Effective Approximations for Planning with Spatially Distributed Tasks

D. Claes[a] P. Robbel[b] F. A. Oliehoek[a] D. Hennes[c] K. Tuyls[a]

[a] *Department of Knowledge Engineering, Maastricht University*
[b] *Massachusetts Institute of Technology*
[c] *European Space Agency, ESTEC*

Abstract

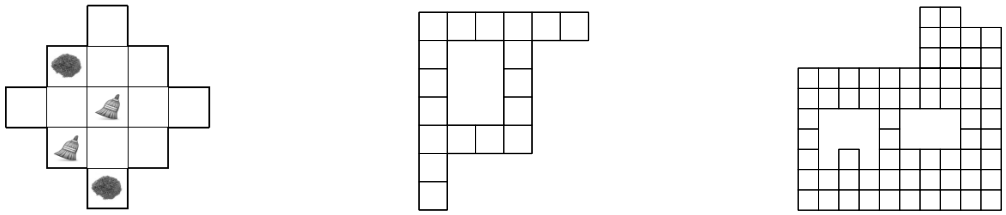
Planning in cooperative multiagent systems can be neatly formalized using Multi-Agent MDPs, but solving these models is computationally costly. This paper introduces a sub-class of problems called spatial task allocation problems (SPATAPS) that model problems in which a team of agents has to service a dynamically changing set of tasks that is spatially distributed in the environment. We propose 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.

1 Introduction

Sequential decision making in cooperative multiagent systems (MASs) is an active area of research. To deal with uncertain outcomes of actions (e.g., due to wheel slip), researchers have extended Markov decision processes (MDP) [12] to the multiagent case resulting in multiagent MDPs (MMDPs) [2]. However, even though solving an MDP can be done in polynomial time, the MMDP suffers from the fact that its state and action spaces are huge - the number of joint actions is exponential in the number of agents, and the number of states is exponential in the number of state variables or *factors*- rendering classical solution methods impractical for the most interesting problems. To overcome these barriers, we identify an sub-class of MMDPs that we refer to as *Spatial Task Allocation Problems (SPATAPS)*, and we develop a number of on-line planning approximations that are tailored to exploit the characteristics of these problems.

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. Since there are nearly no restrictions on the sort of tasks, SPATAPS provide a very powerful and general model. A direct result is that, even though SPATAPS are a special case of (factored) MMDPs, they are not a special case that is easier to solve optimally in general. However, as we demonstrate in this paper, it is possible to exploit the key characteristics of SPATAPS—independence of agent movement and the locality of tasks—*approximately* during online planning. In particular, we propose two approximation strategies that are particularly suitable for these problems.

First, we introduce *phase-myopic approximation*, which restrict planning effort to the current 'task phase': the set of currently active tasks. Second, the combination of the high degree of independence of agents and spatial distribution of agents and tasks make SPATAPS ideal candidates for distributed online planning via *subjective* or *self-absorbed approximations*. We investigate such approximations 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 future value achievable from tasks that are likely to be serviced by other agents. Such approximate *empathic* reasoning was recently exploited in the context of multi-robot exploration [10] by making use of a modification of distributed value functions [13] henceforth referred to as MDVF. We introduce a simplification of this algorithm that we refer to as *empathy by fixed weight discounting (E-FWD)*.



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

Figure 1: Three SPATAP environments.

While phase-myopic approximations and subjective approximations each bring improvements in planning efficiency, these are not enough to make the planning problem tractable. A crucial contribution of our approach is therefore the combination of these techniques, which forms the basis of an approximate online planning technique without any exponential dependence on the number of agents or number of state factors. The empirical evaluation shows that these techniques yield near-optimal solutions and is highly scalable, while outperforming the state-of-the-art.

2 Spatial Task Allocation Problems

The problems we consider in this paper describe a set of spatially distributed tasks that a team of agents needs to solve. As a running example, we consider a dirt cleaning scenario where cleaning robots cooperate to address dirt-removal tasks that are spatially distributed in the environment (see Figure 1). A key characteristic of such problems is that the outcome of an action is uncertain (cleaning the dirt may fail with some probability), and new tasks can appear due to unforeseen exogenous events (e.g., a human spilling some dirt). As such, SPATAPS can be seen as a special case of MMDP:

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 [1], since agents in an MMDP can observe the (global) state. A (joint) policy in an MMDP $\pi : \mathcal{S} \rightarrow \mathcal{A}$ 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

$$Q^t(s,a) = R(s,a) + \sum_{s'} \Pr(s'|s,a) V^{t+1}(s'). \tag{2.1}$$

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 [12]. 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 in practice.

