Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks*

Daniel Claes
University of Liverpool
dclaes@liverpool.ac.uk

Philipp Robbel
Massachusetts Institute of Technology
robbel@mit.edu

Frans A. Oliehoek
University of Amsterdam
University of Liverpool
fao@liverpool.ac.uk

Karl Tuyls
University of Liverpool
k.tuyls@liverpool.ac.uk

Daniel Hennes
European Space Agency, ESTEC
daniel.hennes@gmail.com

Wiebe van der Hoek
University of Liverpool
wiebe.van-der-hoek@liverpool.ac.uk

ABSTRACT

Although multi-robot systems have received substantial research attention in recent years, multi-robot coordination still remains a difficult task. Especially, when dealing with spatially distributed tasks and many robots, central control quickly becomes infeasible due to the exponential explosion in the number of joint actions and states. We propose a general algorithm that allows for distributed control, that overcomes the exponential growth in the number of joint actions by aggregating the effect of other agents in the system into a probabilistic model, called *subjective approximations*, and then choosing the best response. We show for a multi-robot grid world how the algorithm can be implemented in the well studied Multiagent Markov Decision Process framework, as a sub-class called spatial task allocation problems (SPATAPs). In this framework, we show how to tackle SPATAPs using online, distributed planning by combining subjective agent approximations with restriction of attention to current tasks in the world. An empirical evaluation shows that the combination of both strategies allows to scale to very large problems, while providing near-optimal solutions.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms; Experimentation

Keywords

Robotics; Robot Planning and Plan execution; Robot Teams; Multi-Robot Systems; Robot Coordination

1. INTRODUCTION

Although multi-robot systems are becoming more common in real world applications, in many cases, control still has a centralized component, e.g. approaches using centralized planning or a single auctioneer for task allocation [5, 34]. These robots have limited autonomy and rely on the central control component of the system for planning purposes. Such approaches have two major limitations, namely robustness and scalability. If a centralized planning algorithm fails, for example, the whole system is affected and becomes unavailable. Second, for each additional robot to be added, the joint action and state spaces increase exponentially, rendering it infeasible for any central algorithm to solve larger problems optimally in general.

Trying to overcome these issues by decentralized solutions introduces an important challenge in autonomous coordination of the multiple robots in the system. There are different approaches to tackle this problem, for instance swarm approaches inspired by nature [22, 2, 8], or others based on decentralized auctions [9, 3, 23], in which each robot bids for tasks and task assignment ultimately follows from the winning bid values. The swarm approaches are generally reactive approaches that lack the possibility to plan ahead (e.g., when tasks may appear stochastically in the environment) and the auctioning approaches rely heavily on reliable communication during the bidding phase. Furthermore, after the tasks are auctioned off, agent policies are fixed and remain unchanged until the next bidding phase.

In order to avoid these issues we develop a generic algorithm that applies to various instances of the coordination problem. We build on the idea from mean field game theory [17] that for many tasks it may be sufficient to reason about the aggregate effect of the other agents. This idea has been proven to be useful in particular instances of offline planning for large agent teams, such as congestion-like games and resource-coupled multiagent planning [16, 32, 1]. In this paper, we demonstrate that these ideas also bring great improvements in effectiveness in a very general class of online multi-robot planning problems. The key idea is to aggregate the effect of other robots in the area by using their locations as a proxy to predict their future actions and then choosing the best response. Using this principle, a robot does not need to reason about every other individual robot but only needs to know in general which tasks are likely serviced by another robot, in order to reduce the complexity of its own planning. Because planning is done online and at every time step, the algorithm is flexible in responding to changes in the environment. This also implies that communication between robots is strictly not required, however, the robots do need to be able to observe the current state (i.e., locations of the other robots), which can be enabled through (local) communication.

We define the problem as sub-class of Multiagent MDPs

*This paper also appears at AAMAS 2015.

Appears in: *The 10th Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (MSDM-2015)*, held in conjunction with *AAMAS*, May 2015, Istanbul, Turkey.

(MMDPs) [7] that we collectively refer to as *Spatial Task Allocation Problems (SPATAPs)*, and develop our algorithm as a number of online planning approximations that are tailored to exploit the characteristics of these problems. In particular, we investigate the general algorithm for settings with *negative interactions* in which each task can be serviced by a single agent. In such tasks, it makes sense to discount the reward for tasks that are likely attended to by other agents. Such approximate *empathic* reasoning was recently exploited in the context of multi-robot exploration [24] by making use of a modification of distributed value functions [27] (henceforth referred to as 'MDVF'). We introduce a simplification of this algorithm, *empathy by fixed weight discounting (E-FWD)*, that applies to all SPATAPs with negative interactions.

While these subjective approximations bring improvements in planning efficiency, they are not sufficient to make the planning problem tractable. Another crucial contribution of our approach is therefore the combination of the algorithm with another approximation method, called *Phase-Approximation*, that forms the basis of an approximate online planning technique without any exponential dependence on the number of agents or the number of state factors.

An empirical evaluation shows that these combined techniques yield near-optimal solutions in cases in which the optimum can be calculated and is highly scalable, while outperforming the state-of-the-art.

2. PROBLEM DESCRIPTION & APPROACH

We consider a problem class that we refer to as spatial task allocation problems (SPATAPs). In particular, SPATAPs describe settings in which a team of agents needs to service a set of tasks that are spatially distributed in an environment. Each task can be performed by one or more agents, and new tasks can appear in the world due to exogenous events outside of the agents' control.

As a running example we consider a cleaning task in for instance an office building. Dirt can appear at any location at any given time in this world, since a human can spill coffee, or leave paper on the floor, etc. A team of cleaning robots, in this case the agents, has the task to continuously keep the building clean. There is no central control of any sort, but the robots can observe the location of other robots, e.g. by using overhead cameras or communication. We will assume that the office building can be represented using a grid world, where each tile corresponds to a part of a room of a fixed size, but other representations (such as a more abstract graph) are also possible. The grid worlds can have arbitrary shape, see Figure 1 for examples that we consider in this paper.

