i \in [n]$. The columns are sorted w.r.t. the optimal allocation order τ^*. (See Fig S.4 for full visualization of $\pi_{\beta^{(e)}}^u(\sigma_t = i)_{(t,i)\in[n]^2}$ for all episodes e.)

5.2 Sampling permutation and actions

Consider the following reward function

$$\begin{aligned} R : \mathfrak{S}(n) \times \mathcal{A} &\longrightarrow \mathbb{R} \\ (\tau, \mathbf{a}) &\longmapsto \frac{1}{n} \sum_{t\in[n]} \nu_{t,\tau_t,a}, \end{aligned} \tag{10}$$

where $\nu_{t,\tau_t,a}$ is the reward of machine τ_t allocated to agent t after action $a \in [k_{\tau_t}]$ is chosen.

In this section, we consider $n = 9$ agents, for which each machine i has $k_i \in \{1, \ldots, 7\}$ actions, and 3×10^5 episodes.

We keep model 2 for the permutation parameter β, and test our policy models $\pi_{\beta,\theta}^{2,v}$, where $v \in \{A, B\}$, for two cases

- Case 1: when there exists one action for each machine that returns the highest reward among the other actions at any time step. That is,

$$\forall i \in [n],\ \exists a \in [k_i] :\ \forall t \in [n], \quad \underset{b \in [k_i]}{\operatorname{argmax}}(\nu_{t,i,b}) = \{a\}.$$

  In that case, the action that should be chosen on a machine does not depend on the agent the machine is allocated to.

- Case 2: when the hypothesis in case 1 is not satisfied. That is, the action with the highest reward may depend on the agent the machine is allocated to. In this case, model A is not be appropriate, as shown in Section 5.2.2.

For case 1, we order the vector $(\nu_{t,i,a})_{a \in [k_i]}$ in increasing order for each t and i to ensure that action k_i is the optimal action on machine i, whichever the agent may be.

A notable difference between Algorithm 1 and the one we use in practice is the update of the parameters in this section: instead of using the same learning rate α for β and θ, we update it as

$$\begin{pmatrix} \beta^{(e+1)} \\ \theta^{(e+1)} \end{pmatrix} \leftarrow \begin{pmatrix} \beta^{(e)} \\ \theta^{(e)} \end{pmatrix} + \begin{pmatrix} \alpha_\beta \operatorname{Grad}_1^{u,v}\left(\sigma^{(e)}, A^{(e)}, \beta^{(e)}, \theta^{(e)}, R^{(e)} - B^{(e)}\right) \\ \alpha_\theta \operatorname{Grad}_2^{u,v}\left(\sigma^{(e)}, A^{(e)}, \beta^{(e)}, \theta^{(e)}, R^{(e)} - B^{(e)}\right) \end{pmatrix},$$

with $\alpha_\beta = \dfrac{\alpha_\theta}{n}$.
While not covered by Theorem 3, this update scheme allows our algorithms to converge much faster in practice.

5.2.1 Case 1

In this section, we assume that the best action on any given machine does not depend on the agent it is allocated to.

Fig. 8 shows the average rewards obtained by both models of policies $\pi_{\beta,\theta}^{2,A}$ and $\pi_{\beta,\theta}^{2,B}$. Both models were able to converge to an optimal policy for which the reward is maximized for any value of the learning rate α.

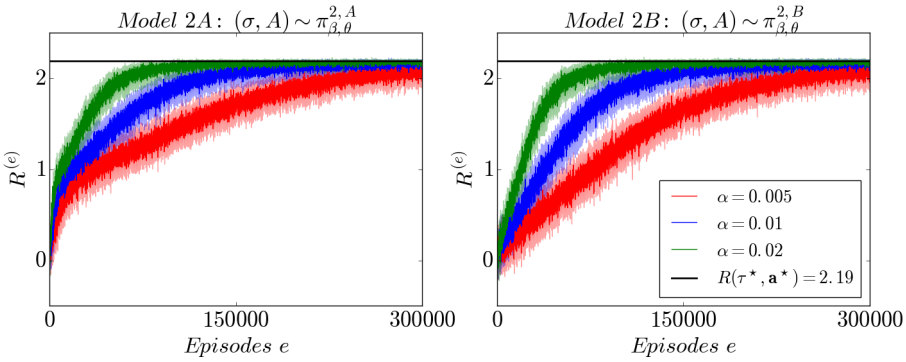

Figure 8: Average reward R on model 2A (left) and on model 2B (right) for different values of the learning rates α. The horizontal curve $y \approx 2.19$ corresponds to the optimum of the function R.

