Lifting Partially Observable Stochastic Games

Nazlı Nur Karabulut, Tanya Braun

Computer Science Department, University of Münster, Münster, Germany
nnur.karabulut@uni-muenster.de, tanya.braun@uni-muenster.de

Abstract. Partially observable stochastic games (POSGs) are a Markovian formalism used to model a set of agents acting in a stochastic environment, in which each agent has its own reward function. As is common with multi-agent decision making problems, the model and runtime complexity is exponential in the number of agents, which can be prohibitively large. Lifting is a technique that treats groups of indistinguishable instances through representatives if possible, yielding tractable inference in the number of objects in a model. This paper applies lifting to the agent set in POSGs, yielding so-called isomorphic POSGs that have a model complexity no longer dependent on the number of agents, and presents a lifted solution approach that exploits this lifted agent set for space and runtime gains.

1 Introduction

Multi-agent decision making lies at the heart of artificial intelligence (AI) and is one of the most challenging problems in AI research. A common formalisation lies in partially observable stochastic games (POSGs). In POSGs, a set of agents acts in an environment modelled as a stochastic Markovian process, each with the goal to optimise its own reward function. POSGs are a generalisation of other formalisms based on Markovian modelling, from single-agent decision making in Markov decision processes (MDPs) and partially observable MDPs (POMDPs) to multi-agent decision making in decentralised POMDPs (DecPOMDPs), in which the agent set shares a single reward function. Both POSGs and DecPOMDPs have in common that their space and runtime complexity depends exponentially on the number of agents, making them all but unsolvable except for small problem sizes.

Unfortunately, there are applications that have an agent set size in the thousands and still require working with such a model-based approach to be able to argue about its reasoning. One such example are nanoscale medical systems, e.g., used for diagnosis or treatment [13]. Here, the idea is that a network of nanoagents is used to detect markers for a specific disease or deliver medicine to locations in the cardio-vascular system. Given its application in the medical domain, one needs to find accurate solutions [24] that can be justified and allows for arguing about agent set sizes as to not poison a patient. While the agent sets are typically very large, an upside is that they rarely come without any structure. Typically and true for nanoscale systems, there are a small number of different types of agents to fulfil specific purposes within the system.

One promising direction that actually uses this structure has been explored when lifting the agent set of DecPOMDPs [3]. Lifting is a technique from probabilistic inference in probabilistic graphical models (PGMs), in which indistinguishable random variables are encoded using logical variables and treated through a representative [17], which leads to tractability in terms of domain sizes of those logical variables [15]. Indistinguishability of random variables is characterised by subgraph isomorphisms in the PGM between the variables and identical factors describing the influences between a variable and its neighbours in those subgraphs. In terms of the probabilistic behaviour, lifting assumes conditional independences between these indistinguishable instances, which yields the isomorphic structure in the graph. Lifting the agent set in DecPOMDPs leads to conditions for the transition, sensor, and reward function, under which representatives for groups of agents can be used during computations, yielding tractability in terms of agent numbers [3], which partitions the agent set.

However, lifting DecPOMDPs stops one step short of fully applying the lifting idea to multi-agent decision making, by still having a joint reward function. In this paper, we go this next step in providing each agent partition with its own reward function, which makes the formalisation a POSG. While lifting POSGs as an extension of lifted DecPOMDPs can be thought of as a theoretical endeavour, doing so has practical value as lifting POSGs can be helpful in modelling, e.g., nanoscale medical systems, where different types of agents may interact with each other or external factors and may have competing or complementary objectives. Additionally, we can further reduce the complexity of these models while appropriately modelling diverse goals of the agents, ensuring accurate and efficient decision-making in dynamic medical environments. Specifically, the contributions are twofold: (i) isomorphic POSGs based on a partitioning of the agent set for an explicit encoding of structure in the agent set and (ii) a lifted version of the multi-agent dynamic programming operator [7] to solve isomorphic POSGs. To the best of our knowledge, we are the first to apply the lifting idea to POSGs and show its potential for solving problem instances of new scale.

The remainder of the paper is structured as follows: We start with a recap of POSGs as well as dynamic programming for POSGs. Then, we define indistinguishability between agents and present isomorphic POSGs. Afterwards, we provide a lifted dynamic programming operator and end with a discussion.

2 Preliminaries

This section defines POSGs and recaps dynamic programming for POSGs. We look through a PGM lens for definitions, which are based on [7] and [18], using random variables, S, which can take discrete values, referred to as range, $ran(S) = \{s_1, \dots, s_n\}$. So-called *decision random variables* have actions as ranges. Setting S to a value $s \in ran(S)$ (an *event*) is denoted as $S = s$ or s for short if S is clear from its context. We denote sequences over a discrete time interval $[t_s, t_e]$ with subscript $t_s{:}t_e$, e.g., $(S_{t_s}, \dots, S_{t_e}) = S_{t_s:t_e}$. We use π to denote projection and θ to denote permutation.

2.1 Partially Observable Stochastic Game

A POSG encodes a set of agents working towards their own goals, represented through individual reward functions.

Definition 1. *A* POSG M *is a tuple* $(\boldsymbol{I}, S, \boldsymbol{A}, \boldsymbol{O}, T, \boldsymbol{R})$*, with*

- $\boldsymbol{I}$ *a set of* N *agents,*
- S *a random variable with a set of states as range,*
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{I}}$ *a set of decision random variables* A_i*, one for each agent* $i \in \boldsymbol{I}$*, each with a set of local actions as range, with* $ran(\boldsymbol{A}) = \times_{i \in \boldsymbol{I}} ran(A_i)$ *the set of* joint actions*,*
- $\boldsymbol{O} = \{O_i\}_{i \in \boldsymbol{I}}$ *a set of random variables* O_i*, each with a set of local observations as range, with* $ran(\boldsymbol{O}) = \times_{i \in \boldsymbol{I}} ran(O_i)$ *the set of* joint observations*,*
- $T(S', S, \boldsymbol{A}, \boldsymbol{O}) = P(S', \boldsymbol{O} \mid S, \boldsymbol{A})$*, called transition function to give it a unique name, which is a probability distribution denoting the probability of moving from state* s *with joint action* $\boldsymbol{a}$ *to state* s' *and making the joint observation* $\boldsymbol{o}$*, with* $T(S_0, ., ., .) = P(S_0)$ *referring to a state prior, and*
- $\boldsymbol{R} = \{R_i(S, \boldsymbol{A})\}_{i \in \boldsymbol{I}}$ *a set of reward functions, one for each agent* $i \in \boldsymbol{I}$*.*