The main problem is how the agents should plan their behavior in order to ensure that all tasks are serviced, i.e., in this case, that all dirt is continuously cleaned while minimizing interference between robots. Since there are nearly no restrictions on the sort of tasks, SPATAPs provide a very powerful and general model which are very hard to solve optimally. This is due to the fact that the joint action and state spaces increase exponentially with the number of agents in the system.

In fact, only a few methods can deal with such large problems. One approach that offers the required scalability is the 'partition organization' [29]. In this approach, the problem is partitioned in (overlapping) regions and each agent is assigned to one region. This approach is well suited for problems in which there is a straightforward partitioning, such as symmetric worlds. However this is not always possible, especially when the task appearance probabilities are not evenly distributed over the world or when specific tasks can only be served by particular agents.

Algorithm 1 Aggregated best response

Require: $\mathcal{D}_{-i}$: set of other Agents
Require: h : planning horizon (timesteps)
  for all agents j in $\mathcal{D}_{-i}$ **do**
    compute policy π_{SA} for j
    for $t = 1$ to h **do**
      compute probability distribution p_j^t following π_{SA}
    end for
  end for
  for $t = 1$ to h **do**
    //aggregate:
    $pm = \sum_{j \neq i} p_j^t$
  end for
  compute best response of agent i given pm

Another common approach, which we will treat in some more detail in Sec. 4.1.1 under the name *self-absorbed approximation*, is to ignore the presence of other agents during planning and to consider them as mere noise to each individual planning model. In the case of task allocation problems, however, such approximations lead to poor performance (as supported by our empirical evaluation), since the difficulty in this type of problems revolves around the coordination of which agent addresses which task.

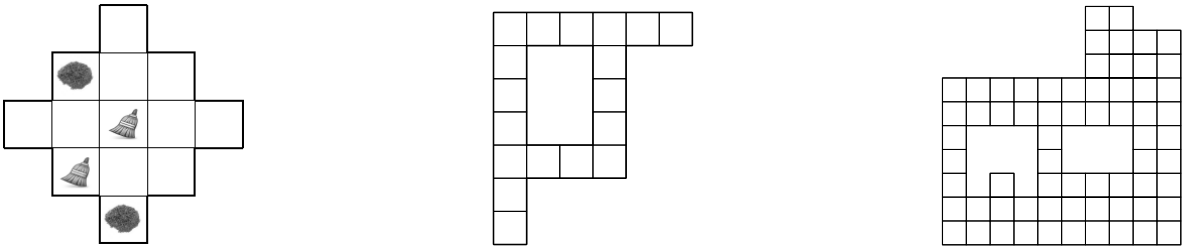
We propose to exploit the key characteristics of SPATAPs—independence of agent movement and the locality of tasks—for *efficient approximations* during online planning. The key idea is that an agent does not need to reason about all other agents individually, but can reason about their aggregated effect. In particular, if another agent is close to a task location, we can reason that this task will be serviced with a certain probability at some future time step. If we assume a reasonable policy for the other agents we can predict the probabilistic movements for all agents over the planning horizon and use the aggregated *presence mass (pm)* as a sufficient statistic for calculating our best response.

Algorithm 1 summarizes the idea described above. For all other agents, we fix a policy π and, for the time span of our planning horizon, predict their movements as following π. By doing this we obtain a probability distribution of the agents' location for all time steps. These probability distributions can be aggregated and used by each individual agent to calculate a best response.

In the next section, we show that SPATAPs can be directly modeled as MMDPs, and explain why this is not of immediate help since the complexity of solving the problems with standard MMDP methods remains prohibitively high. Section 4 presents a remedy by introducing online approximate methods that exploit the key characteristics of SPATAPs.

3. SPATAPS FORMULATED AS MMDPS

The problems we consider in this paper describe a set of spatially distributed tasks that a team of agents or robots needs to solve. A key characteristic of such problems is that the outcome of an action is uncertain (cleaning the dirt or



(a) diamond world (b) corridor world (c) office world

Figure 1: Three possible SPATAP environments.

moving in the world may each fail with some probability, e.g., due to wheel slip), and new tasks can appear due to unforeseen exogenous events (e.g., a human spilling some dirt). Therefore, SPATAPs can be seen as a special case of MMDPs, which we will introduce next.

3.1 Multiagent MDPs

Markov decision processes (MDPs) provide a general framework for sequential decision making under uncertainty [26]. The Multiagent MDP (MMDP) is the straight-forward extension to the case of multiple decision makers who observe the full state of the environment:

Definition 1. A *Multiagent Markov decision process (MMDP)* is defined as a tuple $\langle \mathcal{D},\mathcal{S},\mathcal{A},P,R \rangle$, where $\mathcal{D} = \{1,\ldots,n\}$ is the set of n agents, $\mathcal{S}$ a finite set of states s of the environment, $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ the set of joint actions $a = \langle a_1,\ldots,a_n \rangle$, T the transition probability function specifying $P(s'|s,a)$, and $R(s,a)$ the immediate reward function.

An MMDP is called *factored* if its state space is spanned by a set of state variables. Note that an MMDP is significantly different from a Dec-MDP [6], since agents in an MMDP, per definition, can observe the (global) state. A (joint) policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ in an MMDP maps states s to joint actions a, and is equivalent to a tuple of individual policies $\pi_i : \mathcal{S} \rightarrow \mathcal{A}_i$. The Q-value of (s,a) under policy π is defined as the expected sum of rewards when executing a in s and following π afterwards. In this paper we will consider (undiscounted) h-stage look-ahead planning, i.e., constructing a plan that specifies actions from 'now', $t = 0$, to stage $t = h-1$. For this setting, the value function for each stage t equals $V^t(s) = \max_a Q^t(s,a)$, where

$$Q^t(s,a) = R(s,a) + \sum_{s'} \Pr(s'|s,a) V^{t+1}(s'). \qquad (1)$$

The optimal policy π^* and corresponding optimal value functions Q^t, maximize the expected reward for every (s,a). Solving MMDPs can be done in a similar fashion as (single-agent) MDPs [26]. However, since the number of joint actions is exponential in the number of agents and the number of states is exponential in the number of factors (itself usually dependent on the number of agents), this is intractable for even small problems in practice.

