# Reinforcement Learning for DEC-MDPs with Changing Action Sets and Partially Ordered Dependencies

# (Short Paper)

Thomas Gabel and Martin Riedmiller
Neuroinformatics Group, Department of Computer Science, Institute of Cognitive Science
University of Osnabrück, 49069 Osnabrück, Germany
{thomas.gabel|martin.riedmiller@uos.de}

## ABSTRACT

Decentralized Markov decision processes are frequently used to model cooperative multi-agent systems. In this paper, we identify a subclass of general DEC-MDPs that features regularities in the way agents interact with one another. This class is of high relevance for many real-world applications and features provably reduced complexity (NP-complete) compared to the general problem (NEXP-complete). Since optimally solving larger-sized NP-hard problems is intractable, we keep the learning as much decentralized as possible and use multi-agent reinforcement learning to improve the agents' behavior online. Further, we suggest a restricted message passing scheme that notifies other agents about forthcoming effects on their state transitions and that allows the agents to acquire approximate joint policies of high quality.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

## General Terms

Algorithms, Design, Theory

## Keywords

Decentralized MDPs, Interaction, Communication

## 1. INTRODUCTION

Research on distributed control of cooperative multi-agent systems has received a lot of attention during the past years. Among the models discussed in the literature, the DEC-MDP framework [4], that is characterized by each agent having only a partial view of the global system state, has been frequently investigated. In this regard, it has been shown that the complexity of general DEC-MDPs is NEXP-complete, even for the benign case of two cooperative agents.

Decentralized decision-making is required in many real-life applications. Examples include distributed sensor networks, teams of autonomous robots, or production planning and optimization scenarios. Being important for practice, the enormous computational complexity of solving DEC-MDPs conflicts with the fact that real-world tasks do typically have a considerable problem size. Therefore, in this paper we will identify a subclass of general DEC-MDPs that features regularities in the way the agents interact with one another. For this class, we can show that the complexity of optimally solving an instance of such a DEC-MDP is provably lower (NP-complete) than the general problem (Section 2). Moreover, we analyze methods for the agents to benefit from partially knowing about the state transition dependencies. To this end, we propose the use of a restricted message passing scheme that notifies other agents about forthcoming effects on their state transitions and we investigate its usefulness (Section 3). For adapting the agents' policies, we propose the usage of a multi-agent reinforcement learning (RL) approach, where the agents are independent learners and do their learning online which we evaluate in the context of larger-sized scheduling problems (Section 4).

**Cite as:** Reinforcement Learning for DEC-MDPs with (...) (Short Paper), Thomas Gabel, Martin Riedmiller, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp.1333-1336.


## 2. PROBLEM DESCRIPTION

### 2.1 DEC-MDP Framework

Basically, the subclass of problems we are focusing on in this paper may feature an arbitrary number of agents whose actions influence, besides their own, the state transitions of maximally one other agent in a specific manner. We embed the problem settings of our interest into the framework of decentralized Markov decision processes (DEC-MDP) [4].

*Definition 1.* A factored $m$-agent DEC-MDP $M$ is defined by a tuple

$$\langle Ag, S, A, P, R, \Omega, O \rangle$$

where $Ag = \{1, \ldots, m\}$ is a set of agents, $S$ is the set of world states that can be factored into $m$ components $S = S_1 \times \cdots \times S_m$ ($S_i$ belong to one of the agents each), $A = A_1 \times ... \times A_m$ is the set of joint actions to be performed by the agents ($a = (a_1, \ldots, a_m) \in A$ denotes a joint action that is made up of elementary actions $a_i$ taken by agent $i$), $P$ is the transition function with $P(s'|s, a)$ denoting the probability that the system arrives at state $s'$ upon executing $a$ in $s$, $R$ is a reward function with $R(s, a, s')$ denoting the reward for executing $a$ in $s$ and transitioning to $s'$.

