On the Distributivity of Multi-agent Markov Decision Processes for Mobile-Robotics

Guillaume Lozenguez

On the Distributivity of Multi-agent Markov Decision Processes for Mobile-Robotics

Guillaume Lozenguez[1]

IMT Lille Douai, F-59 000 Lille, FRANCE,
Instut Mines-Télécom, Univ. Lille,
guillaume.lozenguez@imt-lille-douai.fr

Abstract. Today, mobile robot systems are designed for open public environments and can benefit from modeling uncertainties in their decision processes. For this reason, approaches based on Markov Decision Processes (*MDP*) are proposed to control autonomous robots since few decades. Optimally solving an *MDP* based model to coordinate a fleet of robots can rapidly become intractable. This paper focuses on the distributive property of Multi-agent Markov Decision Processes (*MMDP*) and identify an *MMDP* model as "Completly Distributable", "Partially Distributable" or "Distributable with Coordination". Based on those definitions, the classical multi-mobile-robot problems addressed in the literature are classified and discussed accordingly regarding the potential to compute an optimal solution in a distributive way.

Keywords: Distributed computation, MDP, Mobile Robotics

1 Introduction

Today, mobile robot systems are designed for usage in open public spaces (homes, commercial centers, hospitals, offices, museums, streets, etc.). Robot decision-making needs to handle the uncertainty that such environments implies (potentially on: action outcome, time windows, dynamic environments, interactions, etc.). Markov Decision Process (*MDP*) frameworks represent powerful tools to model control possibilities over systems evolving under uncertainty. Such models permit theoretically to compute an optimal policy of actions to perform while considering all the trajectories the system could take.

MDP frameworks (and more specifically Decentralized Partially Observable MDP - Dec-POMDP) allow to model decentralized systems; systems where perceptions and action capabilities are distributed on agents in the environment *Dec-POMDP* models the stochastic system evolution and the stochastic agent perception capabilities. The complexity to solve *Dec-POMDPs* prevents from computing optimal decentralized policies in real applications [14]. A Multi-agent Markov Decision Process *MMDP* [4] takes advantage of fully observable system states to drastically reduce the complexity. The model matches decentralized systems with efficient communication, where each agent could have, at each time-step, an efficient knowledge of the overall situation.

However, more and more applications experiment MDP based approaches to control decentralized systems evolving under uncertainty and mobile robots is an emblematic one. Considering a fleet of robots deployed in the environment, the goal is to plan and coordinate their movements to achieve a common mission [6, 18, 17, 15]. In top-down approaches, the idea is to identify structures in the problem definition to constrain and speedup the solving [7, 3]. Potentially, the structure can be decomposed into independent sub-processes (allowing distributable optimizations). However, the use of *MMDP* to optimize decentralized controls remains limited to scale real applications as it is recently highlighted in [9] to control water-way network or in [12] for collision free path planning.

Bottom-up approaches start with the assumption that the robots of a fleet are equipped with computing resources. Those approaches rely on Agent-Based Models (*ABM*) to distribute the policy computations over all the robots. THe main idea is to allocate an *MDP* to each agent/robot allowing it to compute its own policy with interesting empirical results [6, 17]. However, The union of all *MDPs* are incomplete to model the global problem and could not guarantee an optimal solution. "Heuristic" *MDP* based approaches are more about handling dynamic and uncertain environments than computing an optimal joint-policy.

In this paper, we are interested in the capability to distribute *MMDPs* by proposing a classification as "Completly Distributable", "Partially Distributable" and "Distributable with Coordination" problems. This position paper is organized as follows: Section 2 presents the frameworks based on *MDPs* to address multi-robot decision-making under uncertainty. Section 3 develops the properties that permit a classification over distributability. In Section 4, a discussion about the state-of-the-art approaches based on *MDPs* approaches for multi-robot systems, and finally a conclusion is presented in Section 5.

2 Multi-Robot Decision Making Under Uncertainty

Planning in mobile robot applications mainly relies on path or trajectory planning that ends up in Traveling Salesman Problems (*TSP*). In multi-robot context, path planning requires to take congestion problems or tasks and resource allocation into account. Solving those problems in a deterministic case is already challenging [23] and even more in a stochastic case.

Because of the difficulty to solve and learn *Dec-POMDPs*, its usage for real multi-robot scenarios is limited. This paper focus on fully observable Markov Decision Process by investigating factorized properties.