SPATAPS are 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

Table 1: Sizes of the state and actions spaces of the considered models. $\mathcal{A}_*$ 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	$\lvert\mathcal{L}\rvert \cdot \lvert\mathfrak{T}\rvert^{k}$	$\lvert\mathcal{A}_*\rvert$

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.[1] Each agent i can perform movement and task actions.

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

$$P(\lambda',\tau'|\lambda,\tau,a) = \left[\prod_{x\in\mathcal{L}} p_x^T(\tau'_x|\tau_x,\lambda,a)\right]\left[\prod_{i\in\mathcal{D}} p_i^M(\lambda'_i|\lambda_i,a_i)\right] \tag{2.2}$$

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 (e.g., somebody spilling dirt).

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

$$R(s,a) = \left[\sum_{x\in\mathcal{L}} R_x^T(\tau_x,\lambda,a)\right] + \left[\sum_i R_i^M(\lambda_i,a_i)\right]. \tag{2.3}$$

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, also those 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 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, it 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 $R_x^T(\tau_x,\lambda_{\mathbb{L}},a_{\mathbb{L}})$.

We would hope that the special structure of SPATAPS might make them easier to solve. Unfortunately, this is in general not the case:

Theorem 1. *Optimally solving a SPATAP is MMDP-hard.*

Proof Sketch. 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.

[1] For some tasks it may not be necessary to explicitly perform a task action, e.g., in exploration just reaching a location is sufficient.

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

3 Subjective Approximations

The first set of techniques by which we bring computational leverage to the (online) planning process are subjective approximations, which aim to address the complexity that is introduced by 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.

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. 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

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

$$R_i^{SA}(s_i,a_i) = \left[\sum_{x \in \mathcal{L}} R_x^{T,SA}(\tau_x,\lambda_i,a_i) \right] + R_i^M(\lambda_i,a_i). \tag{3.1}$$

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 [6], e.g., by assuming some (e.g. uniform) distribution over λ_{-i},a_{-i}.

Formally, we define a *subjective MDP (S-MDP)* for agent i as a tuple $\left\langle \mathcal{S}_s, \mathcal{A}_i, p_i^{SA}, R_i^{SA} \right\rangle$, 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 (2.1).

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-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.

Empathy by Predicting other Agents' Locations. The key idea that allows us to take into account interactions without falling back in the complexity of exponentially many joint actions is that, from the perspective of one agent, in order to compute a best-response 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.

In particular, from the perspective of an agent i, we want to be able to predict the location λ_j^t of an agent j, 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., 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 [10]. 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 and take actions according to a Boltzmann policy using the self-absorbed agent approximation V_{SA}, which prevents this problem, as illustrated in Fig. 2 (middle). When there are multiple agents present, we can accumulate the presence mass distributions and therefore do not need to take every single agent into account. We use the accumulated presence mass distribution for the next model.

Empathy by Fixed Weight Discounting. The MDVF [10] approach is inspired by DVFs [13] and allows agents to share their value function. This is achieved by using a different value function (namely 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.

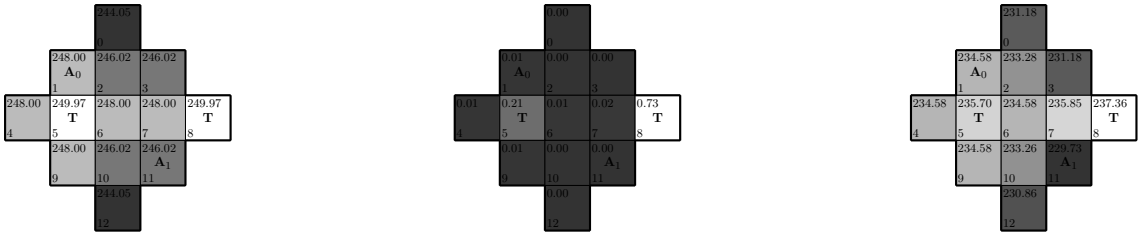


Figure 2: Left: A sample state for a diamond shaped gridworld with two agents (A_0, A_1) and two active task locations and V_{SA} for the current state. Middle: The presence mass of A_1 from the viewpoint of A_0. Right: The discounted value function V_{EFWD} for A_0 resulting for this configuration.

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

$$Q_i^{FWD,t}(s_i,a_i) = R^{SA}(s_i,a_i) + \sum_{s'} p^{SA}(s_i'|s_i,a_i)\left[\left(1 - f_i \sum_{j\neq i} \Pr(s_j^{t+1} = s_i'|s^0)\right) V_i^{FWD,t+1}(s_i')\right]. \quad (3.2)$$

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 just the presence mass of an agent j being at the location specified by s_i', 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 [10] 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 Phase-Myopic 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, we focus only on the current 'task phase', i.e., the *current* set of tasks (i.e., only on 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.