$\Omega = \Omega_1 \times \cdots \times \Omega_m$ is the set of all observations of all agents ($o = (o_1, \ldots, o_m) \in \Omega$ denotes a joint observation with $o_i$ as the observation for agent $i$) and $O$ denotes the observation function that determines the probability $O(o_1, \ldots, o_m|s, a, s')$ that agent 1 through $m$ perceive observations $o_1$ through $o_m$.

Moreover, $M$ is jointly fully observable, i.e. the current state is entirely determined by the amalgamation of all agents' observations: if $O(o|s,a,s') > 0$, then $Pr(s'|o) = 1$.

We refer to the agent-specific components $s_i \in S_i$, $a_i \in A_i$, and $o_i \in \Omega_i$ as the local state, action, and observation of agent $i$. A joint policy $\pi$ is a set of local policies $\langle \pi_1, \ldots, \pi_m \rangle$ each of which is a mapping from agent $i$'s sequence of local observations to local actions, i.e. $\pi_i : \overline{\Omega}_i \to A_i$. In the following, we allow each agent to fully observe its local state, i.e. we consider factored $m$-agent DEC-MDPs with local full observability which implies that for all agents $i$ and for all local observations $o_i$ there is a local state $s_i$ such that $Pr(s_i|o_i) = 1$. Note that joint full observability and local full observability of a DEC-MDP do generally not imply full observability, which would allow us to consider the system as a single large MDP and to solve it with a centralized approach.

A factored $m$-agent DEC-MDP is called *reward independent*, if there exist local functions $R_1$ through $R_m$, each depending on local states and actions of the agents only, as well as a function $r$ that amalgamates the global reward value from the local ones, such that maximizing each $R_i$ individually also yields a maximization of $r$. Throughout this paper, we will consider the global reward to be the sum of the local ones.

If, in a factored $m$-agent DEC-MDP, each agent's observation depends only on its current and next local state and on its action, then that DEC-MDP is called *observation independent*. Then, in combination with local full observability, the observation-related components $\Omega$ and $O$ are redundant and can be removed from Definition 1.

While the DEC-MDPs of our interest are observation independent and reward independent, they are *not* transition independent. That is, the state transition probabilities of one agent may very well be influenced by another agent.

## 2.2 Variable Action Sets

We assume that there are some regularities that determine the way local actions exert influence on other agents' states. First, we assume that the sets of local actions $A_i$ change over time.

*Definition 2.* A factored $m$-agent DEC-MDP is said to feature *changing action sets*, if the local state of agent $i$ is fully described by the set of actions currently selectable by $i$ ($s_i = A_i \setminus \{\alpha_0\}$) and $A_i$ is a subset of the set of all available local actions $\mathcal{A}_i = \{\alpha_0, \alpha_{i1} \ldots \alpha_{ik}\}$, thus $S_i = \mathcal{P}(\mathcal{A}_i \setminus \{\alpha_0\})$. Here, $\alpha_0$ represents a null action that does not change the state and is always in $A_i$. We abbreviate $\mathcal{A}_i^r = \mathcal{A}_i \setminus \{\alpha_0\}$.

Concerning state transition dependencies, one can distinguish between dependent and independent local actions. While the former influence an agent's local state only, the latter may additionally influence the state transitions of other agents. As pointed out, our interest is in non-transition independent scenarios. In particular, we assume that an agent's local state can be affected by an arbitrary number of other agents, but that an agent's local action affects the local state of maximally one other agent.

*Definition 3.* A factored $m$-agent DEC-MDP is said to have *partially ordered transition dependencies*, if there exist functions $\sigma_i$ for each agent $i$ with

1. $\sigma_i : \mathcal{A}_i^r \to Ag \cup \{\emptyset\}$ and
2. $\forall \alpha \in \mathcal{A}_i^r$ the directed graph $G_\alpha = (Ag \cup \{\emptyset\}, E)$ with $E = \{(j, \sigma_j(\alpha)) | j \in Ag\}$ is acyclic and contains a path of length $m$

and it holds $P(s_i'|s,(a_1 \ldots a_m),(s_1' \ldots s_{i-1}', s_{i+1}' \ldots s_m')) = P(s_i'|s_i, a_i, \{a_j \in \mathcal{A}_j | i = \sigma_j(a_j), j \neq i\})$.