Fig. 9 shows the evolution of the permutation parameter $(\beta_{t,i})_{i \in [n]}$ for agent $t = 3$. Both models were able to discriminate the "good" machine to be allocated to that agent ($\tau_3^\star = 8$, where $(\tau^\star, \mathbf{a}^\star)$ is the optimal trajectory). Both models eventually recover to the optimal allocation $\tau^\star$.

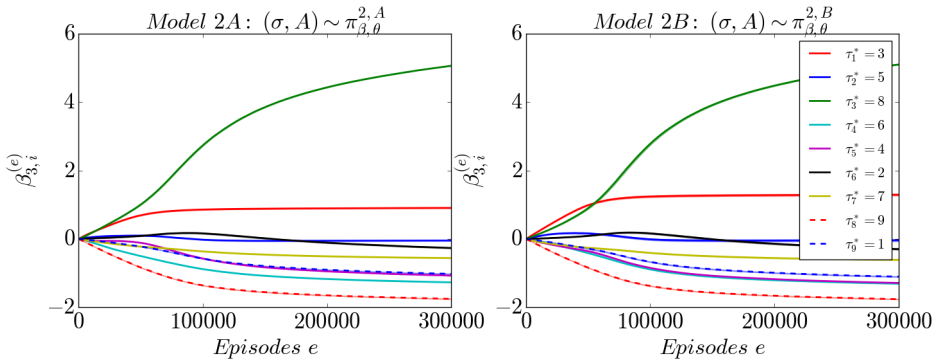

Figure 9: Evolution of $(\beta_{t,i})_{i\in[n]}$ on model 2A (left) and on model 2B (right) for agent $t = 3$ (See Fig S.5 for full visualisation of $\beta_{t,i}$). Each color and/or shape corresponds to a different machines. The legend shown in the right are sorted w.r.t. the optimal allocation order τ^*.

We show in Fig. 10 the end policy (as defined in the previous section) ordered with respect to the optimal order τ^*. For both models, the policy of the permutation converges to a Dirac measure at τ^*.

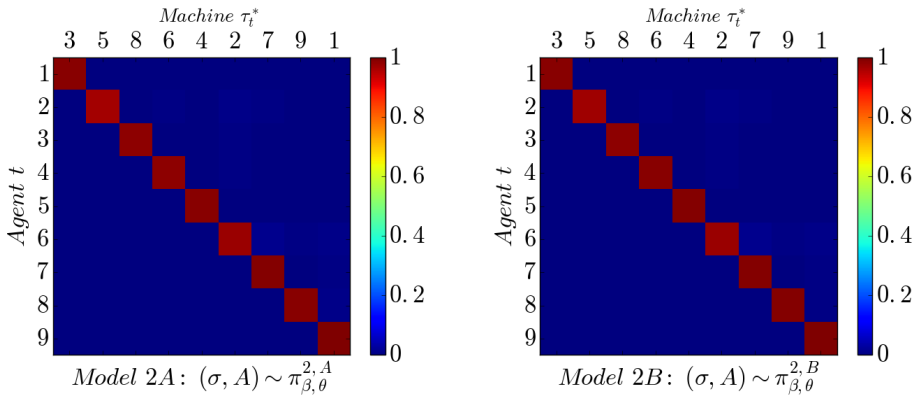

Figure 10: $\pi^{2,v}_{\hat{\beta},\hat{\theta}}(\sigma_t = i)_{(t,i)\in[n]^2}$, $v \in \{A, B\}$. Lines correspond to the agents $t \in [n]$ and the columns correspond to the machines $i \in [n]$. The columns are sorted w.r.t. the optimal allocation order τ^*. See Fig S.6 for full visualization of $\pi^{2,v}_{\beta^{(e)},\theta^{(e)}}(\sigma_t = i)_{(t,i)\in[n]^2})$ for all episodes e.