2.1 Planning and Uncertainty in Mobile Robotics

Uncertainty is omnipresent in robotics applications due to the perception and action loop in the real world, the lack in knowledge and/or the uncontrolled stochastic events. Markov Decision Process *MDP* is a stochastic automaton modeling the dynamics of a system under uncertainty regarding action possibilities. Multi-agent *MDPs* (*MMDPs*) increase the model by considering that agents performed actions simultaneously.

Multi-agent *MDP* is a tuple $\langle S, J, t, r\rangle$ where S is the set of system states, J the set of joint-actions and t and r are respectively the transition and reward functions. The transition function models the probable dynamics of the system. $t(s, j, s')$ return the probability to end in state s' by doing the joint-action j in state s. The reward function evaluates the interest to perform the joint-action j in the state s ($r(s, j) \in \mathbb{R}$).

In applications, states and joint-actions are defined by variables with finite domains. The state space results from the Cartesian product of state variables $X_i \in \mathbb{X}$. It has an exponential size over the number of variables (e.g. $|S| = 2^{|\mathbb{X}|}$ in the case of binary variables with $|\mathbb{X}|$ state variables).

$$s = (x_1,\ x_2 \ldots, x_{|\mathbb{X}|}) \in S, \quad S = \prod_{X_i \in \mathbb{X}} X_i, \quad |S| \geq 2^{|\mathbb{X}|} \tag{1}$$

Similarly to state definition, the joint-actions result from the Cartesian product of the action variables. Each action variable $A_i \in \mathbb{A}$ matches a control lever or actuator in the overall systems. To notice that agents can be responsible for several action variables (for instance, rotation and linear speeds of non-honolomic robots).

$$j = (a_1,\ a_2 \ldots, a_{|\mathbb{A}|}) \in S, \quad J = \prod_{A_i \in \mathbb{A}} A_i, \quad |A| \geq 2^{|\mathbb{A}|} \tag{2}$$

Solving an *MMDP* consists in allocating an join-action to perform for each state in a policy function $\pi(s)$ by optimizing the expected gain. In formal terms, the expected gain can be modeled by the Bellman equation.

$$V^{\pi}(s) = r(s, j) + \gamma \sum_{s' \in S} t(s, j, s') V^{\pi}(s'), \quad \text{with } j = \pi(s) \tag{3}$$

With an infinite horizon, the ratio $\gamma \in [0, 1]$ balances between immediate reward and future gains, γ is generally chosen close to 1.

By using *MMDP* to model multi-robot coordination problems, the number of state variables increases with the number of robots (e.g. the position of each robot). Controlling multi-robots in uncertain environments induces intractable problems [18].

2.2 Factorized Model in Multi-agent Context

Factorized *MDP* take advantage of structure in the transition and reward functions to speed-up the policy computation [5]. Discriminant state variables are addressed first and potentially overweight the other variables in the decision process. Those techniques permit solving *MDPs* without a complete enumeration over all the states.

Multi-agent systems are characterized by sub-systems working together. That for, *MMDPs* includes Transition and reward functions (t and r), in *MMDPs*, are defined in a factorized way with two sets T and R of transition probabilities and reward criteria defined on subsets of state and action variables.

This way, a transition probability $P \in T$ returns the probabilistic assignment of target variables st'_P depending on parent variables sp'_P and action variables j_P. $P(st'_P|sp_P, j_P)$ is the probability that target state variables would be on sp'_P at the next time step knowing that current parent state and action variables are sp_P and j_P. The transition function results from the product of transition probabilities (sub-transitions):

$$t(s, j, s') = \prod_{P \in T} P(st'_P|sp_P, j_P), \quad \text{with } P : ST_P|SP_P \times J_P \rightarrow [0, 1] \tag{4}$$

This way, state variables can be aggregated in subsets where each subset matches a part of the system affected only by a subset of actions. For example, the pose of a robot (its position and orientation variables) evolves according to its control. 3 variables (x_i, y_i, o_i) per robot i would evolve according to 2 actions (linear speed and rotation).