The influence exerted on another agent always yields an extension of that agent's action set: If $\sigma_i(\alpha) = j$, $i$ takes local action $\alpha$, and the execution of $\alpha$ has been finished, then $\alpha$ is added to $A_j(s_j)$, while it is removed from $A_i(s_i)$.

So, the $\sigma_i$ functions indicate whose other agents' state is affected when agent $i$ takes a local action. Also, condition 2 in Definition 3 implies that for each local action $\alpha$ there is a total ordering of its execution by the agents. While these orders are total, the global order in which actions are executed is only partially defined by that definition and subject to the agents' policies. Lemma 1 states that for the problems considered any local action may appear only once in an agent's action set and, thus, may be executed only once.

*Lemma 1.* In a factored $m$-agent DEC-MDP with changing action sets and partially ordered transition dependencies it holds: $\forall i \in Ag$, $\forall \alpha \in \mathcal{A}_i^r$, $\forall t \in \{1 \ldots T\}$ and $\forall \overline{s}_i = (s_i^1 \ldots s_i^t)$: If there is a $t_a$ ($1 \leq t_a < T$) with $\alpha \in s_i^{t_a}$ and a $t_b$ ($t_a < t_b \leq T$) with $\alpha \notin s_i^{t_b}$, then $\forall \tau \in \{t_b \ldots T\} : \alpha \notin s_i^\tau$.

Proof. The proofs of this and of the following lemmas are omitted due to space constraints. □

## 2.3 Implications on Complexity

While the complexity of solving general DEC-MDPs is known to be NEXP-complete [4], several authors have identified subclasses of the general problem that provably yield lower (NP-complete) complexity (e.g. [3, 6, 2]). As shown in [9], a key factor that determines whether the problem complexity is reduced to NP-completeness is whether the agents' histories can be compactly represented. In particular, there must exist an encoding function $Enc_i : \overline{\Omega}_i \to E_i$ such that

1. a joint policy $\pi = \langle \pi_1 \ldots \pi_m \rangle$ with $\pi_i : E_i \to \mathcal{A}_i$ is capable of maximizing the global value and
2. the encoding is polynomial, i.e. that $|E_i| = O(|S|^{c_i})$.

For our class of factored $m$-agent DEC-MDPs with changing action sets and partially ordered transition dependencies we can define an encoding that adheres to both of these conditions, thus showing that those problems are NP-complete.

The interaction history of a DEC-MDP is the sequence of local observations $\overline{o}_i \in \overline{\Omega}_i$ which in our case corresponds to the history of local states $\overline{s}_i \in \overline{S}_i = \times_{t=1}^{T} S_i$, since we assume local full observability (recall that $S_i = \mathcal{P}(\mathcal{A}_i^r)$).

*Definition 4.* Given a local action set $\mathcal{A}_i = \{\alpha_0 \ldots \alpha_k\}$ and a history $\overline{s}_i = (s_i^1 \ldots s_i^t) \in \overline{S}_i$ of local states of agent $i$, the encoding function is defined as $Enc_i : \overline{S}_i \to E_i$ with $E_i = C_{\alpha_1} \times \cdots \times C_{\alpha_k}$ and $C_{\alpha_j} = \{0, 1, 2\}$. And it holds $Enc_i(\overline{s}_i) = (c_{i,\alpha_1} \ldots c_{i,\alpha_k}) \in E_i$ with

$$c_{i,\alpha_j} = \begin{cases} 0 & \text{if } \nexists \tau \text{ with } \alpha_j \in s_i^\tau \\ 1 & \text{if } \alpha_j \in s_i^t \\ 2 & \text{else} \end{cases}$$

Basically, the encoding guarantees that each agent knows whether some local action has not yet been, is currently, or had been in its action set. Proving that this encoding is capable of representing the optimal policy and showing that it is a polynomial encoding, we can conclude that the subclass of DEC-MDPs we identified is NP-complete.

*Lemma 2.* $Enc_i$ provides a polynomial encoding of agent $i$'s observation history.