3.2 SPATAPs

SPATAPs can be defined as a sub-class of MMDPs with some additional structure. Underlying a SPATAP is a map that specifies the potential task locations $\mathcal{L}$ and that defines $\mathcal{A}_M$, the set of movement actions. E.g., for the "dirt cleaning" example, all agents are homogeneous and share a common (movement) action space $\mathcal{A}_M = \{N,\ E,\ S,\ W,\ STAY\}$. There further exists a *task structure*, defined by a set of task types $\mathcal{T} = \{\mathcal{T}_0,\mathcal{T}_1,\ldots,\mathcal{T}_{|\mathcal{T}|}\}$. Each type $\mathcal{T}_k$ has an associated set of task states $\mathfrak{T}_k$ that indicate the status of the task. In our running example, $\mathcal{T}_1$ could have states $\mathfrak{T}_1 = \{\text{very dirty, dirty, nearly clean}\}$. $\mathcal{T}_0$ refers to a special type indicating there is no task and only has one state $\mathfrak{T}_0 = \{CLEAR\}$. We use $\mathfrak{T} = \bigcup_k \mathfrak{T}_k$ to denote the set of all task states. Each task type $\mathcal{T}_k$ may optionally be associated with one (or more) particular action $a_{\mathcal{T}_k}$ to perform that task. Each agent i can perform movement and task actions. The agent team can be homogeneous, i.e., have the same capabilities, but the framework also allows agents that have different capabilities, or differences in how effective an agent is at a particular task.[1]

These SPATAP-specific components can now be used to define the induced MMDP. Agents and their actions are unchanged. A state is a tuple $s = \langle \lambda,\tau \rangle$, where λ is the vector of locations (λ_i denotes the location of agent i), and τ is the task status vector (τ_x denotes the task status at location x). The transition function can be factored as

$$P(\lambda',\tau'|\lambda,\tau,a) = \prod_{x \in \mathcal{L}} p_x^T(\tau_x'|\tau_x,\lambda,a) \prod_{i \in \mathcal{D}} p_i^M(\lambda_i'|\lambda_i,a_i) \qquad (2)$$

where p^T are task transition probabilities and p^M are agent movement probabilities.[2] The task transition probabilities p^T are assumed to be conditionally independent given the locations and actions of the agents and encode the probability of progressing toward finishing the tasks, as well as exogenous events that spawn new tasks.

The reward function is additively factored and is the sum of task rewards R^T and movement costs R^M:

$$R(s,a) = \sum_{x \in \mathcal{L}} R_x^T(\tau_x,\lambda,a) + \sum_{i \in \mathcal{D}} R_i^M(\lambda_i,a_i). \qquad (3)$$

As the name 'movement cost' implies, the functions R_i^M will typically specify negative rewards. For the task reward functions, it is also natural to specify a cost for every stage that the task is not completed (consider the robot rescue setting where victims require attention as quickly as possible). More options are possible, e.g., a business may want to use as the reward function the actual amount of money a particular assignment will generate. Note that the model also supports rewards that depend on the next state by taking

[1]The approximation method we introduce in Sec. 4.1, does assume homogeneous agents, but the approach can be extended to heterogeneous teams by applying the same technique to each 'type' of agent.

[2]Movements are independent, effectively assuming that lower-level path-planning will avoid collisions within the same location.

the expectation over next states. E.g.,

$$R_x^T(\tau_x,\lambda,a) = \sum_{\tau'_x} \Pr(\tau'_x|\tau_x,\lambda,a) R_x^T(\tau_x,\lambda,a,\tau'_x). \quad (4)$$

3.3 Locality Assumptions

While the above lays out the general form of SPATAPs, such problems are, in general still very difficult since the terms p_x^T and R_x^T are *non-local*, i.e., they depend on all the agents, including those that are far away from location x. In order to gain more traction on the problem, we will assume that a task at a particular location x will only be influenced by a subset of agents. This subset we call the *locality scope* $\mathbb{L}(x,\tau_x,\lambda,a)$ and it depends on the location x, the task type and state encoded by τ_x, and λ,a. For instance, in our example $\mathbb{L}(x,\tau_x,\lambda,a)$ for a dirty location x will only contain those agents at x that perform the 'clean' action. In the remainder of this paper, we will simply write $a_{\mathbb{L}}$ for the action profile of agents in the locality scope. As such, we will consider task transitions of the form $p_x^T(\tau'_x|\tau_x,\lambda_{\mathbb{L}},a_{\mathbb{L}})$. Similarly, we will assume that the task rewards can be expressed as $R_x^T(\tau_x,\lambda_{\mathbb{L}},a_{\mathbb{L}})$.

Each SPATAPs is an MMDP, so MMDP solution methods apply. Unfortunately, both the number of states and joint actions are very large for these problems (see Table 1). As a result, even state-of-the-art MDP solvers that exploit factored structure [19] have problems with the smallest SPATAPs. One would hope that the special structure of SPATAPs may make them easier to solve but, unfortunately, this is not the case in general:

THEOREM 1. *Optimally solving SPATAPs is as hard as solving MMDPs; i.e., SPATAPs are 'MMDP-hard'.*

PROOF. We reduce from the problem of solving an MMDP by creating a SPATAP with a single location and a single task. The task states correspond to the states of the MMDP and similarly can we derive p_x^T and R_x^T from the transitions and reward of the MMDP. The optimal solution of this SPATAP is the optimal solution of the MMDP. □

This theorem illustrates that while the concept of tasks is very general and powerful, this comes at a worst-case computational cost. Nevertheless, SPATAPs offer ample opportunities to exploit their specific characteristics. In particular, in the following sections, we propose two types of approximation techniques that each directly exploit problem structure.

4. ONLINE APPROXIMATIONS

We introduce two orthogonal approximation approaches for online planning for SPATAPs, that increase efficiency by exploiting two characteristics of real-world SPATAPs.

First, SPATAPs exhibit independence of movement and locality of task transitions and rewards, which implies that agents in many cases can act relatively independently of each other. To exploit this property, we consider subjective approximations that reduce the complexity introduced by the large number of agents. In particular, we propose a novel 'empathic' subjective approximation method that combines the computational benefits of subjective approximations with an approximate reasoning over the team members as introduced in Algorithm 1.