Optional are a finite horizon τ*, a discount factor* $\gamma \in [0, 1]$ *(default* 1*), and an error margin* $\epsilon > 0$*. Each agent* $i \in \boldsymbol{I}$ *has a local policy* $\pi_i : ran((O_{i,0:t})) \mapsto ran(A_i)$ *mapping observation histories* $o_{i,0:t}, t \leq \tau - 1$*, to actions* a*, with* $\boldsymbol{\pi} = (\pi_i)_{i \in \boldsymbol{I}}$ *a* joint policy*. A set of policies for agent* i *is denoted by* Π_i*. A set or sequence of elements over all agents except agent* i *is denoted by subscript* $-i$*.*

Each agent has its own set of actions and observations and its own reward function[1] whereas the state is joint. An agent's belief $b_i(s, \boldsymbol{\pi}_{-i})$ is a probability distribution over the state space $ran(S)$ and the other agents' policies $\boldsymbol{\Pi}_{-i}$. The value V_i of a belief b_i is defined by

$$V_i(b_i) = \max_{\pi_i \in \Pi_i} \sum_{s \in ran(S)} \sum_{\boldsymbol{\pi}_{-i} \in \boldsymbol{\Pi}_{-i}} b_i(s, \boldsymbol{\pi}_{-i}) V_i'(s, \pi_i, \boldsymbol{\pi}_{-i})$$

where $V_i'(s, \pi_i, \boldsymbol{\pi}_{-i})$ refers to the value of acting according to joint policy $\pi_i, \boldsymbol{\pi}_{-i}$ in state s and incorporates the reward functions in each step. Modelling-wise, DecPOMDPs only differ from POSGs by having a single reward function for all agents. However, having self-interested agents has consequences for potential solution methods: Whereas a DecPOMDP allows for brute-force joint policy evaluation, POSGs no longer enable such computation as it is no longer possible to define what an optimal policy is, which is why some of the most common DecPOMDP solution methods do not translate to the POSG setting [16]. In POSGs, one usually aims for a Nash equilibrium and possibly a Pareto optimal one, meaning that in an equilibrium, no agent has an incentive to change its policy without any agent changing its policy. The one solution method that both formalisms share is dynamic programming, which we recap next.

[1] $A_i = A_j$, $O_i = O_j$, or $R_i = R_j$, $i, j \in \boldsymbol{I}$, are possible but not mandatory.

Algorithm 1 Multi-agent Dynamic Programming Operator

```
function MA-DP(set of policies $\mathit{\Pi}_i^{t-1}$ for each agent $i \in \boldsymbol{I}$ with value vectors $\boldsymbol{V}_i^{t-1}$)
    $\mathit{\Pi}_i^t \leftarrow$ Perform exhaustive backup using $\mathit{\Pi}_i^{t-1}$ for each agent $i \in \boldsymbol{I}$
    $\boldsymbol{V}_i^t \leftarrow$ Calculate new value vectors for each agent $i \in \boldsymbol{I}$
    while $\exists \pi_{i,j}^t \in \mathit{\Pi}_i^t$ : Eq. (1) holds do
        $\mathit{\Pi}_i^t \leftarrow \mathit{\Pi}_i^t \setminus \{\pi_{i,j}^t\}$, $\boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^t \setminus \{v_{i,j}^t\}$
    return $\{(\mathit{\Pi}_i^t, \boldsymbol{V}_i^t)\}_{i \in \boldsymbol{I}}$
```

2.2 Dynamic Programming for POSGs

Solving a POSG with horizon τ can be done through a combination of dynamic programming and pruning [7]. Such a routine involves an iterative application of a dynamic programming operator where in each iteration $t \in \{0, \ldots, \tau - 1\}$, the operator, first, does an exhaustive backup that generates all possible policies of depth t for each agent given the existing policies of depth $t-1$, and second, prunes policies that are very weakly dominated until no more policies can be pruned. In the following, we define the dynamic programming operator as well as what constitutes weak domination between policies.

Algorithm 1 shows the dynamic programming operator. It takes a set of policies $\mathit{\Pi}_i^{t-1} = \{\pi_{i,j}^{t-1}\}_{j=1}^m$ of depth $t-1$ for each agent i. For each policy, there exists a value vector $v_{i,j}^{t-1}$ that denotes the value of that policy in each possible combination of state $s \in ran(S)$ and policies of the other agents $\boldsymbol{\pi}_{-i} \in \boldsymbol{\mathit{\Pi}}_{-i}^{t-1}$. We denote the set of corresponding value vectors by $\boldsymbol{V}_i^{t-1}$. The first step involves generating all possible policies of depth t given the policies of depth $t-1$, expanding all existing policies with all possible combinations of observations and actions. Pruning involves finding policies that are very weakly dominated by other policies, meaning that over the complete space of $S \times \boldsymbol{\mathit{\Pi}}_{-i}^t$ there is always another strategy with higher value. Formally, a policy $\pi_{i,j}^t$ is pruned if the following holds, which can be solved using a linear programme:

$$\forall s \in ran(S), \boldsymbol{\pi}_{-i}^t \in \boldsymbol{\mathit{\Pi}}_{-i}^t \exists \pi_{i,j'}^t \in \mathit{\Pi}_i^t : V_i(s, \pi_{i,j}^t, \boldsymbol{\pi}_{-i}^t) \leq V_i(s, \pi_{i,j'}^t, \boldsymbol{\pi}_{-i}^t), \quad (1)$$

3 Lifting Partially Observable Stochastic Games

POSGs are characterised by a model and runtime complexity for exact solution methods that is exponential in the number of agents. However, large multi-agent systems such as nanoscale medical systems often exhibit some form of structure among its agent set that we can exploit for efficiency gains. If assuming that there are groups of agents that are essentially indistinguishable, i.e., one expects them to behave identically if faced with the same situation, it is possible to reduce model and runtime complexity by being able to work with representatives for groups. A complexity reduction from exponential to logarithmic is possible for so-called isomorphic DecPOMDPs [3] if assuming that the number of groups K is much smaller than the overall number of agents N, i.e., $K \ll N$. In the following, we apply the idea to POSGs.

3.1 Indistinguishable Agents for Agent Tractability

Before considering agents that are indistinguishable within a group that is distinguishable from the next group, we consider indistinguishability between two agents. Intuitively, two agents are indistinguishable if they act the same way given the same situation and the possible effect is identical, which also means that they have the same set of policies available. This translates into the following aspects, which we explain below: (i) The available actions and observations as well as the reward function are identical. (ii) The behaviour in functions T and R is symmetric (what that means we see below). (iii) The two agents are conditionally independent in T and R given the state and the other agents' actions. The first item is essentially a precondition for the other two items as two agents cannot be indistinguishable in any regard if they have a different set of actions and observations and a different reward function, making them automatically distinguishable. Also, if the available actions and observations differ, the histories cannot be the same and the agents cannot perform the same action. The second item describes that in T and R, those two agents could switch their actions, while the resulting distribution over state and joint observation as well as the resulting reward would be the same, making them indistinguishable. The last item is a consequence of saying that the two agents have the same policies available to them. The same set of policies implies that both agents would act identically given the same situation, which requires conditional independence between those two agents. Next, we formalise these conditions for two agents before generalising the conditions for groups of agents.