*Lemma 3.* $Enc_i$ provides an encoding of agent $i$'s observation history such that a joint policy $\pi = \langle \pi_1 \dots \pi_m \rangle$ with $\pi_i : E_i \rightarrow \mathcal{A}_i$ is sufficient to maximize the global value.

As deciding a polynomially encodable DEC-MDP is NP-hard [9], solving a factored $m$-agent DEC-MDP with changing action sets and partially ordered dependencies is so, too.

# 3. RESOLVING DEPENDENCIES

Besides using an encoding of an agent's interaction history (Section 2), there are other options for exploiting the regularities in the transition dependencies of the class of DEC-MDPs we identified that.

## 3.1 Reactive Policies and Their Limitations

An agent taking its action based solely on its most recent local observation $s_i \subseteq \mathcal{A}_i$ is in general not able to contribute to optimal joint behavior: It faces difficulties in assessing the value of its idle action $\alpha_0$. Because a purely reactive agent has no information related to other agents and dependencies at all, it is incapable of properly distinguishing when it is favorable to remain idle and when not. For these reasons, we exclude $\alpha_0$ from all $\mathcal{A}_i$ for purely reactive agents.

*Definition 5.* For an $m$-agent DEC-MDP with changing action sets and partially ordered transition dependencies, a *reactive policy* $\pi^r = \langle \pi_1^r \dots \pi_m^r \rangle$ consists of $m$ reactive local policies with $\pi_i^r : S_i \rightarrow \mathcal{A}_i^r$ where $S_i = \mathcal{P}(\mathcal{A}_i^r)$.

That is, purely reactive policies always take an action $\alpha \in \mathcal{A}_i(s_i) = s_i$ (except for $s_i = \emptyset$), even if it was better to stay idle and wait for a transition from $s_i$ to some $s_i' = s_i \cup \{\alpha'\}$ induced by another agent, and then execute $\alpha'$ in $s_i'$.

## 3.2 Awareness of Dependencies

In Definition 4, we stated that the probability that agent $i$'s local state moves to $s_i'$ depends on that agent's current local state $s_i$, its action $a_i$, as well as on the set $\{a_j \in \mathcal{A}_j | i = \sigma_j(a_j), i \neq j\} =: \Delta_i$, i.e. on the local actions of all agents that may influence agent $i$'s transition. Although knowing $\Delta_i$ is in general not feasible for each agent, we may enhance the capabilities of a reactive agent $i$ by allowing it to get at least some partial information about this set. For this, we extend a reactive agent's local state space from $S_i = \mathcal{P}(\mathcal{A}_i^r)$ to $\hat{S}_i$ such that for all $\hat{s}_i \in \hat{S}_i$ it holds $\hat{s}_i = (s_i, z_i)$ with $z_i \in \mathcal{P}(\mathcal{A}_i^r \setminus s_i)$. So, $z_i$ is a subset of the set of actions currently *not* in the action set of agent $i$.

*Definition 6.* Let $1 \dots m$ be reactive agents acting in a DEC-MDP, as specified in Definition 3, whose local state spaces are extended to $\hat{S}_i$. Assume that current local actions $a_1 \dots a_m$ are taken *consecutively*. Given that agent $j$ decides for $a_j \in A_j(s_j)$ and $\sigma_j(a_j) = i$, let also $s_i$ be the local state of $i$ and $\hat{s}_i$ its current extended local state with $\hat{s}_i = (s_i, z_i)$. Then, the transition dependency between $j$ and $i$ is said to be *resolved*, if $z_i := z_i \cup \{a_j\}$.

Resolving transition dependencies according to Definition 6 means letting agent $i$ know some of those current local actions of other agents by which $i$'s local state will soon be influenced. Since, for the class of problems we are dealing with, inter-agent interferences are always exerted by changing (extending) another agent's action set, in this way agent $i$ gets to know which further action(s) will soon be available in its action set. Integrating this piece of information into $i$'s extended local state description $\hat{S}_i$, $i$ gets the opportunity to willingly stay idle (execute $\alpha_0$) until the announced action $a_j \in z_i$ enters its action set and can finally be executed.