Second, for SPATAPs that exhibit only infrequent appearance of new tasks, planning about the currently active tasks can be expected to yield good performance. We propose to exploit this using a novel 'phase approximation' technique, which reduces the complexity introduced by the large number of states.

Both techniques transform a larger input MMDPs into approximate, smaller output (M)MDP models that leverage the unique properties of SPATAPs. The resulting models can be solved optimally using standard online planning methods. Moreover, both techniques are highly complementary and their combination yields approximate solution methods that are robust, scalable, and easy to implement: each agent can simply use its individual, online (single-agent) MDP planning method applied to a *subjective phase MDP*, which leverages the techniques described above.

In this paper we assume that the agents use the resulting MDPs to perform online planning over a fixed lookahead horizon (h), but more sophisticated (e.g., adaptive horizon [14]) methods can be applied too.

4.1 Subjective Approximations

The first set of techniques by which we bring computational leverage to the (online) planning process is the implementation of the approach described in Section 2, which aims to address the complexity due to the presence of multiple agents. They increase planning efficiency by distributed approximation: decomposing the larger problem into a set of approximate smaller planning problems, one for each agent.

4.1.1 Self-absorbed Agent Approximation

The extreme case of subjective approximation is to plan for each agent independently, assuming that it is the only agent present in the problem. This is a common approach [15], and we refer to this type of approach as the *'self-absorbed agent' approximation*. A self-absorbed agent i only models its own location and thus has individual states $s_i = \langle\lambda_i,\tau\rangle$. It also assumes that the transitions only depend on its own actions

$$p_i^{SA}(s'_i|s_i,a_i) = \left[\prod_{x\in\mathcal{L}} p_x^{T,SA}(\tau'_x|\tau_x,\lambda_i,a_i)\right] p_i^M(\lambda'_i|\lambda_i,a_i)$$

$$R_i^{SA}(s_i,a_i) = \left[\sum_{x\in\mathcal{L}} R_x^{T,SA}(\tau_x,\lambda_i,a_i)\right] + R_i^M(\lambda_i,a_i). \quad (5)$$

It may be difficult to map $p_x^T(\tau'_x|\tau_x,\lambda_{\mathbb{L}},a_{\mathbb{L}})$, $R_x^T(\tau_x,\lambda_{\mathbb{L}},a_{\mathbb{L}})$ to $p_x^{T,SA}(\tau'_x|\tau_x,\lambda_i,a_i)$ and $R_x^{T,SA}(\tau_x,\lambda_i,a_i)$ respectively. However, in many cases, it is possible to assume a default effect or default action for the other agent (e.g., we can assume that there will be no other agent cleaning the same spot). Another approach is to treat the agents as noise [15], by imposing some (e.g., uniform) distribution over λ_{-i},a_{-i}.

Formally, we define a *subjective MDP (S-MDP)* for agent i as a tuple $\langle\mathcal{S}_s,\mathcal{A}_i,p_i^{SA},R_i^{SA}\rangle$, where $\mathcal{S}_s$ is the subjective state space of states $s_i = \langle\lambda_i,\tau\rangle$. Solving a S-MDP can be done with standard techniques, yielding value functions $V_i^{SA,t}(s_i)$ and $Q_i^{SA,t}(s_i,a_i)$, which directly follow from (1).

The S-MDP improves significantly over the MMDP formulation in terms of complexity (see Table 1). As shown, there is no longer any exponential dependence on the number of agents in an S-MDP, which directly means that it admits more efficient solutions. However, we expect that self-

Table 1: Sizes of the state and actions spaces of the considered models. $|\mathcal{D}|$ is the number of agents, $\mathcal{A}_*$ denotes the largest individual action set. Planning times are a polynomial function of these quantities.

	state space	action space
MMDP	$\lvert\mathcal{L}\rvert^{\lvert\mathcal{D}\rvert} \cdot \lvert\mathfrak{T}\rvert^{\lvert\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert^{\lvert\mathcal{D}\rvert}$
S-MDP	$\lvert\mathcal{L}\rvert \cdot \lvert\mathfrak{T}\rvert^{\lvert\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert$
Phase-MMDP	$\lvert\mathcal{L}\rvert^{\lvert\mathcal{D}\rvert} \cdot \lvert\mathfrak{T}\rvert^{\lvert p\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert^{\lvert\mathcal{D}\rvert}$
SP-MDP	$\lvert\mathcal{L}\rvert \cdot \lvert\mathfrak{T}\rvert^{\lvert p\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert$
k-SP-MDP	$\lvert\mathcal{L}\rvert \cdot \lvert\mathfrak{T}\rvert^{k}$	$\lvert\mathcal{A}_*\rvert$

absorbed agent approximations are insufficient in domains where agents need to perform a fair amount of coordination. Next, we propose a number of approaches that do account for interactions between agents.

4.1.2 Empathy by Predicting other Agents' Locations

As summarized in Algorithm 1, the key idea is the following: in order to compute a best-response from the perspective of one agent, it only needs to predict what tasks will be tackled by the other agents. That is, it only cares about the aggregate effect of the actions of the rest of the team, but not about which team member addresses which task in particular. In order to predict what tasks will be addressed by the rest of the team, we use the predicted probability for agents being at a location as a proxy for them addressing the task at that location. We refer to this as *presence mass (pm)*.

In particular, from the perspective of an agent i, we want to be able to predict the location λ_j^t of all other agents $j \neq i$, t-stages from now. That is, we want to compute the probability distribution $\Pr(\lambda_j^t|s^0)$ where s^0 is the full MMDP state 'now' (i.e., at the time when the agent performs this prediction).

To compute these 'presence mass' distributions, one needs to assume particular behavior of the other agents. One possibility is to assume that other agents perform a random walk [24]. This assumption, however, leads to uninformative uniform distributions over states when predicting further into the future. To avoid this problem, we assume that the other agents use a self-absorbed model with quantile response (i.e., we assume that the other agents take actions according to a Boltzmann distribution [31] specified using the self-absorbed agent approximation V_{SA}). Fig. 2 (middle) illustrates that this leads to more sensible predictions. When there are multiple agents present, we can accumulate the presence mass distributions into a single 'sufficient statistic' and therefore do not need to account for every single agent during planning. We use the accumulated presence mass distribution for the next model.