R. Becker et al [2] proposed transition independent model where the actions of each robot only impact its own variables. Varakantham et al. [22] presented a more generalized approach by taking advantage of shared and private state variables. Transition-independent approach was adapted to *MMDP* and tested on intrusion detection scenarios [21]. However, in a generic form, a transition probability P (or a sub-transition) is defined by three subsets: the target (or controlled) state variables $\mathbb{XT}_P \subseteq \mathbb{X}$ (reached at time $t + 1$), the parent state variables $\mathbb{XP}_P \subseteq \mathbb{X}$ (at time t) and the action variables $\mathbb{A}_P \subseteq \mathbb{A}$.

$$ST_P = \prod_{X_j \in \mathbb{XT}_P} X'_j, \quad SP_P = \prod_{X_j \in \mathbb{XP}_P} X_j, \quad J_P = \prod_{A_j \in \mathbb{A}_P} A_j \tag{5}$$

The target and parent state variables are not necessarily the same and some parent variables can be shared between several sub-transitions. For instance, door state variables (*open* or *closed*) could affect the navigation of all the robots, in indoor environment. However, a state variable x_i is defined in the target set of one and only one sub-transitions.

$$\bigcup_{P \in T} ST_P = S, \quad \forall (P1, P2) \in T^2, \quad P1 \neq P2 \;\Rightarrow\; ST_{P1} \cap ST_{P2} = \emptyset \tag{6}$$

Each transition component P marks a sub-process to control and in a similar way, the reward function could be factorized according to the sub-evaluations. The reward function will be defined as a sum of sub-rewards (identified as criteria $C \in R$) applied on subsets of the state and action variables. For instance, in multi-robot missions, movement cost criteria could be defined independently for each robot.

$$r(s, j) = \sum_{C \in R} C(s_C, j_C), \quad C : S_C \times J_C \rightarrow \mathbb{R} \tag{7}$$

2.3 Interdependent *MMDP* sub-processes

sub-processes are linked together by sharing common variables. A sub-process $P1$ depends on another sub-process $P2$ if one of its target variables is a parent variable of $P2$. Those variables can be identified as the output of $P1$ (the

variables controlled by $P1$ and useful to another sub-process) and the input of $P2$ (the variables influencing $P2$). The sub-processes together form an oriented graph of dependencies (edges) between sub-processes (nodes).

Factorized *MDP* is directly applicable when sub-transitions and sub-rewards follow the same decomposition (i.e. all state and action variables of each sub-reward are included in a unique sub-transition actions and target variables). In other terms, each reward criteria can be attached to a sub-process node of the sub-process dependency graph.

Approaches based on structured models take advantage of low interaction between agents to reduce the complexity to solve them [3, 7, 10]. The theoretical work presented in [11], focuses on factorized *MMDP* by proposing a message-based agent optimizations. Each agent is responsible for a sub-process and solves it iteratively by exchanging values over shared variables. In fact, agent is requiring information from its dependent sub-processes to control its output variables in a efficient way. This family of algorithms permits for instance to handle cycles in the sub-process dependency graph.

Distributing a *MDP* solving generally gives the capacity to each agent to compute its own policy in a cooperative way [8, 19]. However, solving factorized *MMDP* is not necessarily distributable depending on the structure of it sub-process dependency graph. More importantly, the alignment constraint over sub-transition and sub-reward is restrictive when applied on realistic scenarios. For instance, when the reward depends on the overall duration of the mission, the optimal plan for a robot is impacted by the plan of the robot with the worst duration. The distributivity of *MMDP*s is not obvious if all the robots are bound together as in task allocation (multiple traveling salesman problem [20, 15]) or in collision free path planning (multiple path planning problem [23]).

3 Distributivity of an *MMDP*

This paper investigates the possibility to distribute the policy computation by splitting a large *MMDP* into smallest, almost independent *sub-MMDP*s solvable in a parallelized way. The idea is to classify *MMDP*s over three levels of distributivity: *complete*, *partial* or *coordinated*. This section proposes a formal definition of that distributivity before a discussion, in the next section, regarding classical mobile robot applications.

3.1 Complete Distributivity

Complete Distributivity is achieved if a global *MMDP* can be decomposed into several independent and smaller *MMDPs*, defined on subsets of state and action variables. The main idea is to decompose the value function of a global *MMDP* model M into a sum of several independent value functions. Each subset of the state and action variables defines a sub *MMDP* model M^α, allocated to an agent $\alpha \in Ag$, (Ag set of all the agents) that will be responsible of solving it.

$$M \equiv \bigcup_{\alpha \in Ag} M^\alpha, \quad V^\pi(s) \equiv \sum_{\alpha \in Ag} V^{\alpha\pi}(s^\alpha), \quad \pi = \bigcup_{\alpha \in Ag} \alpha\pi \tag{8}$$

There is an equivalence between a global *MMDP* and a collection of sub-*MMDP* if optimally solving the global model corresponds to optimizing the bellman equation of all the sub-*MMDP*s.