An optimal allocation order does not mean that the reward is maximized. It remains to be checked whether each agent has selected the best action or not. Fig. 11 shows the evolution of machine $i = 6$'s action parameters $(\theta_{i,a})_{a\in[k_i]}$ and $(\theta_{i,t,a})_{a\in[k_i]}$ for model 2A and model 2B respectively. In the latter, we considered agent $t = 4$, as it was the best match to machine $i = 6$.

Action $a = 6$ is most likely to be chosen in both cases, as its parameter is significantly larger than the other actions. In model 2A, it increases faster than in model 2B. This may explains why the reward grows faster at the start for model 2A than for model 2B on Fig. 8.

Fig. S.8 shows the $\pi^{2,v}_{\beta^{(e)},\theta^{(e)}}(A_i = a)$ for each machine $i \in \tau^*$, $v \in A, B$ at each episode e. The red curves represent the action with the highest reward. Both models reinforce the actions with the highest reward.

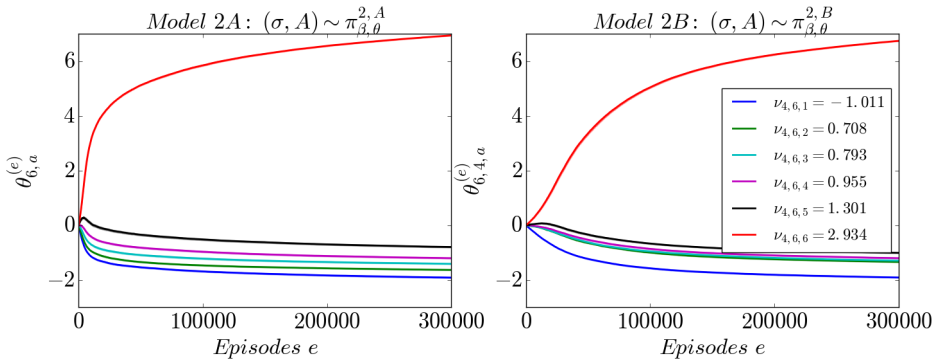

Figure 11: Evolution of $(\theta_{i,a})_{a\in[k_i]}$ for machine i on model 2A (left) and $(\theta_{i,t,a})_{a\in[k_i]}$ for the same machine allocated to agent $t = 4$ on model 2B (right) See Fig S.7 for full visualization of θ on both models. Each color corresponds to a different action. The optimal action is in red.

5.2.2 Case 2

Now, we allow the optimal action on a machine to depend on the agent it is allocated to.

Fig. 12 shows the average reward obtained for both models 2A and 2B (left and right respectively). Model 2A does not converge to an optimal ordered trajectory for which the reward is maximized, since it does not account for the fact that the optimal action depends on the agent, whereas model 2B does. This is not an issue of the optimal trajectory being unreachable, since a near-Dirac at the optimal trajectory is a valid policy under model 2A, but rather that the lack of flexibility hinders the convergence of the algorithm.

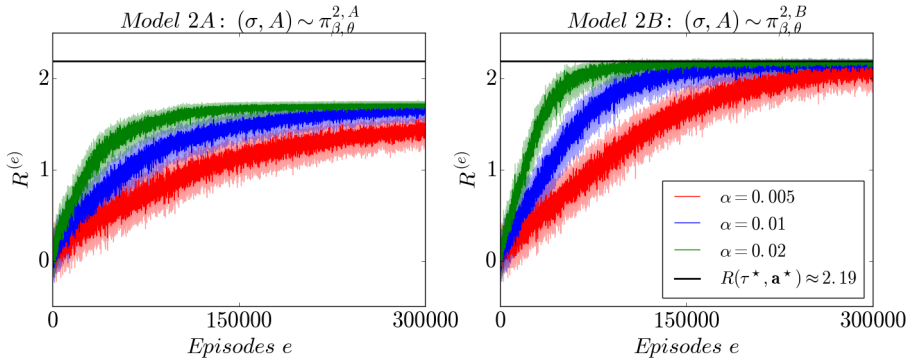

Figure 12: Average reward R on model 2A (left) and on model 2B (right) for different values of the learning rates α. The horizontal curve $y \approx 2.19$ corresponds to the optimum of the function R.