4.1.3 Empathy by Fixed Weight Discounting

After calculating the aggregated presence mass of the other agents, the planning agent has to choose its response. Distributed value functions (DVFs) [27] allow agents to share their value function. This, however, leads to a lot of communication overhead. The MDVF [24] approach is inspired by DVFs and implements agent collaboration by using a second value function (specifically, V_{SA}) to discount the values of future states. In the resulting formulation, however, MDVF agents do *not* share their value functions. Instead, each agent computes V_{SA} in parallel and uses it to discount the V^{MDVF} values.

Realizing this, we propose a more straightforward approach: we do not discount using V_{SA}, but just use the next-stage value function. We refer to this simplification as *empathy by fixed-weight discounting (E-FWD)*. The resulting value function is given by

$$Q_i^{FWD,t}(s_i,a_i) = R^{SA}(s_i,a_i) + \sum_{s'} p^{SA}(s_i'|s_i,a_i) \left[\left(1 - f_i \sum_{j\neq i} \Pr(s_j^{t+1} = s_i'|s^0)\right) V_i^{FWD,t+1}(s_i')\right]. \quad (6)$$

where R^{SA} and p^{SA} are the self-absorbed model components (these are the same for all agents and hence we drop the subscript i to simplify notation). The last probability term is the presence mass of an agent j being at the location specified by s_i' $t+1$ stages into the future, i.e., $\Pr(s_j^{t+1} = s_i'|s^0) = \Pr(\lambda_j^{t+1}|s^0)$ with λ_j^{t+1} the location specified by s_i'. Finally, f_i is a fixed weight that determines how much the value of a next state is discounted. We follow [24] and set f_i to $\max R(s,a)/\max V(s,a)$. An example illustrating this discounting is shown in Fig. 2 (right).

4.2 Phase-Approximations

While subjective approximations reduce the complexity due to multiple agents, they do not sufficiently reduce the complexity of the state space. To overcome the complexity of the state space, we propose a different way of approaching the problem. Rather than seeing each location $x \in \mathcal{L}$ as a potential location for a task that may appear or disappear over the planning horizon, we focus only on the current 'task phase', i.e., the *currently active* set of tasks indicated by those locations x for which $\tau_x \neq CLEAR$. By focusing only on these locations, the number of task states induced is much smaller, allowing for big increases in planning efficiency.

4.2.1 Phase MMDPs

We formalize this idea by means of the so-called *phase-MMDP*, which, given a global state $s = \langle\lambda,\tau\rangle$, can be defined as follows. A *Phase-MMDP for state s*, is an MMDP $\langle\mathcal{D},\mathcal{S}_p,\mathcal{A},P^p,R^p\rangle$. The considered task locations in this MMDP, however, are restricted to the set of $p\mathcal{L} = \{x \in \mathcal{L} \mid \tau_x \neq CLEAR\}$ of *phase task locations*, i.e., the set of 'active' locations where there is a task. Thus, the state space $\mathcal{S}_p$ is spanned by the set $\mathcal{L}^{|\mathcal{D}|}$ of joint locations and the set $\mathfrak{T}^{|p\mathcal{L}|}$ of all possible task vectors for the active locations. We write $p\tau \in \mathfrak{T}^{|p\mathcal{L}|}$ for the restriction of τ to the locations in $p\mathcal{L}$. A phase-MMDP state is a tuple $ps = \langle\lambda,p\tau\rangle$. The transition (and reward) function follow from equation 2 (and 3) by restricting the product (summation) to $p\mathcal{L}$.

A phase-MMDP provides leverage by restricting the number of states compared to the regular MMDP formulation. However, this is highly dependable on the number of active tasks, and, in the worst case, it does not reduce the state space at all. Also, it does not address the large joint action space (see Table 1). This motivates the combination of the previously introduced techniques.

4.2.2 Subjective Phase MDPs

Realizing that subjective and phase-approximations yield complementary gains, we propose to combine both approximations in a formalism that we refer to as *subjective phase*

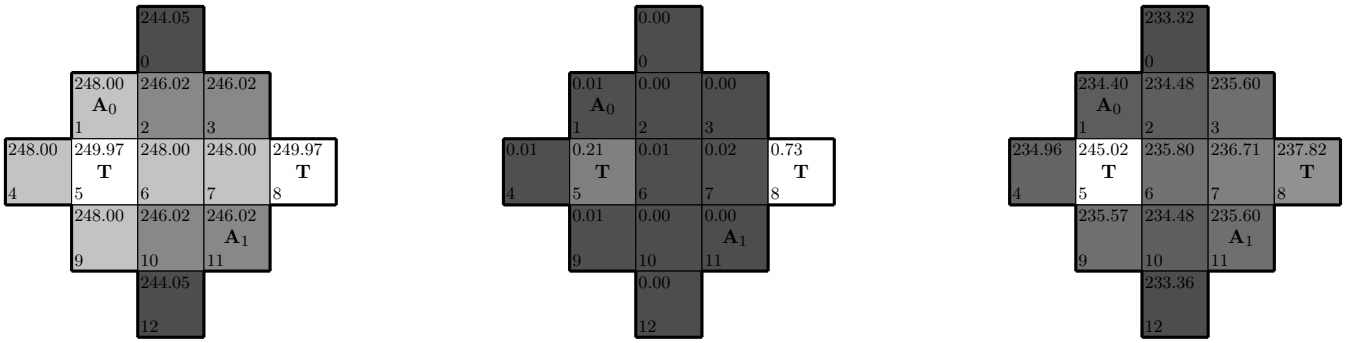


Figure 2: Left: A sample state for a diamond shaped gridworld with two agents (A_0, A_1) and two active task locations and V_{SA} for the current state. Middle: The presence mass of A_1 from the viewpoint of A_0. Right: The discounted value function V_{EFWD} for A_0 resulting for this configuration.

MDP (SP-MDP). An SP-MDP for agent i is a subjective model, meaning that it includes only the actions of agent i itself, moreover, it is a phase approximation, meaning that the states only include task states for active tasks. Specifically, a local state is a tuple $s_i = \langle \lambda_i, p\tau \rangle$, where λ_i is the location of agent i and $p\tau$ is the phase task vector. In an SP-MDP, the number of actions is the number of individual actions and the number of states is potentially much smaller due to the phase-approximations assumption (see Table 1).