The first item, which is rather straight-forward, can be formalised by stating that for two indistinguishable agents $i, j \in \boldsymbol{I}$,

$$\begin{aligned} &ran(A_i) = ran(A_j) \wedge ran(O_i) = ran(O_j) \\ &\wedge \forall s \in ran(S), \boldsymbol{a} \in ran(\boldsymbol{A}) : R_i(s, \boldsymbol{a}) = R_j(s, \boldsymbol{a}). \end{aligned} \tag{2}$$

The second item, symmetric behaviour, manifests itself in the functions T and R by being able to exchange the actions and observations of two indistinguishable agents and still receiving the same probability and reward. Formally, this behaviour between two indistinguishable agents $i, j \in \boldsymbol{I}$ can be characterised as follows, with $R_h = R_i = R_j$ due to Eq. (2):

$$\begin{aligned} &\forall \boldsymbol{a} \in ran(\boldsymbol{A}), \boldsymbol{a} = (a_i, a_j, \boldsymbol{a}_{-i,-j}) : \forall \boldsymbol{o} \in ran(\boldsymbol{O}), \boldsymbol{o} = (o_i, o_j, \boldsymbol{o}_{-i,-j}) : \\ &\qquad T(s', s, a_i, a_j, \boldsymbol{a}_{-i,-j}, \boldsymbol{o}) = T(s', s, a_j, a_i, \boldsymbol{a}_{-i,-j}, \boldsymbol{o}) \\ &\qquad T(s', s, \boldsymbol{a}, o_i, o_j, \boldsymbol{o}_{-i,-j}) = T(s', s, \boldsymbol{a}, o_j, o_i, \boldsymbol{o}_{-i,-j}) \\ &\qquad R_h(s, a_i, a_j, \boldsymbol{a}_{-i,-j}) = R_h(s, a_j, a_i, \boldsymbol{a}_{-i,-j}), h \in \{i, j\} \end{aligned} \tag{3}$$

The third item, conditional independence means distributions factorise according to $P(A, B \mid C) = P(A \mid C) \cdot P(B \mid C)$. Conditional independence between two indistinguishable agents $i, j \in \boldsymbol{I}$ affects the observations o_i, o_j and actions a_i, a_j, which are conditionally independent given the state and the other agents' actions. This independence yields a factorisation of T, where o_i, o_j as well as o_i (o_j) and a_j (a_i) are conditionally independent, and a factorisation of R regarding a_i, a_j, which looks as follows:

$$\begin{aligned}
&\forall \boldsymbol{a} \in ran(\boldsymbol{A}), \boldsymbol{a} = (a_i, a_j, \boldsymbol{a}_{-i,-j}) : \forall \boldsymbol{o} \in ran(\boldsymbol{O}), \boldsymbol{o} = (o_i, o_j, \boldsymbol{o}_{-i,-j}) : \\
&\quad T(s', s, a_i, a_j, \boldsymbol{a}_{-i,-j}, o_i, o_j, \boldsymbol{o}_{-i,-j}) = P(s', o_i, o_j, \boldsymbol{o}_{-i,-j} \mid s, a_i, a_j, \boldsymbol{a}_{-i,-j}) \\
&\quad = P_i(s', o_i, \boldsymbol{o}_{-i,-j} \mid s, a_i, \boldsymbol{a}_{-i,-j}) \cdot P_j(s', o_j, \boldsymbol{o}_{-i,-j} \mid s, a_j, \boldsymbol{a}_{-i,-j}) \qquad (4) \\
&\quad R_h(s, a_i, a_j, \boldsymbol{a}_{-i,-j}) = R_h^i(s, a_i, \boldsymbol{a}_{-i,-j}) + R_h^j(s, a_j, \boldsymbol{a}_{-i,-j}), h \in \{i, j\}
\end{aligned}$$

Note the dimension reductions in the factorisations compared to T, R, which we use for efficiency when applied to groups of agents. Equations (2) to (4) together actually yield $P_i = P_j$ and $R_h^i = R_h^j$, which we show next.

Lemma 1. *If Eqs.* (2) *to* (4) *hold, then the factors* P_i, P_j *and summands* R_h^i, R_h^j *in Eq.* (4) *are identical.*

Proof. As mentioned above, Eq. (2) is a prerequisite for indistinguishability and, specifically, for Eq. (3). If Eq. (3) holds, permutations of actions and observations for indistinguishable agents map to the same probability p in T and reward r in R. Let us first show that $P_i = P_j$ for the first line of Eq. (3). Inserting Eq. (4) leads to:

$$\begin{aligned}
&T(s', s, a_i, a_j, \boldsymbol{a}_{-i,-j}, \boldsymbol{o}) = T(s', s, a_j, a_i, \boldsymbol{a}_{-i,-j}, \boldsymbol{o}) = p \\
\Leftrightarrow &P_i(s', \boldsymbol{o} \mid s, a_i, \boldsymbol{a}_{-i,-j}) \cdot P_j(s', \boldsymbol{o} \mid s, a_j, \boldsymbol{a}_{-i,-j}) = p^{a_i} \cdot p^{a_j} \\
&= P_j(s', \boldsymbol{o} \mid s, a_j, \boldsymbol{a}_{-i,-j}) \cdot P_i(s', \boldsymbol{o} \mid s, a_i, \boldsymbol{a}_{-i,-j}) = p^{a_j} \cdot p^{a_i} = p
\end{aligned}$$

There are two cases to consider, one where $a_i = a_j = a$ (or a') and one where $a_i = a \neq a_j = a'$, which leads to the following set of equations:

$$\begin{aligned}
P_i(. \mid s, a, \boldsymbol{a}_{-i,-j}) \cdot P_j(. \mid s, a, \boldsymbol{a}_{-i,-j}) &= p_i^a \cdot p_j^a = p \Leftrightarrow p_j^a = \frac{p}{p_i^a} \Leftrightarrow p_i^a = \frac{p}{p_j^a} \\
P_i(. \mid s, a, \boldsymbol{a}_{-i,-j}) \cdot P_j(. \mid s, a', \boldsymbol{a}_{-i,-j}) &= p_i^a \cdot p_j^{a'} = p' \\
P_i(. \mid s, a', \boldsymbol{a}_{-i,-j}) \cdot P_j(. \mid s, a, \boldsymbol{a}_{-i,-j}) &= p_i^{a'} \cdot p_j^a = p' \\
P_i(. \mid s, a', \boldsymbol{a}_{-i,-j}) \cdot P_j(. \mid s, a', \boldsymbol{a}_{-i,-j}) &= p_i^{a'} \cdot p_j^{a'} = p'' \Leftrightarrow p_j^{a'} = \frac{p''}{p_i^{a'}} \Leftrightarrow p_i^{a'} = \frac{p''}{p_j^{a'}}
\end{aligned}$$