Fig. 13 shows the evolution of the permutation parameter $(\beta_{t,i})_{i\in[n]}$ for agent $t = 3$. Only model 2B was able to discriminate the "good" machine to be allocated to that agent ($\tau^*_3 = 8$), while model 2A tends to selecte a non-optimal machine (here machine $i = 1$ is most likely to be allocated to agent $t = 3$). When comparing the allocation order $\hat{\tau}^{2,A}$ and $\hat{\tau}^{2,B}$ learned by both models, only model 2B matches the optimal order τ^* whereas $\hat{\tau}^{2,A} \neq \tau^*$.

The suboptimality of the limit of model 2A can be seen by comparing its end policy. Fig. 14 shows the end allocation policy for both models. Model 2A does not converge to the Dirac measure in τ^*, even if it does approach a Dirac measure, whereas model 2B converges as in the previous section.

Fig. 15 shows the evolution of machine $i = 6$'s action parameters $(\theta_{i,a})_{a\in[k_i]}$ and $(\theta_{i,t,a})_{a\in[k_i]}$ for model 2A and model 2B respectively. For the latter, we took the agent $t = 4$, as it was the best match for machine $i = 6$.

In addition to the fact that the order did not converge to the optimal order τ^*, the best action available to each agent was not chosen all the time by model 2A. In this example, we action $a = 2$ is the most likely, rather than the optimal action $a = 5$. In contrast, for model 2B, the best action ($a = 6$) is the most likely to be chosen.

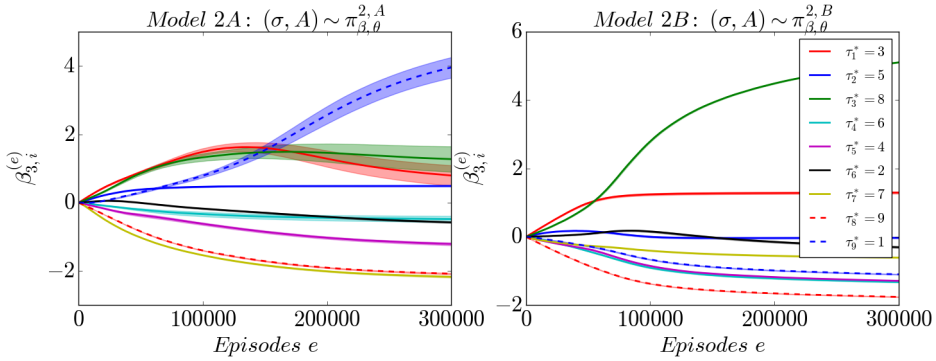

Figure 13: Evolution of $(\beta_{t,i})_{i\in[n]}$ on model 2A (left) and on model 2B (right) for agent $t = 3$. See Fig S.9 for full visualization of $\beta_{t,i}$. Each color and/or shape corresponds to a different machine. The legends shown in the right are sorted w.r.t. the optimal allocation order τ^*.