4.2.3 *k*SP-MDPs

As mentioned, in the worst case there are many active tasks, which means that the number of states will still be prohibitive. However, by the combination of subjective and phase-approximations, it is possible to exploit the problem structure even further. In particular, *the subjective model of each agent may make different approximations* by exploiting what parts of the current state are relevant to that agent.

For instance, in the construction of the SP-MDP for an agent i, we can now make use of the location of that agent, by restricting the state space of the SP-MDP to include only task locations for the k nearest tasks. We refer to the resulting model as kSP-MDP. The number of states of the kSP-MDP is given by $|\mathcal{S}_{ksp}| = |\mathcal{L}| \cdot |\mathfrak{T}|^k$.

As is clear from Table 1, the kSP-MDP is *the only model* that is guaranteed not to have any exponential complexities. Standard dynamic programming for a h-step lookahead MDP takes time $O(h|\mathcal{S}|^2|\mathcal{A}|)$, and thus is feasible for large problems when using the kSP-MDP model.

4.3 Social Laws

One of the drawbacks of subjective approximations is that when deploying a team of identical agents, this can lead to agents behaving identically when this is not desired. For instance, when two agents are in the same location, they will make the same assumptions about the other agents' behavior and thus compute the same value function. Consequently they will take the same action and (with probability depending on the movement model) end up in the same next location.

Although this phenomenon exposes a principal flaw of subjective approximations, in SPATAPs these issues are easy to deal with by employing social laws [28]. For instance, in our experimental evaluation we employ the following social law: whenever more than one agent is in the same location, these agents select their actions based on their IDs: the agent with the lowest ID selects the action with the highest expected reward, the next agent selects the second best action, etc.

5. EXPERIMENTS

In this paper we have introduced a number of approximations that culminated in a model without any exponential dependence on the number of agents or state factors. This model can therefore be efficiently solved online using standard MDP techniques. Since the approximations that we introduced are not bounded, we report the results of an empirical evaluation aimed at determining the solution quality afforded by these approximations. For this purpose, we implemented a dirt-world simulation in which agents plan online, in a distributed fashion, using the kSP-MDP model. The movement transition probabilities p_i^M are such that a movement can fail (the agent remains at its previous location) with 10% probability. The task at location x is deterministically completed if any agent i performs action $STAY$ at that location. A task appears at a location x with probability 0.05 (but an agent staying at a location prevents task appearance). The team of agents receive reward +1 for every clean location at every time step. We do not consider movement costs. Agents solve their individual kSP-MDPs for (a maximum of) $h = 20$ steps lookahead, using regular dynamic programming.[3] Unless reported differently, we use the $k = 4$ nearest tasks.

In order to assess overall solution quality, we compare the approach with the global MMDP solution. Note that the global MMDP, unlike the phase-MMDP approximation, considers all locations on the board potential task locations, even currently 'inactive' ones. We use SPUDD [19], the state-of-the-art optimal solver for factored MDPs, to provide the value of the optimal solution for horizon 10, and compare this to the average value generated by 100 dirt-world simulations with online planning[4]. Table 2a shows the results for this comparison. SPUDD was only able to scale to 2x2 and 3x3 gridworlds with two and three agents respectively. For these problems, the approximations perform very well; even the naive self-absorbed approximation achieves over 93 % of optimal. The proposed simplification

[3]Increasing the lookahead beyond 20, did not increase the performance in our experiments. Shortening it does hurt performance in the larger problems, since it may lead agents to conclude that a task can never be addressed.

[4]Even though SPUDD is not specialized for MMDPs, it currently still is the best optimal solver for such problems.

world	SA	MDVF	EFWD	SPUDD
2x2	93.32%	97.86%	98.41%	100%
3x3	94.73%	96.83%	97.24%	100%

(a)

world	$\|\mathcal{L}\|$	n	$\|\mathcal{S}\|$	$\|\mathcal{A}\|$
Line	12	2	$5.90e+05$	25
Diamond	13	3	$1.80e+07$	125
Corridors	18	3	$1.53e+09$	125
4x4	16	4	$4.29e+09$	625
6x6	36	5	$4.16e+18$	3125
Office	66	6	$6.10e+30$	15625

(b)

Table 2: (a) Relative values of the three approaches averaged across a set of randomly drawn starting states and compared to the SPUDD optimum value function. (b) Larger dirt-world benchmarks.

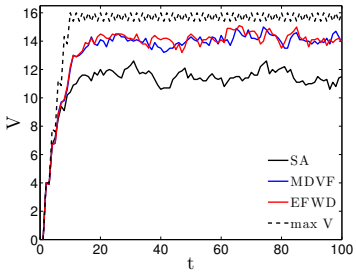


Figure 3: Mean reward over time for a 4x4 gridworld (initially full) with three agents.

	U.B.	EFWD	% of U.B.
Line	1104.8	875.5	79.2%
Diamond	1227.9	1052.6	85.7%
Corridors	1657.2	1379.3	79.6%
4x4	1518.0	1332.1	87.8%
6x6	3242.9	2490.2	76.8%
Office	5137.7	3618.6	70.4%

Table 3: Performance of EFWD as compared to an upper bound for problems that cannot be solved optimally.

E-FWD even yields slightly higher rewards than the more complex MDVF.

In order to provide insight in the differences in behavior between the approaches, we investigate the rewards received over the different stages per episode. Figure 3 shows the mean reward per time step (mean is taken over 10 episodes) in a 4x4 gridworld filled with tasks. As the figure shows, the reward increases at first and then converges close to the maximum reward achievable per time step (which is 16 and received when all squares are clean). Interesting to note is that V_{SA} outperforms the empathic approximations in the first time steps but then converges to a lower average reward. This is probably again due to the lack of considering the effects of other agents when using V_{SA}. The agents first select the tasks greedily, thus close tasks, and the tasks further away are not covered. The empathic approaches avoid this pitfall: by discounting the values of locations quickly reachable by many other agents, the locations only reachable by a protagonist are incentivized.