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 of joint locations $\mathcal{L}^{|\mathcal{D}|}$ 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 2.2 (and 2.3) 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, in the worst case, there are active tasks everywhere and there is no reduction. Also, it does not address the large joint action space (see Table 1).

Subjective Phase MDPs. Realizing that subjective and phase-myopic approximations yield complementary gains, we propose to combine both approximations in a formalism that we refer to as *subjective phase 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-myopic assumption (see Table 1).

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.

kSP-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-myopic 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.

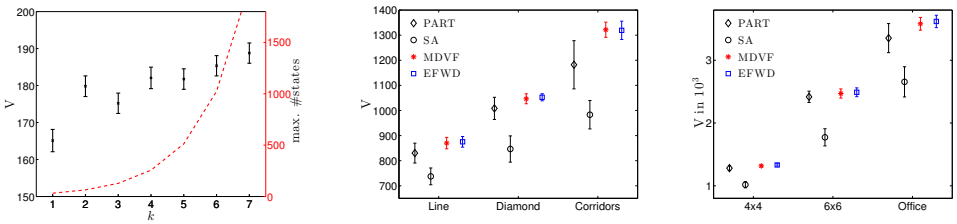
5 Experiments

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 simulator in Python 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. Unless reported differently, we use $k = 4$.

In order to asses 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 [7], 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. 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 E-FWD even yields slightly higher rewards than the more complex MDVF.

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 3a, are averages 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



(a) varying k (b) Different problem sizes

Figure 3: (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.

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., office world has $6.10e+30$ states and 15625 actions as shown in Table 2b. In fact, few methods can deal with such large problem. The only other approach that we found to offer the required scalability is the 'partition organization' [14]. 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 straight forward partitioning, i.e. very symmetric worlds.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, we addressed this 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 3b 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 and all agents are able to clean every second time step (i.e., in one step each agent uses a 'teleport' move they can reach the location of a next task, which is then serviced in the other step). Clearly, this upper bound is a vast overestimation of the optimal value, since each time step new tasks appear at random locations and agents need more than one step travel times to these tasks. However, the proposed approximations yield rewards relatively close to this (unrealistic) upper bound. Generally over 90% of the upper bound is achieved in the smaller worlds up to 4x4, and about 75% in the 6x6 world and about 70% in the "Office" world are achieved.

6 Related Work

In this work, we define approximate models which we can solve optimally. This should be contrasted with efforts to approximately solve exact models (e.g. [8]). Combining such approaches (approximately solving the approximate models) may lead to even further scalability, required for real-life problems. 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., [4], Dec-MDP solution methods do not nearly scale to problems of the size considered here. While there have been other approximate methods for solving MMDPs, these typically depend on pre-specifying the fixed, or context-dependent coordination structure [9, 15]. For SPATAPS, however, fixed coordination structures are a poor choice and the number of contexts to be considered is huge. To overcome the problem of pre-specifying interaction structures one can try to learn them [11, 3], 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 what tasks they service at all states. SPATAPS relate to resource allocation [16] (agents can be interpreted as resources that are assigned to different tasks). We, however, allow reallocation at every time step and consider spatially distributed tasks and travel times. Finally, the subjective approximations presented in this paper can be interpreted as online planning for a special instance of a level 1 interactive POMDP [5]. In contrast to standard interactive POMDP solution methods, however, we propose dedicated approximation algorithms that exploit the characteristics of SPATAPS.

7 Conclusions & Future Work

This paper introduces SPATAPS, a general sub-class of MMDPs suitable for domains such as multi-robot exploration. To combat the complexity of general MMDP algorithms, we propose to use phase-myopic and subjective approximations, and combine both to yield an efficient online planning method for SPATAPS. Current work investigates a theoretical understanding of these methods including which guarantees, i.e. bounds, can be given for the proposed approaches. The transition of the dirt-world example to a real-world application, as well as identifying methods for 'positive interaction' settings (e.g., there are joint tasks for which two agents are required), and mixtures of negative and positive interactions are promising directions for future work.

References

[1] 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.

[2] 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.

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

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

[5] 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.

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

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

[8] 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 Berlin / Heidelberg, 2006.

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

[10] 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.

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

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

[13] 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.

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

[15] Matthijs T. J. Spaan and Francisco S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *AAMAS*, pages 525–532, 2008.

[16] 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.