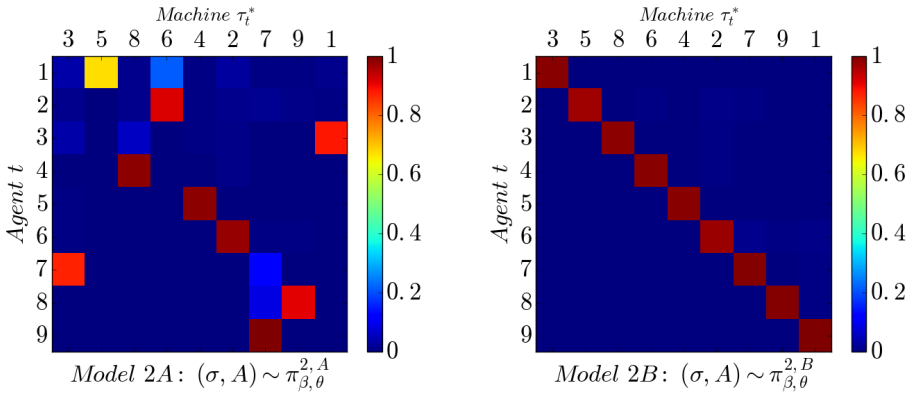

Figure 14: $\pi^{2,v}_{\hat{\beta},\hat{\theta}}(\sigma_t = i)_{(t,i)\in[n]^2}$, $v \in \{A, B\}$. Lines correspond to the agents $t \in [n]$ and the columns correspond to the machines $i \in [n]$. The columns are sorted w.r.t. the optimal allocation order τ^*. See Fig S.10 for full visualisation of $\pi^{2,v}_{\beta^{(e)},\theta^{(e)}}(\sigma_t = i)_{(t,i)\in[n]^2}$ for all episodes e.

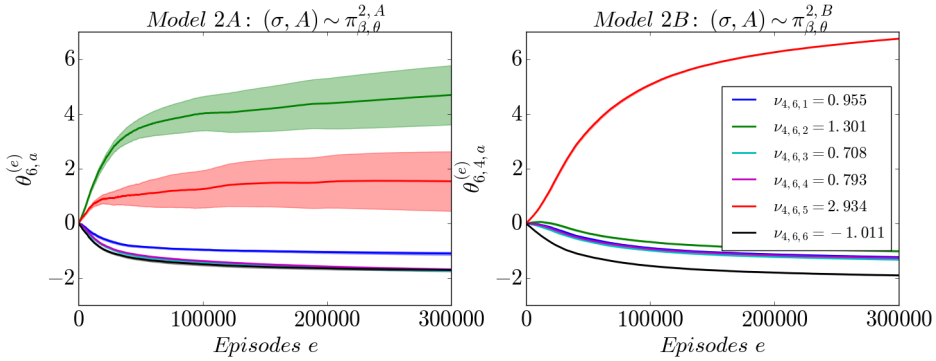

Figure 15: Evolution of $(\theta_{i,a})_{a\in[k_i]}$ for machine $i = 6$ for model 2A (left) and $(\theta_{i,t,a})_{a\in[k_i]}$ allocated to agent $t = 4$ for model 2B (right) (See Fig S.11 for full visualization of θ on both models). Each color corresponds to a different action. The optimal action is in red.

6 Performance comparison: M1, M2 and a naive MAB

In this section, a comparison analysis is performed between the two GAtACA-Series variants designed for the agents-machine assignment problem. For comparison, a naive deterministic simulation of a multi-armed bandit is executed. As baseline, we consider the naive multi-armed bandit approach, where each possible permutation of the agents defines an arm. Since it has to see each arm at least once, it needs at least $n!$ to find the best permutation.

To facilitate a fair and unbiased comparison between the two GAtACA-Series models, the learning rates α_1 and α_2 are optimized, for M1 and M2 respectively. The objective is to find the rates α_1^* and α_2^* that ensure the fastest convergence. This ensures that any observed differences in scalability are attributable to the models themselves and not to a sub-optimal choice of learning rate. While the assumption in Theorem 3 suggests that the learning rate should satisfy the Robbins-Monro conditions [18], we use a constant learning rate to maintain a straightforward and efficient learning procedure.

Algorithm 2 illustrates the iterative process of searching the optimal learning rate for two models. Since, as mentioned stated in Section 5.1, Model 1 works for a specific class of reward functions (see Table 1), we will consider the reward function given by Eq. (7).