By considering a Factored-*MMDP* each sub-*MMDP* model M^α is built over a set of sub-processes. M^α is built over the sub-transitions and sub-rewards associated to the bound sub-processes.

$$t^\alpha(s^\alpha, j^\alpha, s'^\alpha) = \prod_{P \in T^\alpha} P(st_P | sp_P, j_P) \tag{9}$$

$$r^\alpha(s^\alpha, j^\alpha) = \sum_{C \in R^\alpha} C(s_C, j_C) \tag{10}$$

Each sub-model M^α is required to be independent. State and action variables need to be sufficient to predict their evolution and evaluate the consequences. In other words, whatever a sub-process included in M^α, all dependent sub-processes are also included to M^α. By considering the direct graph built over the dependencies of Factored *MMDP* sub-processes, the M^α sub-*MMDP* matches a disconnected sub-graph. Searching for the distributivity of a Factored *MMDP* consist in searching disconnect part in the associated sub-process graph.

It is important to notice that the sub-models are not necessarily disjoint. The system could include state variables not controllable by any action variable, and each of then is added to all sub-models that include controllable state variables depending on it. Typically, temporal problems include a state variable modeling discreet times that would affect some elements (outdoor luminosity for instance). In the direct graph, those variables are considered only as input variables of sub-problems and will not be an output variable in any of the edges.

The *MMDP* would be classified as "completely distributable" if such a decomposition can be defined. The solving complexity is reduced in a logarithmic scale over the sizes of the global state and action space. However, "completely distributive" *MMDPs* have a very poor interest while they only model systems with components evolving without any interaction between them.

3.2 Partial Distributivity

Partial distributivity is based over the sub-process dependency graph resulting from Factorized *MMDP*. An *MMDP* is considered partly distributable if several of its sub-processes are independent enough to be solved in parallel.

A sub-process *PI* is independent of *PJ* if no one of the output variables of *PI* can affect the sub-process *PJ* in a direct way (included in the input variables of *PJ*) or a recursive way (by following all the descendants of *PI*). In other words, there is no direct path from *PI* to *PJ* in the oriented graph modeling the dependencies of the sub-processes. Solving *PI* and *PJ* in a distributable way is possible if *PI* is independent of *PJ* and vice versa.

The resolution of a Factored *MMDP* resulting in a direct acyclic graph of sub-process dependencies can be performed in a hierarchical way from the independent sub-processes (sub-processes with no output variables) toward the root sub-processes (sub-processes with no controllable input variables). Then,

the computation can be distributed by following separate branches in the direct acyclic graph. Only a subset of the sub-processes can be optimized in a parallel way, and the distributive potentiality (the number of sub-processes solved in a parallel way) depends on the structure of the graph, the average connectivity of the sub-process nodes. A distributivity ratio of a Factored *MMDP* can be computed by comparing the total number of sub-processes $|T|$ and the size of the longest path in the graph $|maxPath|$:

$$distributivityRatio = \frac{|T| - |maxPath|}{|T|} \tag{11}$$

It is possible to relax the acyclic constraint by artificially cutting cycles in the direct graph of sub-processes before distributing the hierarchical optimization. The algorithm would require to visit several times each sub-process before converging. The message-based agent optimizations approach [11] addresses such configuration. However, *Partial* Distributivity relies on factorized Multi-agent *MDP* (i.e. sub-processes are defined on similar sub-transitions and sub-rewards).

3.3 Coordinated Distributivity

We propose to investigate a "forced" distributivity where join-policies could be built and optimized in a distributed way, in a constrained search space. The idea is to restrict the action capabilities in order to force a distributable model. Then the action restrictions can be negotiated by the agents during coordination phase. Intuitively, by "fixing" or "forbidding" some of the actions, it is possible to solve or erase some interaction dilemmas. For example, in collision-free multiple path planning, definitely fixing passages into a specific one-way direction allows each robot to plan its movements more independently. A robot does no more need to know the other robots trajectories to decide whether it can enter a narrow passage.