To examine the impact of restricting planning to only the k nearest tasks, we performed an experiment in which we vary k, holding other parameters fixed. For this experiment, we used a "full" 4x4 gridworld, i.e. dirt is present everywhere, with three agents that used the E-FWD algorithm to select their actions. Results shown in Figure 4a, are averaged over 100 runs of horizon 20 with 95% confidence intervals. Additionally shown are the number of states for each k (the dashed line). The figure clearly shows that, although $k = 1{,}3$ perform poorly, there is no significant difference for $k \geq 4$, which explains our choice of $k = 4$ for all the other experiments. Finally, we test the performance of our approximations on a number of larger test problems, listed in Table 2b. The "Line" world is a straight line of 12 states, and the "Corridors" and "Office" world are shown in Figure 1. These problems are too large to be solved optimally, e.g., our largest test domain "Office" has $6.10e+30$ states and 15625 actions, which makes it well beyond anything that can be solved optimally.

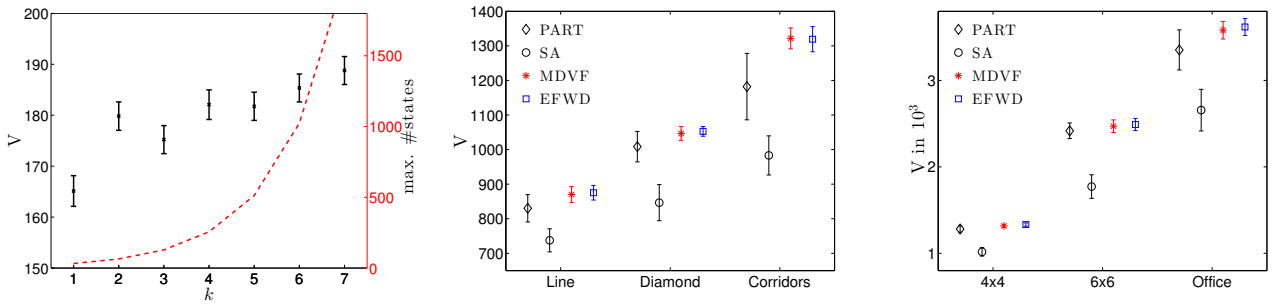
We compare our approach against the "partition" approach as described in Section 2. We automatically calculate the partitions by assigning each location to the closest agent. If there are multiple agents with the same distances, the location is added to both partitions. We refer to this approach as "PART". PART still suffers from the fact that large regions lead to too large local problems which we addressed by also restricting to the k nearest tasks in these problems.

Each method is run for 100 steps and repeated 10 times with random initial positions for the agents, while the world always being "full". Figure 4b shows the mean total reward including the 95% confidence intervals, i.e. non-overlapping error-bars mean statistically significant results. The self-absorbed approach performs the worst for every setting. The simplifications of E-FWD do not lead to a loss: there is no significant difference with MDVF and both have higher means and smaller variance than PART. Especially in more complex worlds, i.e. the "Corridors" and the "Office", the PART approach has a very high variance due to the different partitioning for each run. E-FWD is more reliable because it does not depend on the initial partitioning.

Additionally, we computed a theoretical (loose) upper bound by assuming that at every stage the expected number of tasks appears (fractions of tasks are allowed) and that all agents are able to clean an assigned task within two time steps. In other words, the upper bound assumes that in one step each agent uses a 'teleport' move to reach the location of a next task, which is then serviced in the second step. Clearly, this upper bound is an overestimation of the optimal value, since at each time step new tasks appear at random locations and agents generally need more than one-step travel times to these tasks. However, as demonstrated in Table 3 the proposed approximations yield rewards relatively close to this upper bound. For instance, about 75% in the 6x6 world and about 70% in the "Office" world are achieved, indicating that our proposed approach demonstrates reasonable behavior even for these huge problems.

6. RELATED WORK

In this work, we define approximate models which we can solve optimally. This should be contrasted with efforts to ap-



(a) varying k

(b) Different problem sizes

Figure 4: (a) Mean reward and number states for k-nearest task phase MDP with a 4x4 gridworld and three agents while increasing k. (b) Mean reward for various gridworlds as presented in Table 2b.

proximately solve exact models (e.g., [20]). The restrictions of the local problem of each agent to a subset of state factors is reminiscent of converting to a Dec-MDP, but in fact fundamentally different, since *the observation of the global state s is used to construct the agents' kSP-MDPs*. Moreover, despite recent advances, e.g., [12], Dec-MDP solution methods do not nearly scale to problems of the size considered here, or are suitable only for transition and observation independent settings [4, 11] (which our setting is not). While there have been other approximate methods for solving MMDPs, these typically depend on pre-specifying the fixed, or context-dependent coordination structure [18, 21, 30]. For SPATAPs, however, fixed coordination structures are a poor choice and the number of contexts to be considered is huge. In addition, these methods are not aimed at exploiting the particular structure present in SPATAPs, independence of movement and locality of tasks. To overcome the problem of pre-specifying interaction structures one can try to learn them [25, 10], but the premise underlying these methods is that there are only few states in which the agents need to coordinate. In contrast, in SPATAPs, the agents need to coordinate their task selection in *all* states.

Swarm approaches based on ant and honeybee colony behavior rely on local interactions in the environment to achieve coordination in multi-robot systems. From these local interactions global intelligence emerges at the group level, i.e. self-organization, capable of achieving efficient coordination in foraging and coverage tasks. Although these swarm solutions are efficient and effective for foraging and coverage problems, they do not consider the dynamic appearance of new or different (sub)tasks that can arrive at unpredictable moments in the environment [22, 2, 8]. Furthermore, [8] considers sequential interdependent tasks in which subtasks must be completed one after the other in order to complete an overall task. This work however is limited to the foraging problem and two subtasks.

Approaches for assigning agents to tasks based on auctions [9, 3] are closely related, but either do not reason about subsets of tasks [9], or do not properly address the sequential nature of the task in SPATAPs [3, 23]. SPATAPs also relate to more general resource allocation problems [33] (agents can be interpreted as resources). We, however, allow reallocation at every time step and consider spatially distributed tasks and travel times.