The notification of agent $i$, which instructs him to extend its local state component $z_i$ by $a_j$, may easily be realized by a simple message passing scheme (assuming cost-free communication between agents) that allows agent $i$ to send a single directed message to agent $\sigma_i(\alpha)$ upon the local execution of $\alpha$. Obviously, this kind of partial resolving of transition dependencies is particularly useful in applications where the execution of atomic actions takes more than a single time step and where, hence, decision-making proceeds asynchronously across agents. Under those conditions, up to half of the dependencies in $\Delta_i$ (over all $i$) may be resolved.

# 4. DISCUSSION AND EVALUATION

Distributed problem solving in practice is often characterized by a factored system state description where the agents base their decisions on local observations. Also, our assumptions that local actions may influence the state transitions of maximally one other agent and that any action has to be performed only once are frequently fulfilled. Sample real-world applications include scenarios from manufacturing, production planning, or assembly line optimization, where typically the production of a good involves a number of processing steps that have to be performed in a specific order. In a factory, however, usually a variety of products is assembled concurrently, which is why an appropriate sequencing of single operations is of crucial importance for overall performance. Thus, the class of factored $m$-agent DEC-MDPs with changing action sets and partially ordered transition dependencies covers a variety of such scheduling problems, for example flow-shop and job-shop scheduling scenarios [7]. Beyond that, a big portion of supply chain problems where complex items are assembled through a series of steps are covered. Other practical application domains to which our model is of relevance include network routing (e.g. sub-task of determining the order of forwarding packets), railway traffic control (e.g. task of allowing trains to pull into the station via agent-based track switches), or workflow management.

### *Joint Policy Acquisition with RL*

Solving a DEC-MDP optimally is NEXP-hard and intractable for all except the smallest problem sizes. Unfortunately, the fact that the subclass of DEC-MDPs we identified is in NP and hence simpler to solve, does not rid us from the computational burden implied. So, our goal is not to develop yet another optimal solution algorithm applicable to small problems only, but to use a technique capable of quickly finding approximate solutions in the vicinity of the optimum.

We let the agents acquire their local policies independently of the other agents by repeated interaction with the DEC-MDP and concurrent evolvement of their policies. Our learning approach is made up of alternating data collection and learning stages that are being run concurrently within

all agents. At its core, a specialized variant of a neural fitted Q iteration (NFQ) algorithm [8], enhanced for usage in multi-agent domains, is used that allows the agents to determine a value function over their local state-action spaces. A detailed description of that approach can be found in [5].

### Experiments

For the purpose of evaluation, we focus on various job-shop scheduling (JSS) benchmark problems (taken from [1]) that are known to be NP-hard. The goal of scheduling is to allocate a given number of jobs to a limited number of resources such that some objective is optimized. In job-shop scheduling, $n$ jobs must be processed on $m$ machines in a pre-determined order, while minimizing maximum makespan $C_{max}$, which corresponds to finishing processing as quickly as possible. Each job consists of a specific number of operations that each have to be handled on a certain resource for a certain duration, where the whole job is finished after its last operation has been entirely processed.

JSS problems are suited to be modelled as factored $m$-agent DEC-MDPs with changing action sets and partially ordered transition dependencies: The world state can be factored, if we assume that to each of the resources one agent $i$ is associated whose local action is to decide which waiting job to process next. Further, the local state of $i$ can be fully described by the changing set of jobs currently waiting for further processing, and after having finished an operation of a job, this job is transferred to another resource, which corresponds to influencing another agent's local state by extending that agent's action set. Examining the formal definition of JSS problems [7], it is obvious that we can also easily define $\sigma_i : \mathcal{A}_i^r \rightarrow Ag \cup \{\emptyset\}$ (see Definition 3) for all agents/resources $i$ and that the corresponding directed graph $G_\alpha$ is indeed acyclic with a path of length $m$.