The two lines in the middle say that $p_i^{a'} \cdot p_j^a = p_i^a \cdot p_j^{a'} = p'$, which yields the following if replacing p_i^a and $p_i^{a'}$ as well as p_j^a and $p_j^{a'}$ with the corresponding fractions from above:

$$\begin{aligned}
&p_i^{a'} \cdot \frac{p}{p_i^a} = p_i^a \cdot \frac{p''}{p_i^{a'}} = p' \Leftrightarrow (p_i^{a'})^2 \cdot p = (p_i^a)^2 \cdot p'' = p_i^{a'} \cdot p_i^a \cdot p' \\
&\frac{p''}{p_j^{a'}} \cdot p_j^a = \frac{p}{p_j^a} \cdot p_j^a = p' \Leftrightarrow (p_j^a)^2 \cdot p'' = (p_j^{a'})^2 \cdot p = p_j^{a'} \cdot p_j^a \cdot p'
\end{aligned}$$

Since the first halves before the equivalence sign both equal p', we have that the parts with p in them are equal, which we can further reformulate:

$$\frac{p_i^{a'}}{p_i^a} \cdot p = \frac{p_j^{a'}}{p_j^a} \cdot p \Leftrightarrow \frac{p_i^{a'}}{p_i^a} = \frac{p_j^{a'}}{p_j^a} \Leftrightarrow p_j^a = \frac{p_i^a}{p_i^{a'}} \cdot p_i^{a'}$$

The second halves after the equivalence sign are also equal, which yields:

$$p_i^{a'} \cdot p_i^a \cdot p' = p_j^{a'} \cdot p_j^a \cdot p' \Leftrightarrow p_j^a = \frac{p_i^{a'} \cdot p_i^a}{p_j^{a'}}$$

Taking both expressions for p_j^a, we get

$$\frac{p_i^a}{p_i^{a'}} \cdot p_j^{a'} = \frac{p_i^{a'} \cdot p_i^a}{p_j^{a'}} \Leftrightarrow p_i^a \cdot (p_j^{a'})^2 = p_i^a \cdot (p_i^{a'})^2 \Leftrightarrow p_j^{a'} = p_i^{a'}$$

which leads to $P_i = P_j$. The same reformulations can be done for the other two lines of Eq. (3), showing that $P_i = P_j$ as well as $R_i = R_j$ given Eq. (4). □

3.2 Partitions of Indistinguishable Agents

The previous section groups two agents together. The idea is of course to have a small number of groups with many indistinguishable agents in each group. As such, Eqs. (2) to (4) have to hold for all agents within a group, which leads to a partitioning of the agent set.

The set of agents $\boldsymbol{I}$ partitions into K sets $\mathfrak{I}_k$, i.e., $\boldsymbol{I} = \bigcup_{k=1}^{K} \mathfrak{I}_k$, $\mathfrak{I}_k \neq \emptyset$, and $\forall k, l \in \{1, \dots, K\}, k \neq l : \mathfrak{I}_k \cap \mathfrak{I}_l = \emptyset$, and it holds for each partition $\mathfrak{I}_k$ of indistinguishable agents that:

$$\begin{aligned} \forall i, j \in \mathfrak{I}_k : ran(A_i) = ran(A_j) \wedge ran(O_i) = ran(O_j) \\ \wedge\, \forall s \in ran(S), \boldsymbol{a} \in ran(\boldsymbol{A}) : R_i(s, \boldsymbol{a}) = R_j(s, \boldsymbol{a}), \end{aligned} \tag{5}$$

which is a straightforward generalisation of Eq. (2). The equation denotes that each agent in a partition needs to have the same actions and observations available as well as the same reward function. As such, it is sufficient to keep K decision and observation random variables A_k, O_k and reward functions R_k instead of N variables A_i and O_i and reward functions R_i.

Generalising Eq. (3) for partitions means that all permutations θ of those actions $\boldsymbol{a}_{-k}$ that belong to a partition $\mathfrak{I}_k$ in a joint action $\boldsymbol{a}$, i.e., $\boldsymbol{a}_k = \pi_{\mathfrak{I}_k}(\boldsymbol{a})$, map to the same numbers. The same holds for permutations of observations $\boldsymbol{o}_k = \pi_{\mathfrak{I}_k}(\boldsymbol{o})$. Formally, with $\boldsymbol{a}_{-k}$ ($\boldsymbol{o}_{-k}$) referring to the remaining actions (observations), i.e., $\boldsymbol{a}_{-k} = \boldsymbol{a} \setminus \boldsymbol{a}_k$ ($\boldsymbol{o}_{-k} = \boldsymbol{o} \setminus \boldsymbol{o}_k$), the condition changes to:

$$\begin{aligned} &\forall \boldsymbol{a} \in ran(\boldsymbol{A}), \boldsymbol{a} = (\boldsymbol{a}_k, \boldsymbol{a}_{-k}) : \forall \theta(\boldsymbol{a}_k) : \forall \boldsymbol{o} \in ran(\boldsymbol{O}), \boldsymbol{o} = (\boldsymbol{o}_k, \boldsymbol{o}_{-k}) : \forall \theta(\boldsymbol{o}_k) : \\ &\qquad T(s', s, \boldsymbol{a}_k, \boldsymbol{a}_{-k}, \boldsymbol{o}) = T(s', s, \theta(\boldsymbol{a}_k), \boldsymbol{a}_{-k}, \boldsymbol{o}) \\ &\qquad T(s', s, \boldsymbol{a}, \boldsymbol{o}_k, \boldsymbol{o}_{-k}) = T(s', s, \boldsymbol{a}, \theta(\boldsymbol{o}_k), \boldsymbol{o}_{-k}) \\ &\qquad R_k(s, \boldsymbol{a}_k, \boldsymbol{a}_{-k}) = R_k(s, \theta(\boldsymbol{a}_k), \boldsymbol{a}_{-k}) \end{aligned} \tag{6}$$