Considering a subset of joint-policies Π, it is possible to degrade a model M in a model DM^{Π} by considering only the transitions and rewards activated by the joint-policies in Π. DM^{Π} is built from M by reducing all the probabilities $P \in T$ and the reward criteria $C \in R$ definitions to handle only the actions that are included in, at least, one of the policies in Π.

With a Factorized *MMDP*, it consists in refining the inside structure of sub-processes in order to limit the dependencies. This process results in removing edges artificially in the direct graph modeling the sub-process dependencies in a way that the degraded model becomes completely distributable.

One subset of joint-policies Π potentially permits to distribute the computation. In most cases, applying a "coordinated distributivity" to a problem modeled as M consists in forbidding some actions (definitely or in specific configuration). The distributed computation converges to an optimal join-policy in the search space defined by Π.

Guaranteeing the optimality on the global *MMDP* requires to build several subsets of joint-policies in a way that their union covers all possible joint-policies. By assuming that such a decomposition of joint-policies provides the

global *MMDP*, optimally solving the global *MMDP* will consist in computing an optimal solution to each degraded model before selecting the best solution.

To guarantee a distributive computation all over the solving process, each of the subsets of joint-policies has to generate distributable degraded *MMDP*. Distributed solving of *MMDP*s permits to decrease the combinatorial explosion on variables and to parallelize the computation. The interest of the method to compute an optimal solution depends on the number of degraded models to test (i.e. if distributing the computation does not require to test an exponential number of degraded model).

4 Use-Cases and Discussions

The notion of distributivity with degraded model permits to navigate from completely distributable to strongly interactive *MMDPs*. A completely distributive *MMDP* models a system composed of independent sub-systems. A strongly interactive *MMDP* models a complex system where the components are interacting with a large number of the other components that results in strongly connected sub-processes.

By navigating among mobile robot problems addressed in the literature, it is possible to classify problems regarding the possibility to distribute the computation between: "completely distributable", "partially distributable" and "distributable with coordination".

4.1 Independent mobile robots

Classical multi-robot problems rely on planning a path for each of the robots of the fleet. In the simplest definition, each robot has one destination to reach and the movement outcome is under the assumption that a robot (α) movement is never impacted by other robots. The system state is composed of the robots' positions and the evolution of each robot position depends on the performed movement action. Potentially, the position evolution could be different at different time steps (with a discrete time variable t).

$$P_\alpha(Position'_\alpha \mid t,\ Position_\alpha,\ Move_\alpha) \tag{12}$$

Rewards model, for each robot α, a constant cost cst while the destination $dest_\alpha$ is not reached.

$$C_\alpha(Position_\alpha) = \begin{cases} 0 \text{ if } & Position_\alpha = dest_\alpha \\ -cst \text{ else} & \end{cases} \tag{13}$$

This model is very simple and matches poor scenarios. However, it is coherent, for instance, for individual vehicles navigating on the public road networks. The coordination can be achieved through a learning mechanism with a congestion probabilistic knowledge defined over t. *Aroor et Al.* [1] use this approach to allow a robot to navigate in a crowded-environment. There is no guarantee over the optimality of the emergent coordination in multi-robot context.

4.2 Swarm Robotic

Partial distributivity can be applied if the interaction between the robots is sparse and local. Each of the robots is in interaction with a limited number of other robots (ideally always the same other robots).

In fact, Swarm Robotics start from the assumption that complex global behavior can be defined by decisions taken in local interactions. The decisions are taken only regarding the neighboring rather than overall considerations. The platooning scenario is one of the emblematic and successful use cases of Swarm Robotics. In this scenario a fleet of robot moves across a cluttered environment toward a goal position by maintaining a formation at best.

There is no Factored *MMDP* approach applied to compute swarm behaviors, to my knowledge, in robotics. However, "Partial distributivity" presents a formal framework to optimize swarm behaviors. The decision can be handled locally by the agent itself (by reading only the variables attached to the action to perform) but it is also true for the computation of the decisions to take.