The idea of interacting with the aggregate effect of other agents is studied in detail in the field of mean-field games [17], where the focus lies on characterizing equilibria. A few approaches have tried to extend these ideas to engineering settings such as taxi-fleet optimization [32, 1] and theme park crowd management [16] via off-line planning. However, since these approaches perform off-line planning, these approaches are restricted in the richness of the state space that can be used as the basis for action selection. The 'aggregate effect' in these approaches typically consists of the number of agents present in different zones which directly affects utility for the protagonist. In our case, we use the predicted future agent locations as a proxy for their behaviors, which in turn will affect the utility of the protagonist.

Finally, the subjective approximations presented in this paper can be interpreted as online planning for a special instance of a level 1 interactive POMDP [15, 13]. In contrast to standard interactive POMDP solution methods, however, we propose dedicated approximation algorithms that exploit the characteristics of SPATAPs by using location as the proxy for the other agents' policies.

7. CONCLUSIONS & FUTURE WORK

This paper introduces SPATAPs, a general sub-class of MMDPs suitable for spatially distributed problems that a team of agents or robots needs to solve. Such tasks are characteristic of many realistic multi-robot systems, such as mobile sensor nets, distributed transportation and task assignment, multi-robot exploration, etc. To combat the complexity of general MMDP algorithms, we propose to use phase- and subjective approximations, and combine both to yield an efficient online planning method for SPATAPs. An extensive empirical evaluation shows that the proposed combination of approximation techniques yields near-optimal results for problem instantiations that we could solve optimally and further scales to much larger problems with thousands of states and joint actions. In future work we will focus on quality guarantees for the proposed approach, as well as demonstrating the approach on real teams of robots. Other promising directions of future work include application of approximate online MDP planning methods, and identifying methods for 'positive interaction' settings, in which there are joint tasks for which two or more agents are required.

Acknowledgments

F.O. is supported by NWO Innovational Research Incentives Scheme Veni #639.021.336.

REFERENCES

[1] Asrar Ahmed, Pradeep Varakantham, and Shih-Fen Cheng. Uncertain congestion games with assorted human agent populations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 44–53, 2012.

[2] S. Alers, K. Tuyls, B. Ranjbar-Sahraei, D. Claes, and G. Weiss. Insect-inspired robot coordination: foraging and coverage. In *Artificial life 14*, pages 761–768, 2014.

[3] Sofia Amador, Steven Okamoto, and Roie Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1384–1390, 2014.

[4] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Transition-independent decentralized Markov decision processes. In *AAMAS*, pages 41–48, 2003.

[5] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic Roommates Making Pancakes. In *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[6] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[7] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proc. of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210, 1996.

[8] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 28(1):101–125, 2014.

[9] Jesús Capitán, Matthijs T. J. Spaan, Luis Merino, and Aníbal Ollero. Decentralized multi-robot cooperation with auctioned POMDPs. *International Journal of Robotics Research*, 32(6):650–671, 2013.

[10] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. Learning multi-agent state space representations. In *AAMAS*, pages 715–722, 2010.

[11] Jilles S. Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Exploiting separability in multiagent planning with continuous-state MDPs. In *Proceedings of the Thirteenth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1281–1288, 2014.

[12] Jilles Steeve Dibangoye, Christopher Amato, and Arnaud Doniec. Scaling up decentralized MDPs through heuristic search. In *UAI*, pages 217–226, 2012.

[13] Prashant Doshi, Yifeng Zeng, and Qiongyu Chen. Graphical models for interactive POMDPs: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3):376–416, 2008.

[14] Greg Droge and Magnus Egerstedt. Adaptive look-ahead for robotic navigation in unknown environments. In *IROS*, pages 1134–1139, 2011.

[15] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[16] Geoffrey J. Gordon, Pradeep Varakantham, William Yeoh, Hoong Chuin Lau, Ajay S. Aravamudhan, and Shih-Fen Cheng. Lagrangian relaxation for large-scale multi-agent planning. In *2012 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology, IAT 2012, Macau, China, December 4-7, 2012*, pages 494–501, 2012.

[17] Olivier Guéant, Jean-Michel Lasry, and Pierre-Louis Lions. Mean field games and applications. In *Paris-Princeton Lectures on Mathematical Finance 2010*, volume 2003 of *Lecture Notes in Mathematics*, pages 205–266. Springer Berlin Heidelberg, 2011.

[18] Carlos Guestrin, Shobha Venkataraman, and Daphne Koller. Context specific multiagent coordination and planning with factored MDPs. In *AAAI*, pages 253–259, 2002.

[19] Jesse Hoey, Robert St-Aubin, Alan J. Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *UAI*, pages 279–288, 1999.

[20] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2006.

[21] Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[22] Nyree Lemmens and Karl Tuyls. Stigmergic landmark optimization. *Advances in Complex Systems*, 15(8), 2012.

[23] Han lim Choi, Luc Brunet, and Jonathan P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 2009.

[24] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *AAAI*, pages 2017–2023, 2012.

[25] Francisco S. Melo and Manuela Veloso. Learning of coordination: exploiting sparse interactions in multiagent systems. In *AAMAS*, pages 773–780, 2009.

[26] Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[27] Jeff G. Schneider, Weng-Keen Wong, Andrew W. Moore, and Martin A. Riedmiller. Distributed value functions. In *Proc. of the International Conference on Machine Learning*, pages 371–378, 1999.

[28] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1–2):231 – 252, 1995.

[29] Jason Sleight and Edmund H. Durfee. A decision-theoretic characterization of organizational influences. In *AAMAS*, pages 323–330, 2012.

[30] Matthijs T. J. Spaan and Francisco S. Melo. Interaction-driven Markov games for decentralized

multiagent planning under uncertainty. In *AAMAS*, pages 525–532, 2008.

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[32] Pradeep Varakantham, Shih-Fen Cheng, Geoffrey J. Gordon, and Asrar Ahmed. Decision support for agent populations in uncertain and congested environments. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[33] Jianhui Wu and Edmund H. Durfee. Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, 38:415–473, 2010.

[34] Yu Zhang and Lynne E. Parker. Task allocation with executable coalitions in multirobot tasks. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 3307–3314, 2012.