Generalising Eq. (4) to the partition case means that all agents within a partition are conditionally independent from each other, leading to factorising the functions T and R as follows, with with $a_i = \pi_{A_i}(\boldsymbol{a}_k)$ and $o_i = \pi_{O_i}(\boldsymbol{o}_k)$:

$$\begin{aligned}
&\forall \boldsymbol{a}_{-k} \in ran(\boldsymbol{A}_{-k}) : \forall \boldsymbol{a}_k \in ran(\boldsymbol{A}_k) : \forall \boldsymbol{o}_{-k} \in ran(\boldsymbol{O}_{-k}) : \forall \boldsymbol{o}_k \in ran(\boldsymbol{O}_k) : \\
&\qquad T(s', s, \boldsymbol{a}_k, \boldsymbol{a}_{-k}, \boldsymbol{o}_k, \boldsymbol{o}_{-k}) = \prod_{i \in \mathfrak{I}_k} P_i(s', o_i, \boldsymbol{o}_{-k} \mid s, a_i, \boldsymbol{a}_{-k}) \\
&\qquad R_k(s, \boldsymbol{a}_k, \boldsymbol{a}_{-k}) = \sum_{i \in \mathfrak{I}_k} R_k^i(s, a_i, \boldsymbol{a}_{-k}).
\end{aligned} \tag{7}$$

Given the generalisations in Eqs. (5) to (7), Lemma 1 applies to the group setting with the same argument as above.

Corollary 1. *If Eqs.* (5) *to* (7) *hold, then the factors P_i and summands R_k^i in Eq.* (7) *are identical.*

Corollary 1 allows for a more compact representation of T and R as both functions factorise into a set of functions, one for each partition. Additionally, the factorised functions have lower dimensionality, because they no longer require the joint action (and observation) as input. With $\pi_{\boldsymbol{k}}(\boldsymbol{a})$ denoting the projection of $\boldsymbol{a}$ onto the agents in $\boldsymbol{k}$ (the same for $\boldsymbol{o}$), the functions boil down to:

$$\begin{aligned}
T(s', s, \boldsymbol{a}, \boldsymbol{o}) &= \prod_{\boldsymbol{k} \in \times_{k=1}^{K} \mathfrak{I}_k} \prod_{k \in \boldsymbol{k}} P_k(s', \pi_{\boldsymbol{k}}(\boldsymbol{o}) \mid s, \pi_{\boldsymbol{k}}(\boldsymbol{a})) \\
R(s, \boldsymbol{a}) &= \sum_{\boldsymbol{k} \in \times_{k=1}^{K} \mathfrak{I}_k} R_k(s, \pi_{\boldsymbol{k}}(\boldsymbol{a}))
\end{aligned} \tag{8}$$

Corollary 2. *The complexity of Eq.* (8) *is no longer exponential in the number of agents N in a POSG M, in which Eqs.* (5) *to* (7) *hold.*

Proof. While there are $n^K \cdot K, n = \max_{k \in \{1,\ldots,K\}} |\mathfrak{I}_k|$ many factors in T (n^K summands in R), the complexity of each T_k (R_k) is down to $s \cdot o^K \cdot a^K$ ($s \cdot a^K$), which is no longer exponential in the overall number of agents N.

With indistinguishability defined on a partition level, we next define isomorphic POSGs that compactly encode this indistinguishability.

3.3 Isomorphic POSGs

POSGs that have a partitioned agent set and additionally exhibit symmetric behaviour in the functions T and R as well as conditional independence among the members of the same partition allow for a more compact representation that no longer depends on the number of agents, but only the number of partitions as seen in Eq. (8). We call such a POSG isomorphic and define it as follows:

Definition 2. *An* isomorphic *POSG $\bar{M}$ is a tuple $(\bar{\boldsymbol{I}}, S, \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}}, \bar{\boldsymbol{T}}, \bar{\boldsymbol{R}})$, with*

- $\bar{\boldsymbol{I}}$ *a* partitioning $\{\mathfrak{I}_k\}_{k=1}^K$ *of agents,* $n_k = |\mathfrak{I}_k|$ *and* $|\bar{\boldsymbol{I}}| = \sum_k n_k = N$*,*
- S *a random variable with a set of states as range as in Def. 1,*
- $\bar{\boldsymbol{A}} = \{\bar{A}_k\}_{k=1}^K$ *a set of decision random variables* $\bar{A}_k$*, each with possible* actions *as range,* $ran(A_i) = ran(\bar{A}_k) \forall i \in \mathfrak{I}_k$*, and* $ran(\boldsymbol{A}_k) = \times_{i \in \mathfrak{I}_k} ran(A_k)$*,*
- $\bar{\boldsymbol{O}} = \{\bar{O}_k\}_{k=1}^K$ *a set of random variables* $\bar{O}_k$*, each with a set of* observations *as range,* $ran(O_i) = ran(\bar{O}_k) \forall i \in \mathfrak{I}_k$*, and* $ran(\boldsymbol{O}_k) = \times_{i \in \mathfrak{I}_k} ran(O_k)$*,*
- $\bar{\boldsymbol{T}} = \{\bar{T}_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}})\}_{k=1}^K$ *a set of probability distributions* $\bar{T}_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}}) = P(S', \bar{\boldsymbol{O}} \mid S, \bar{\boldsymbol{A}})$*, and*
- $\bar{\boldsymbol{R}} = \{\bar{R}_k(S, \bar{\boldsymbol{A}})\}_{k=1}^K$ *a set of reward functions.*

Before presenting model complexity results, we show that an isomorphic POSG is equivalent to a standard (ground) POSG that fulfils Eqs. (5) to (7).

Theorem 1. *An isomorphic POSG* $\bar{M}$ *has an equivalent POSG* M*, in which Eqs.* (5) *to* (7) *hold.*

Proof. We transform the isomorphic POSG $\bar{M}$ into a first-order representation by adding logical variables for each partition to represent the agents of each partition, leading to parameterised random variables $\bar{A}_k = A_k(X_k), \bar{O}_k = O_k(X_k)$ as decision and observation random variables, with the domains of the logical variables X_k being the partitions of agents, i.e., $dom(X_k) = \mathfrak{I}_k$. Ranges are not affected. The probability distributions $T_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}})$ and reward functions $R_k(S, S', \bar{\boldsymbol{A}})$ then can be written as

$$\begin{aligned} T_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}}) &= T_k(S, S', A_1(X_1), \dots, A_K(X_K), O_1(X_1), \dots, O_K(X_K)) \\ R_k(S, S', \bar{\boldsymbol{A}}) &= R_k(S, S', A_1(X_1), \dots, A_K(X_K)) \end{aligned}$$

which is only a syntactic transformation with no consequence for the semantics of the functions. We now turn $\bar{M}$ into M by essentially grounding $\bar{M}$, which means we expand the compact representations of $\bar{M}$ into the joint representations of M: Grounding a parameterised random variable replaces each logical variable with a constant, here an agent, leading to a set of random variables $A_k(x_{k,i}), O_k(x_{k,i})$, $i \in \mathfrak{I}_k$, which is equivalent to the random variables A_i, O_i of M with Eq. (5). Grounding a function that contains parameterised random variables yields a set of functions with identical mappings and the logical variables replaced with all possible combinations of constants (agents) over the partitions, i.e., $\times_{k=1}^K dom(X_k)$, which follow multiplicative semantics in T as T is a probability distribution and additive semantics in R as R is an additive reward function. As such, Eq. (7) holds, with $P_i = P_j = P_k$ and $R_i = R_j = R_k$ within each partition $\mathfrak{I}_k$. When multiplying identical functions per partition, symmetric behaviour as formalised in Eq. (6) emerges, since the same probabilities from P_k (rewards from R_k) are multiplied (added) when different permutations of a partition action (observation) are handled. □