4.3 Multiple Path Planning

The coordination of several mobile robots sharing an environment where each of the robots can block other robots in their movement is known as collision-free multiple path planning problem. In its probabilistic formulation, the probability for a robot α to achieve its movement also depends on the position and the movement of all the other robots ($\beta,\ \delta,\ \ldots$) at the same time step t.

$$P_\alpha(Position'_\alpha \mid t,\ Position_\alpha,\ Move_\alpha,\ Position_\beta,\ Move_\beta, Position_\delta,\ Move_\delta, \ldots) \tag{14}$$

In parallel, the fleet of cooperative robots will share tasks to achieve. As a first approximation, each task is located somewhere in the environment and requires a unique robot to be done. Then, in parallel to movement evolution, the probability that the task will be performed depends on if one of the robots is in the appropriate position and doing the appropriate action.

$$P_X(DoneY' \mid DoneY,\ Position_\alpha,\ Act_\alpha,\ Position_\beta,\ Act_\beta,\ Position_\delta,\ Act_\delta, \ldots) \tag{15}$$

Multiple Path Planning and task allocation are difficult problems to solve in a distributed way while it puts global constraints on all the robot's actions. Considering that, the robot moves through a graph (e.g. modeling the possible paths in a warehouse), the collision-free multiple path planning problem can be modeled as an arc orientation problem coupled with a simple multiple path planning problem.

Arc orientation represents the way robots can move between two positions A and B (only A to B or B to A would be authorized at each time step t). By considering a fixed policy on graph configurations the subset of authorized joint-policies Π is composed of the joint robot movements accordingly. The resulting

degraded *MMDP* is distributable over robots computing resources. However, the problem is still hard to solve while there are $3^{|Arc|\times H}$ possible graph configurations with H the planning horizon (i.e. $t \in [0,\ H]$).

The model for planning with several task-positions to reach would include a boolean state variable for each of the tasks. The variable records if a task Y is performed or not ($DoneY = \{True, False\}$). The set of the robot actions (movements) is increased with actions allowing robots to perform tasks. This *MMDP* could be degraded by determining the robot authorized to perform a given task. This way Multi-task multi-robot mission can be decomposed in an allocation problem coupled to a path planning problem. Each test of a new allocation will require an update on the robot paths.

4.4 Limitation in "coordinated distributivity"

By using degraded *MMDP*, both "collision free" and "multi-task" features in multiple path planning can be seen as resources and task allocation coupled to path planning. In [6, 17], the task-targets match the frontiers between known and unknown areas in exploration missions. The robots plan their movements using *MDP* to the next target frontier to extend in a cooperative way. Due to the high dynamics of the robot knowledge, update and plan actualization are frequently performed based on heuristic coordination. Classically, allocations are handled with auction protocols where each agent bids over the allocation of tasks and resources [13].

This approach was extended in [15] to allow robots to coordinate themselves at the beginning of the mission and by considering several tasks to reach per robot. Task Allocation can be coupled with Collision-Free Multiple Path Planning. This problem, defined under a deterministic evolution, is effectively solved optimally [16] by taking advantage of decomposable structure.

In stochastic environment, the joint policy maintains a certain blur over the system evolution. Efficient multi-agent policies would require to plan reallocation of resources and tasks depending on the occurrence of uncontrolled events. The guarantee of finding eventually the optimal joint-policy using "coordinated distributed approach" depends on the theoretical capability of the allocation generator to explore all the possibilities in the time and event space (which would be intractable in practice).

5 Conclusion

This paper focuses on the notion of distributivity for planning under uncertainty and defines a classification of Multi-agent Markov Decision Process (*MMDP*). The classification is over "Completely Distributable", "Partially Distributable" and "Distributable with Coordination". This classification is formalized depending on the structure of the sub-processes inherent to *MMDP*. "Completely distributable" matches independent sub-processes and serves only as theoretical definition with a very poor interest in application. "Partially distributable" can

be applied when sub-processes dependencies connect each sub-process with only few other sub-processes. In that case, the distribution of the computation could follow the topology of the sub-process dependencies graph. In robotic applications some scenarios (multiple-path planning, task allocation) are hard to distribute while maintaining some guarantees over an optimal solving. Each of the robots are potentially in interaction with all the others. "Coordinated distributivity" set up a formal frame to split individual computations process in a distributable way. However, possibilities over the coordination configurations require a shared computation.