The algorithm begins with a small initial learning rate α_0 and conducts K independent simulations. Each simulation terminates when either the maximum number of episodes is reached (indicating lack of convergence) or when the probability of assigning agent t to machine τ_t^*, for all $t \in [n]$, approaches 1 (within a margin of ε), signifying convergence. Upon convergence, the learning rate is incremented by a small step size $\eta > 0$. The algorithm stops when it encounters an episode with a given learning rate α for which the number of episodes exceeds the predefined maximum.

Algorithm 2 Find Best Learning Rate α_u^*.
$(n, u) \in \mathbb{N}^* \times \{1, 2\}$, $\varepsilon > 0$, $\gamma > 0$, $\alpha_0 > 0$, $\eta > 0$, $e_{\max} \in \mathbb{N}^*$, $K \in \mathbb{N}^*$

initialize: $\tau^* \in \mathfrak{S}(n)$, $\alpha \leftarrow \alpha_0$, $E \leftarrow Null$
while $True$ **do**
  $E' \leftarrow []$
  for k **in** $1 : K$ **do**
    initialize: $e = 1$, $\beta^{(e)} = \mathbf{0}$, $B^{(1)} = 0$
    while $e \leqslant e_{\max}$ **and** $1 - \Phi_{\tau^*_{t:n}}(\beta_t^{(e)}, \tau_t^*) < \varepsilon$, $\forall t \in [n]$ **do**
      sample $\sigma^{(e)} \sim \pi^u_{\beta^{(e)}}$
      get $R^{(e)} = \dfrac{1}{n} \displaystyle\sum_{t \in [n]} \mathbb{1}_{\sigma_t^{(e)} = \tau_t^*}$
      update $\beta^{(e+1)} \leftarrow \beta^{(e)} + \alpha \, \mathrm{Grad}(\sigma^{(e)}, \beta^{(e)}, R^{(e)} - B^{(e)})$
      update $B^{(e+1)}$ # as in Algo. 1, line 6.
      $e+ = 1$
    end while
    if $e > e_{\max}$ **then**
      return $\overline{e} = \mathrm{SUM}(E)/K$ **and** $\alpha_u^* = \alpha$
    else
      E'.append$(e - 1)$
    end if
  end for
  $\alpha \leftarrow \alpha + \eta$, $E \leftarrow E'$
end while

Fig. 16 illustrates the comparative analysis between the two models M1 and M2. For each n, the iterative learning rate search was started at $\alpha_0 = 0.1$, conducting $K = 1000$ simulations. For each $u \in \{1, 2\}$, $\alpha_u > 0$ and $n \in \mathbb{N}^*$, write $\overline{e}(\alpha_u, n)$ the average number of episodes ran by model u before stopping across the K simulations. Subsequently, the learning rate was incremented by $\eta = 0.1$. For each n, let $e_{\max}$ be twice the maximum number of episodes required to converge when $\alpha = \alpha_0$, *i.e.*, $e_{\max} = 2 \times \max\{E_k^{\alpha_0}, k \in [K]\}$, where E_k^α is the number of episode the algorithm ran before stopping during simulation k with learning rate α. The optimal learning rate differs between the two models: $\alpha_1^* < \alpha_2^*$. Moreover, upon employing their respective optimal rates, Model 2 exhibits a faster convergence rate compared to M1, and both reach the convergence criterion in polynomial time rather than the factorial of the naive multi-armed bandit (MAB) scenario.