The primary concern of the experiments conducted was on an analysis of the three approaches discussed in this paper. We compared agents that independently learn purely reactive policies $\pi_i^r$ (see Section 3.1) defined over $S_i = \mathcal{P}(\mathcal{A}_i^r)$ that never remain idle when their action set is not empty (RCT), reactive policies $\hat{\pi}_i$ that are partially aware of their dependencies on other agents (being notified about forthcoming influences exerted by other agents, COM), and policies $\pi_i : E_i \rightarrow \mathcal{A}_i$ where $E_i$ is an encoding of that agent's observation history $\overline{S}_i$ according to Definition 4 (ENC).

### Findings

Using RCT-agents, only schedules from the class of non-delay schedules $\mathbb{S}_{nd}$ can be created by applying reactive policies. Since $\mathbb{S}_{nd} \subseteq \mathbb{S}_a$ and it is known that the optimal schedule is always in $\mathbb{S}_a$ [7], but not necessarily in $\mathbb{S}_{nd}$, RCT-agents can at best learn the optimal solution from $\mathbb{S}_{nd}$. By contrast, learning with ENC-agents, the optimal solution can in principle be attained, but we found that the time required by our learning approach for this to happen increases significantly due to larger-sized local state spaces.

We also found that the awareness of inter-agent dependencies achieved by partial dependency resolvements via communication in fact realizes a good trade-off between the former two approaches. On the one hand, resolving a transition dependency according to Definition 6, an agent $i$ can become aware of an incoming job. Thus, $i$ may decide to wait for that arrival, instead of starting to execute another job. Hence, also schedules can be created that are not non-delay. On the other hand, very poor policies with unnecessary idle times can be avoided, since a decision to stay idle may be taken only when a future job arrival has been announced.

Averaged over 24 different benchmark instances [1] of varying sizes (up to 15 agents) for which it is known that the optimal solution is not in $\mathbb{S}_{nd}$, the learned policies nearly reach the theoretical optimum (schedule with minimal $C_{max}$) missing it by 6.18% for RCT-agents, by 9.55% for ENC-agents, and by 4.78% for COM-agents. Dispatching rule based scheduling approaches are clearly surpassed (best conventional scheduling rule reaches a remaining error of 8.59%).

## 5. CONCLUSION

We have identified a class of cooperative decentralized MDPs that features a number of regularities in the way agents influence the state transitions of other agents. Exploiting the knowledge about these correlations, we have proven that this class of problems is easier to solve (NP-hard) than general DEC-MDPs (NEXP-hard). Subsequently, we have looked at possibilities for modeling memoryless agents and enhancing them by restricted allowance of communication. For solving instances of the DEC-MDP class identified we relied on a coordinated batch-mode reinforcement learning algorithm that facilitates the agents to concurrently and independently learn their local policies of action online.

## 6. ACKNOWLEDGEMENTS

This research has been supported by the German Research Foundation (DFG) under grant number Ri-923/2-3.

## 7. REFERENCES

[1] J. Beasley. OR-Library, 2005, `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[2] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov Decision Processes with Event-Driven Interactions. In *Proceedings of AAMAS 2004*, pages 302–309, New York, USA, 2004. ACM Press.

[3] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving Transition Independent Decentralized MDPs. *Journal of AI Research*, 22:423–455, 2004.

[4] D. Bernstein, D. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[5] T. Gabel and M. Riedmiller. Adaptive Reactive Job-Shop Scheduling with Learning Agents. *International Journal of Information Technology and Intelligent Computing*, 2(4), 2008.

[6] C. Goldman and S. Zilberstein. Optimizing Information Exchange in Cooperative Multi-Agent Systems. In *Proceedings of AAMAS 2003*, pages 137–144, Melbourne, Australia, 2003. ACM Press.

[7] M. Pinedo. *Scheduling. Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[8] M. Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proceedings of ECML 2005*, Porto, Portugal, 2005. Springer.

[9] J. Shen, R. Becker, and V. Lesser. Agent Interaction in Distributed POMDPs and Implications on Complexity. In *Proceedings of AAMAS 2006*, pages 529–536, Hakodate, Japan, 2006. ACM Press.