Model Complexity Given the more compact representation of the functions in a POSG, we get a drastic model space reduction:

Corollary 3. *The model complexity of an isomorphic POSG $\bar{M}$ is no longer dependent on the number of agents N.*

Proof. The sizes of the transition function T and reward functions $\boldsymbol{R}$, which follow from their input arguments, dominate the model complexity. Formally, the sizes $\mathbb{T}$ and $\mathbb{R}$ of the functions $T(S', S, \boldsymbol{A}, \boldsymbol{O})$ and $\{R_i(S, \boldsymbol{A})\}_{i \in \boldsymbol{I}}$, respectively, of a POSG M of Def. 1 equivalent to an isomorphic POSG $\bar{M}$ of Def. 2 lie in $\mathbb{T} \in O(s^2 a^N o^N)$ and $\mathbb{R} \in O(Nsa^N)$, with $a = \max_i |ran(A_i)|$ and $o = \max_i |ran(O_i)|$, $i \in \boldsymbol{I}$, which has N as an exponent. The sizes $\bar{\mathbb{T}}$ and $\bar{\mathbb{R}}$ of the functions $\boldsymbol{T}$ and $\boldsymbol{R}$ of an isomorphic POSG $\bar{M}$ of Def. 2 lie in $\bar{\mathbb{T}} \in O(Ks^2 a^K o^K)$ and $\bar{\mathbb{R}} \in O(Ksa^K)$, which does not reference N anymore. □

This indistinguishability only translates into smaller functions than in a ground POSG of Def. 1 if $K \ll N$. Given a particular ground POSG, one could check for Eqs. (5) to (7) to hold and then compactly encode the POSG as an isomorphic one, which yields K and the respective factorisations of T and $\boldsymbol{R}$. However, in such a scenario, one still has to store the ground POSG, which requires exponential space. Then, one has to check Eqs. (5) and (7), which can be especially cumbersome for Eq. (7), and compute the factors referenced in Eq. (6). Thus, the modelling scenario would be to directly formulate an isomorphic POSG for the cases where one assumes Eqs. (5) to (7) to hold to bypass the effort. In such a case, one could even consider tasks such as how many agents are necessary per partition to fulfil certain conditions, where the overall number of agents might no longer be known.

Of course, a compact representation does not help much if a solution algorithm has to revert to the grounded representation. Thus, next, we present a lifted version of the multi-agent dynamic programming operator [7] that works with isomorphic POSGs.

4 Lifting Dynamic Programming for Isomorphic POSGs

Multi-agent dynamic programming solves a finite-horizon POSG by iteratively eliminating weakly-dominated strategies [7] using a dynamic programming operator. For a lifted version that works with isomorphic POSGs, we have to adapt the operator to work with partitions. Given the results of the previous section, we prove in this section that all agents of the same partition use the same set of policies. Thus, it is sufficient to consider a representative agent for each partition and let the operator perform an exhaustive backup, compute corresponding value vectors, and then prune policies once per partition.

4.1 Weakly Dominated Policies among Indistinguishable Agents

Before setting up the lifted dynamic programming operator, we argue why we can work with representatives for each partition of indistinguishable agents.

Lemma 2. *Indistinguishable agents share the same set of available policies as well as the same belief and state values.*

Algorithm 2 Lifted Multi-Agent Dynamic Programming Operator

```
function LIFT-MA-DP(Set of sets of policies with corresponding value vectors $\{(\mathit{\Pi}_k^{t-1}, \boldsymbol{V}_k^{t-1})\}_{k=1}^K$)
    $\mathit{\Pi}_k^t \leftarrow$ Perform exhaustive backup using $\mathit{\Pi}_k^{t-1}$ for each partition $\mathfrak{I}_k \in \bar{\boldsymbol{I}}$
    $\boldsymbol{V}_k^t \leftarrow$ Calculate new value vectors for each partition $\mathfrak{I}_k \in \bar{\boldsymbol{I}}$
    while $\exists \pi_{k,l}^t \in \mathit{\Pi}_k^t$ : Eq. (9) holds do
        $\mathit{\Pi}_k^t \leftarrow \mathit{\Pi}_k^t \setminus \{\pi_{k,l}^t\}$, $\boldsymbol{V}_k^t \leftarrow \boldsymbol{V}_k^t \setminus \{v_{k,l}^t\}$
    return $\{(\mathit{\Pi}_k^t, \boldsymbol{V}_k^t)\}_{k=1}^K$
```

Proof. With agents i, j of a partition $\mathfrak{I}_k$ sharing the same actions and observations, the same reward function R_k, and the same transition function T_k due to conditional independences among them, the same possible policies are available to agents of the same partition by construction. Additionally, the values V_i, V_j of a belief $b_i = b_j$ are identical, i.e., $V_i = V_j = V_k$ given a policy $\pi_i = \pi_j$, a state s, and the other agents' policies $\boldsymbol{\pi}_{-i} = \boldsymbol{\pi}_j$, which are identical due to $\pi_i = \pi_j$. The same holds for the values V'_i, V'_j of $s, \pi_i = \pi_j, \boldsymbol{\pi}_i = \boldsymbol{\pi}_j$, which can be computed with the factorised versions of T and R, meaning $V'_i = V'_j = V'_k$. □

Theorem 2. *Policies that are weakly dominated for agent i are also weakly dominated for an indistinguishable agent j.*

Proof. As a direct consequence of Lemma 2, the same policies can be pruned if Eq. (1) holds. □

The effect of Theorem 2 is that we only need to back up policies once for each partition using a representative agent and then perform a pruning on those policies. Formally, the condition for pruning a partition policy $\pi_{k,l}^t$ becomes:

$$\forall s \in ran(S), \boldsymbol{\pi}_{-k}^t \in \boldsymbol{\mathit{\Pi}}_{-k}^t \exists \pi_{k,l}^t \in \mathit{\Pi}_k^t : V_k(s, \pi_{k,l}^t, \boldsymbol{\pi}_{-k}^t) \leq V_k(s, \pi_{k,m}^t, \boldsymbol{\pi}_{-k}^t), \quad (9)$$

The linear programme can be adapted accordingly (and is left out due to space restrictions). Next, we present the dynamic programming operator.