Some mobile robot problems remain strongly interactive and do not permit to simply distribute the model with a theoretical guarantee to converge to an optimal solution. By using "Coordinated distributivity" the number of coordination possibilities is exponential and prevents from investigating all of them. In future work, we aim to address strongly interactive multi-robot path planning under uncertainty. Forcing a distributed computation of the coordination configuration could be heuristically performed by including coordination variables inside the sub-processes individually solved.

References

1. Anoop Aroor, Susan L. Epstein, and Raj Korpan. Online learning for crowd-sensitive path planning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 1702–1710, 2018.
2. Raphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *J. Artif. Intell. Res. (JAIR)*, 22:423–455, 2004.
3. Aurélie Beynier and Abdel-Illah Mouaddib. Solving efficiently decentralized mdps with temporal and resource constraints. *Autonomous Agents and Multi-Agent Systems*, 23(3):486–539, 2011.
4. Craig Boutilier. Planning, Learning and Coordination in Multiagent Decision Processes. In Yoav Shoham, editor, *6th Conference on Theoretical Aspects of Rationality and Knowledge*, page 195–210, 1996.
5. Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49 – 107, 2000.
6. W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated Multi-Robot Exploration. *IEEE Transactions on Robotics*, 21:376–386, 2005.
7. Arnaud Canu and Abdel-Illah Mouaddib. Dynamic Local Interaction Model: framework and algorithms. In *International Conference on Autonomous Agents and Multiagent Systems, (Workshop MSDM)*, 2011.
8. Iadine Chades, Bruno Scherrer, and François Charpillet. A Heuristic Approach for Solving Decentralized-POMDP: Assessment on the Pursuit Problem. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, page 57–62, New York, NY, USA, 2002.
9. Guillaume Desquesnes, Guillaume Lozenguez, Arnaud Doniec, and Éric Duviella. Distributed mdp for water resources planning and management in inland waterways. *IFAC-PapersOnLine*, 50(1):6576 – 6581, 2017.

10. Jilles S. Dibangoye, Olivier Buffet, and François Charpillet. *Error-Bounded Approximations for Infinite-Horizon Discounted Decentralized POMDPs*, pages 338–353. Berlin, Heidelberg, 2014.
11. C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. *Advances in information processing systems*, 2:1523–1530, 2002.
12. Paul Jourdan, Guillaume Lozenguez, Luc Fabresse, and Noury Bouraqadi. Towards a distributed planning of decision making under uncertainty for a fleet of robots. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 800–807, New York, NY, USA, 2020. Association for Computing Machinery.
13. Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. *Multi-robot Task Allocation: A Review of the State-of-the-Art*, pages 31–51. 2015.
14. M. Littman, T. Dean, and L. Kaelbling. On the complexity of solving Markov decision problems. In *11th Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.
15. Guillaume Lozenguez, Lounis Adouane, Aurélie Beynier, Abdel-Illah Mouaddib, and Philippe Martinet. Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation. *Autonomous Robots*, 40(4):599–613, 2016.
16. Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1144–1152, 2016.
17. L. Matignon, J.P. Laurent, and A.I. Mouaddib. Distributed Value Functions for Multi-Robot Exploration. In *International Conference on Robotics and Automation*, 2012.
18. Abdel-Illah Mouaddib, Mathieu Boussard, and Maroua Bouzid. Towards a Formal Framework for Multi-Objective Multi-Agent Planning. In *Autonomous Agents and Multi-Agent Systems*, 2007.
19. R. Nair, M. Tambe, M. Yokoo, S. Marsella, and D.V. Pynadath. Taming decentralized pomdps. In *International Joint Conference on Artificial Intelligence*, 2003.
20. Frans A. Oliehoek, Shimon Whiteson, and Matthijs T.J. Spaan. Approximate solutions for factored dec-pomdps with many agents. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 563–570, 2013.
21. Joris Scharpff, Diederik M. Roijers, Frans A. Oliehoek, Matthijs T. J. Spaan, and Mathijs Michiel de Weerdt. Solving Transition-Independent Multi-Agent MDPs with Sparse Interactions. In *AAAI*, pages 3174–3180. AAAI Press, 2016.
22. P. Varakantham, J. Kwark, M. Taylor, J. Marecki, Scerri P., and M. Tambe. Exploiting coordination locales in distributed pomdps via social model shaping. In *International Conference on Automated Planning and Scheduling*, 2009.
23. Jingjin Yu and Steven M LaValle. Planning optimal paths for multiple robots on graphs. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3612–3617, 2013.