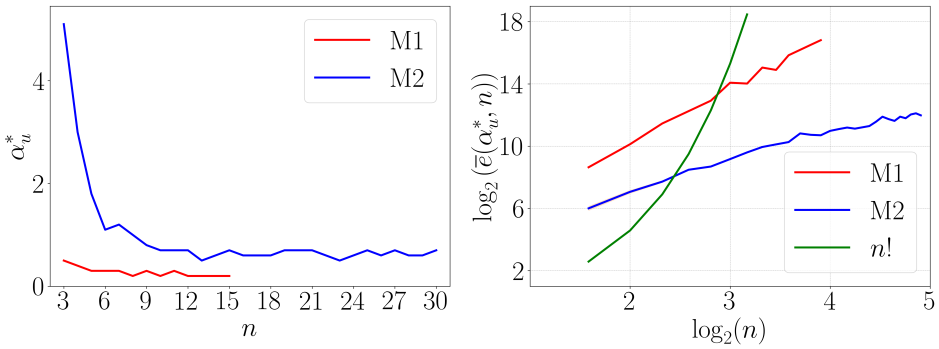

Figure 16: Comparison between $M1$ and $M2$. *Left*: optimal learning rate α_u^* w.r.t. the number of agents/jobs n. As n varies, the best-suited learning rate α^* is determined through Algo. 2. *Right*: the average number of episodes to reach the convergence criteria for each model using best α_u^* w.r.t. the number of agents/jobs n.

7 Conclusion

We proposed a formalization of the multi-agent sequential learning problem. Different reward functions and models of policy for sampling the allocation of machines and actions have been proposed:

- Models 1 and 2, which sample the permutation describing the allocation of machines. In model 1, the same permutation credits β were used throughout the assignment of the agents. In model 2, each step of the assignment process uses a different credit vector β_t. The optimization of the permutation credits were decoupled between individual time steps, making the optimization more flexible than in model 1. Also, as shown by the simulations, the algorithm based on model 1 does not converge to an optimal policy in some situations, whereas the one based on model 2 converges.
- Models 2A and 2B, which sample the agents' actions based on the allocation permutation. In model 2A, the policy of the agent depends only on the machine it has been allocated, not on who the agent is. In model 2B, the policy of the agent depends on the machine it has been allocated as well as who the agent is.
- Several rewards based on Multi-agent Multi-Armed Bandits.

We rather recommend model 2B, as it converges in more cases and seems to converge more quickly than model 2A, based on the simulations we have performed and shown in Fig. 12. This is an empirical result, which would require further study of the rates of convergence.

Another interesting generalization would concern the case where each agent evolves in an environment and changes states according to the choice (τ, A) of both machine allocation and action.

Acknowledgments

This work has been supported by the French government, through the UCA^{JEDI} Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-01 and by the Projet de Recherche Inter-instituts Multi-Equipes (80|PRIME) CNRS. It is part of the Computabrain project and the eXplAIn CNRS research team.

References

[1] Dirk G Cattrysse and Luk N Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European journal of operational research*, 60(3):260–272, 1992.

[2] Daniel Hulse, Kagan Tumer, Christopher Hoyle, and Irem Tumer. Modeling multidisciplinary design with multiagent learning. *AI EDAM*, 33(1):85–99, 2019.

[3] Yan Jin and Raymond E Levitt. i-agents: Modeling organizational problem solving in multi-agent teams. *Intelligent Systems in Accounting, Finance and Management*, 2(4):247–270, 1993.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, may 2002.

[5] Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.

[6] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[7] Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[8] Lucian Busoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 38(2):156–172, 2008.

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021.

[10] Guillaume Bono, Jilles Steeve Dibangoye, Laëtitia Matignon, Florian Pereyron, and Olivier Simonin. Cooperative multi-agent policy gradient. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 459–476, Cham, 2019. Springer International Publishing.

[11] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[12] Yueheng Li, Guangming Xie, and Zongqing Lu. Difference advantage estimation for multi-agent policy gradients. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13066–13085. PMLR, 17–23 Jul 2022.

[13] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[14] Safwan Hossain, Evi Micha, and Nisarg Shah. Fair algorithms for multi-agent multi-armed bandits. *Advances in Neural Information Processing Systems*, 34:24005–24017, 2021.

[15] Oussama Sabri, Luc Lehéricy, and Alexandre Muzy. Multi-agent learning via gradient ascent activity-based credit assignment. *Scientific Reports*, 13(1):15256, Sep 2023.

[16] Ping Xuan and Victor Lesser. Multi-agent policies: from centralized ones to decentralized ones. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1098–1105, 2002.

[17] Dimitri P. Bertsekas and John N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[18] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.