4.2 Lifted Dynamic Programming Operator

Given the results of the previous sections, we set up the lifted dynamic programming operator that operates on representatives for the partitions in an isomorphic POSG.

Algorithm 2 shows the lifted version of the operator, which works exactly as before, just on a partition level. So, it takes a set of policies $\mathit{\Pi}_k^{t-1} = \{\pi_{k,l}^{t-1}\}_{l=1}^m$ of depth $t-1$ for partition $\mathfrak{I}_k$. For each policy, there exists a value vector $v_{k,l}^{t-1}$ that denotes the value of that policy in each possible combination of state $s \in ran(S)$ and policies of the other partitions $\boldsymbol{\pi}_{-k} \in \boldsymbol{\mathit{\Pi}}_{-k}^{t-1}$. The operator then proceeds as before with first performing an exhaustive backup and second pruning weakly dominated policies per partition.

We briefly argue for correctness of Alg. 2 based on the correctness of Alg. 1:

Algorithm 3 Solving isomorphic POSGs

```
function LIFTED-SOLVE(POSG $\bar{M}$, horizon $\tau$)
    $\mathit{\Pi}_k^0 \leftarrow \emptyset$ for each partition $\mathfrak{I}_k \in \bar{\boldsymbol{I}}$
    $\boldsymbol{V}_k^0 \leftarrow \emptyset$ for each partition $\mathfrak{I}_k \in \bar{\boldsymbol{I}}$
    while $t \in \{1, \dots, \tau\}$ do
        $\{(\mathit{\Pi}_k^t, \boldsymbol{V}_k^t)\}_{k=1}^K \leftarrow$ LIFT-MA-DP($\{(\mathit{\Pi}_k^{t-1}, \boldsymbol{V}_k^{t-1})\}_{k=1}^K$)
    return $\{(\mathit{\Pi}_k^\tau, \boldsymbol{V}_k^\tau)\}_{k=1}^K$
```

Theorem 3. *Using Alg. 2 on an isomorphic POSG is equivalent to using Alg. 1 on a POSG, in which Eqs.* (5) *to* (7) *hold.*

Proof. Given the equivalence between an isomorphic POSG and a POSG, in which Eqs. (5) to (7) hold, and Theorem 2, we conclude that Alg. 2 only prunes policies that Alg. 1 would prune.

Algorithm 3 shows how to solve an isomorphic POSG using Alg. 2, which repeatedly calls Alg. 2 for increasingly deeper policy trees until the horizon is reached. The resulting set of policies can then be used to select the best policies given an initial state distribution.

5 Discussion

Before we consider related work, we briefly consider the nanoscale medical system as an isomorphic POSG and discuss the expressivity of isomorphic POSGs, which touches upon the assumptions made and limitations imposed.

5.1 Nanoscale Medocal Systems as Isomorphic POSGs

When simulating a nanoscale medical system based on [2], its components are divided into two categories of agents, nanosensors and nanobots, based on how they respond to particular markers and so-called messages in a blood stream. Together, these agents build the set $\boldsymbol{I}$ of agents, which are divided into K partitions, one partition for each marker / message that should be detected in the blood stream. According to early research in such a nanoscale medical system [2], each partition may contain about 64,000 agents, which means that the agent set must be at least $K \cdot 64,000$, with K being single-digit. Each agent has the ability to observe its marker / message (or not) and to output a marker of its own (or not), yielding two possible observations and actions for each agent. If describing the environment through the presence and absence of such markers and messages, there are 2^K possible states. Considering concrete numbers, e.g., with four markers and one message type, the agent set is of size $N = 5 \cdot 64,000 = 320,000$ partitioned into $K = 5$ partitions and the state space is of size $s = 2^5 = 32$, which leads to model sizes of the following in the ground case:

$$\mathbb{T} \in \mathcal{O}(32^2 \cdot 2^{320,000} \cdot 2^{320,000}), \tag{10}$$

$$\mathbb{R} \in \mathcal{O}(320,000 \cdot 32 \cdot 2^{320,000}) \tag{11}$$

as well as model sizes of the following in the isomorphic case:

$$\bar{\mathbb{T}} \in \mathcal{O}(5 \cdot 32^2 \cdot 2^5 \cdot 2^5), \tag{12}$$

$$\bar{\mathbb{R}} \in \mathcal{O}(5 \cdot 32 \cdot 2^5). \tag{13}$$

Expressivity Isomorphic POSG use the fact that we have some form of structure in the agent set for efficiency gains. Having a small number of different types of agents with a set of predetermined actions and observations available to them is a common occurrence and encoded in Eq. (5). However, the prerequisites regarding symmetric behaviour in Eq. (6) and conditional independence in Eq. (7) do pose some strong assumptions on the transition and reward function T and R in a trade-off with a drastic model complexity reduction and a much more efficient solution approach that is no longer exponentially dependent on N but only K, with $K \ll N$. However, isomorphic POSGs can encode more complex settings than a set of K independent POMDPs as the interaction between the different partitions is still a part of T and R.

Regarding the logical variables used in the proof for Theorem 1, this work introduces constructs of first-order logic into the agent set of a POSG, which allows for using this information explicitly for efficiency gains, which is not possible if using a standard POSG, which has this information only implicitly available.

Related Work We look at related work for POSGs as well as lifting in the context of MDP-based formalisms.

Next to dynamic programming for POSGs [7], there exist approximate pruning methods as an extension [11]. To work on scalability, research has focused on approximations for specific variants of POSGs, such as zero-sum games (e.g., [25,8,22]) and one-sided POSGs (e.g., [10,9,4]), and POSGs with common payoffs [5]. On a more practical level, partially observable game-theoretic Golog extends Golog with game-theoretic multi-agent planning in POSGs [6].

In offline decision making using MDP-based frameworks, lifting has been used in calculations to exploit relational structures in the state space of (PO)MDPs: In first-order MDPs (FOMPDs) [1], the situation calculus [14] is used to describe the state space. Factorised FOMDPs additionally assume a factorised representation of the state space [19]. Although not lifting-adjacent, object-oriented POMDPs similarly factorise the state space based on objects in the state space [23]. In FO-POMDPs, lifting is applied to policies, pruning policies that are indistinguishable [20]. In open-universe FO-POMDPs [21], the open-universe assumption that often comes with first-order representations is added using Bayesian logic as a basis. Most recently, lifting has been applied to the agent set in DecPOMDPs [3]. On a propositional level, probabilistic inference techniques have been used for multi-agent decision making in DecPOMDPs, factorising the state space using dynamic Bayesian networks, a temporal PGM [12].

To the best of our knowledge, we are the first to apply techniques from lifted probabilistic inference to the general formalism of POSGs, providing groundwork for new solution methods and avenues for approximations.

6 Conclusion

This paper presents a new compact encoding of POSGs, called isomorphic POSGs, which uses a partitioning of the agent set based on indistinguishable agents and conditional independences among indistinguishable agents, which allows for a model complexity that is no longer dependent (exponential or otherwise) on the number of agents, but only dependent on the number of partitions. A lifted multi-agent dynamic programming operator works on isomorphic POSGs, using the compact encoding for efficiency gains, computing and evaluating policies for representatives of each partition, avoiding doing repetitive work for each agent in a partition. As such, isomorphic POSGs have great potential for applications that have a huge number of agents such as nanoscale medical systems.

The next step in future work lies in a case study on nanoscale medical systems, combining the research and work of nanotechnology, medicine, and artificial intelligence through simulation runs. Future work also includes working on loosening the assumptions made for isomorphic POSGs, thereby increasing the expressivity of such POSGs, inspired by the lifting tool of counting (in contrast to isomorphism), as well as additionally using lifting for the state space.

References

1. Boutilier, C., Reiter, R., Price, B.: Symbolic Dynamic Programming for First-order MDPs. In: IJCAI-01 Proc. of the 17th International Joint Conference on Artificial Intelligence. pp. 690–697. IJCAI Organization (2001)
2. Braun, T., Fischer, S., Lau, F., Möller, R.: Lifting DecPOMDPs for Nanoscale Systems — A Work in Progress. In: 10th International Workshop on Statistical Relational AI at the 1st International Joint Conference on Learning and Reasoning (2021), https://arxiv.org/abs/2001.02021
3. Braun, T., Gehrke, M., Lau, F., Möller, R.: Lifting in Multi-agent Systems under Uncertainty. In: UAI-22 Proc. of the 38th Conference on Uncertainty in Artificial Intelligence. pp. 1–8. AUAI Press (2022)
4. Carr, S., Jansen, N., Bharadwaj1, S., Spaan, M.T.J., Topcu, U.: Safe policies for factored partially observable stochastic games. In: RSS-21 Proc. of Robotics: Science and Systems XVII. pp. 1–11. RSS Foundation (2021)
5. Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: Approximate solutions for partially observable stochastic games with common payoffs. In: AAAMAS-04 Proc. of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 136–143. IEEE (2004)
6. Finzi, A., Lukasiewicz, T.: Partially Observable Game-Theoretic Agent Programming in Golog. International Journal of Approximate Reasoning **119**, 220–241 (2020)
7. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: AAAI-04 Proc. of the 19th National Conference on Artificial Intelligence. vol. 4, pp. 709–715 (2004)
8. Horák, K., Bošanskỳ, B.: Solving partially observable stochastic games with public observations. In: Proc. of the AAAI conference on Artificial Intelligence. pp. 547–552. AAAI Press (2019)

9. Horák, K., Bošanskỳ, B., Kiekintveld, C., Kamhoua, C.: Compact representation of value function in partially observable stochastic games. In: IJCAI-19 Proc. of the 28th International Joint Conference on Artificial Intelligence. pp. 350–356. IJCAI Organisation (2019)
10. Horák, K., Bošanskỳ, B., Pěchouček, M.: Heuristic search value iteration for one-sided partially observable stochastic games. In: AAAI-17 Proc. of the 31st AAAI Conference on Artificial Intelligence. pp. 558–564 (2017)
11. Kumar, A., Zilberstein, S.: Dynamic programming approximations for partially observable stochastic games. In: FLAIRS-09 Proc. of the 22nd International Florida Artificial Intelligence Research Society Conference. AAAI Press (2009)
12. Kumar, A., Zilberstein, S., Toussaint, M.: Probabilistic Inference Techniques for Scalable Multiagent Decision Making. Journal of Artificial Intelligence Research **53**, 223–270 (2015)
13. Lau, F., Wendt, R., Fischer, S.: Dna-based molecular communication as a paradigm for multi-parameter detection of diseases. In: ACM NanoCom-17 Proc. of the 4th ACM International Conference on Nanoscale Computing and Communication 2017. ACM (2021)
14. McCarthy, J.: Situations, Actions, and Causal Laws. Tech. rep., Standford University (1963)
15. Niepert, M., Van den Broeck, G.: Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In: AAAI-14 Proc. of the 28th AAAI Conference on Artificial Intelligence. pp. 2467–2475. AAAI Press (2014)
16. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralised POMDPs. Springer (2016)
17. Poole, D.: First-order Probabilistic Inference. In: IJCAI-03 Proc. of the 18th International Joint Conference on Artificial Intelligence. pp. 985–991. IJCAI Organization (2003)
18. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson (2021)
19. Sanner, S., Boutilier, C.: Approximate Solution Techniques for Factored First-order MDPs. In: ICAPS-07 Proc. of the 17th International Conference on Automated Planning and Scheduling. pp. 288–295. AAAI Press (2007)
20. Sanner, S., Kersting, K.: Symbolic Dynamic Programming for First-order POMDPs. In: AAAI-10 Proc. of the 24th AAAI Conference on Artificial Intelligence. pp. 1140–1146. AAAI Press (2010)
21. Srivastava, S., Russell, S., Ruan, P., Cheng, X.: First-order Open-universe POMDPs. In: UAI-14 Proc. of the 30th Conference on Uncertainty in Artificial Intelligence. pp. 742–751. AUAI Press (2014)
22. Tomášek, P., Horák, K., Aradhye, A., Bošanskỳ, B., Chatterjee, K.: Solving partially observable stochastic shortest-path games. In: IJCAI-21 Proc. of the 30th International Joint Conference on Artificial Intelligence. pp. 4182–4189. IJCAI Organisation (2021)
23. Wandzel, A., Oh, Y., Fishman, M., Kumar, N., Wong, L.L., Tellex, S.: Multi-Object Search using Object-Oriented POMDPs. In: ICRA-19 Proc. of the 2019 International Conference on Robotics and Automation. pp. 7194–7200. IEEE (2019)
24. Wemmenhove, B., Mooij, J.M., Wiegerinck, W., Leisink, M., Kappen, H.J., Neijt, J.P.: Inference in the Promedas Medical Expert System. In: Conference on Artificial Intelligence in Medicine in Europe. pp. 456–460. Springer (2007)
25. Wiggers, A.J., Oliehoek, F.A., Roijers, D.M.: Structure in the value function of two-player zero-sum games of incomplete information. In: ECAI-16 Proc. of the 22nd European Conference on Artificial Intelligence. pp. 1628–1629. IOS Press (2016)