Lifting Multi-agent A* -
Implementing a New Optimisation Problem in Isomorphic DecPOMDPs:
How Many Agents Do We Need?

Master's thesis
in partial fulfilment of the requirements for the degree of
Master of Science

University of Münster
Computer Science Department

Supervisor:
JProf. Dr. Tanya Braun

First assessor:
JProf. Dr. Tanya Braun

Second assessor:
Prof. Dr. Benjamin Risse

Submitted by:
Constantin Castan

Münster, November 2022

Lifting Multi-agent A* - Implementing a New Optimisation Problem in Isomorphic DecPOMDPs: How Many Agents Do We Need?

Liften von Multi-agent A* -
Umsetzung eines neuen Optimierungsproblem in isomorphen DecPOMDPs:
Wie viele Agenten brauchen wir?

Abstract

Decentralised Partially Observable Markov Decision Process (DecPOMDP) models formalise sequential decision making problems under uncertainty in multi-agent systems. Solving a DecPOMDP means finding an optimal joint action policy for the agents, which yields the highest expected sum of rewards (utility) when executed in a stochastic environment. This problem scales exponentially with particularly the number of agents, calling for additional measures to achieve scalability beyond very small problems. Isomorphic DecPOMDP use lifting techniques on the agent set, leading to tractability regarding the number of agents by using representatives for groups of computationally indistinguishable agents. An optimal policy can then be calculated on the representative agents, using exact solution methods like Multi-agent A* (MAA*), and subsequently extending this policy to any number of agents. This yields a novel type of query on Isomorphic DecPOMDP, asking for the optimal number of agents required to obtain a desired utility value. This thesis contributes a formalisation of the optimisation problem underlying this query, a discussion of several solution approaches, and an implementation within a preexisting software framework for decision-theoretic planning tasks in multi-agent systems.

Contents

1 Introduction

Action planning is one of the most fundamental subfields of artificial intelligence. Ultimately, designing artificially intelligent agents comes down to the question of how to make them reach their specified goals, given the circumstances of the world the agents exist in, and their own capabilities to influence this world via some set of actions. Whenever there is more than one way to reach such a goal, it is especially interesting to find one that fulfills some notion of optimality. Formally defining models of the agent worlds and interactions provides a way of algorithmically solving action planning problems. In particular, sequential decision making is concerned with finding optimal sequences of actions over some planning horizon, as opposed to episodic (one-time) decision making. Naturally, the more realistic a model is, the harder (i.e., computationally complex) it becomes to solve a problem instance of it.

Decentralised Partially Observable Markov Decision Processes (DecPOMDPs) are one way of modelling such problems, accounting for multiple agents in a stochastic world which is not fully observable to any of them, and working towards either individual or, as in our case, collaborating for joint rewards. Acting in a stochastic environment means the agents can neither be certain about outcomes of their actions nor the observations they receive via their sensors. Therefore, solutions to DecPOMDPs cannot be a fixed sequence of (joint) actions, but rather a policy, specifying actions depending on perceptions about the environment's current state.

Finding an optimal policy for a DecPOMDP model is a notoriously hard problem, as the number of possible policies is exponentially dependent on the number of agents, and even doubly exponentially on the planning horizon, preventing scalability beyond very small problem instances. Approximate solutions can remedy this issue, but may be infeasible for problems that necessarily require exact solutions, e.g., in nanoscale medical systems (Lau et al., 2019). Consequently, a considerable amount of effort went into various approaches to reduce the complexity in DecPOMDPs and subclasses, both in terms of compact model representations and solution strategies (Amato et al., 2013).

This thesis is particularly concerned with Isomorphic DecPOMDPs (Braun, Gehrke, et al., 2022), which aim at exploiting symmetries in the behaviour of agents. Agents that behave exactly the same are indistinguishable from a computational point of view, which opens up the opportunity to save on redundant computational effort by only considering representatives for a larger group of indistinguishable agents. In probabilistic inference, this concept is referred to as *lifting* (Poole, 2003), a term that illustrates a move from zeroth up to first order representations. This idea is more commonly known, e.g., from zeroth-order propositional logic and first-order predicate logic. Performing operations on representatives instead of once per individual can lead to significant complexity improvements, reducing the exponential dependency on the agent number (in this particular case) to polynomial (Niepert and Van den Broeck, 2014).

Lifting the agent set in Isomorphic DecPOMDPs enables the definition of a novel query type, asking for the optimal number of agents required to obtain a desired utility, which refers to the rewards that are expected (uncertain) to be obtained by executing a policy. Finding this policy crucially requires an exact solution on the set of representative agents, which is accomplished by the Multi-agent A* algorithm (MAA*, Szer et al., 2005), an adaption of the classic A* algorithm to multiagent decision problems. This thesis describes the optimisation problem underlying this new query type, discusses several approaches to its solution, and presents an implementation based on the MADP Toolbox, a preexisting software framework for decision-theoretic planning tasks in multi-agent systems.

The thesis is structured as follows: Chapter 2 introduces the necessary theoretical foundations surrounding DecPOMDPs. Chapter 3 surveys several works related to lifting and complexity reduction in DecPOMDPs and other related multi-agent models. Chapter 4 explains the concepts behind Isomorphic DecPOMDPs in detail, formalises the aforementioned optimisation problem and different solutions, and presents the implementation, followed by a discussion in Chapter 5. Chapter 6 concludes the thesis.

2 Background and Notation

This chapter introduces the theoretical foundation and concepts that are used throughout this thesis. The first section informally motivates the field of sequential decision making under uncertainty with specific regard to Markov Decision Processes (MDPs), and the notions of partial observability and decentralisation, followed by a formal definition of DecPOMDPs, their computational complexity, and the basic idea of the MAA* algorithm.
For a more thorough introduction to DecPOMDPs, please refer to Oliehoek and Amato (2016).

2.1 Sequential Decision Making under Uncertainty

Consider the example of a robot moving through a grid world. In this grid there are predefined locations yielding some numerical reward or penalty for the robot, e.g., some form of goals or pitfalls. The environment is stochastic, which means that movements are not guaranteed to be in the desired direction, but instead are subject to some probability distribution over a set of different possible outcomes, e.g. due to technical malfunctions. The problem of finding an optimal sequence of movements while accounting for this uncertainty, in particular the risk of accidentally falling into a pit, can be modelled as an MDP.
In general, MDPs model *states* of some *agent* or system over a period of time, and stochastic, Markovian *transitions* between these states for each time step. Aside from the previous state, these transitions also depend on an *action* (or a decision on an action, hence the name). The desired behaviour is modeled by a *reward* function, assigning a numerical, additive value to each state. In the example above, the current system state is the robot's position in the grid, and the possible actions to decide on are movements to an adjacent grid cell.
A deterministic[1] or pure *policy* is an assignment of an action to each state, defining the entire behaviour of the agent. Due to the aforementioned uncertainty, however, this behaviour does not necessarily always result in the same sequence of actions taken, since at any point an action can result in an unintended outcome, diverging from the sequence of most likely outcomes. The impact of such a deviation can have wide ranging consequences, depending on the current state. The robot from the example may fall into an adjacent pit in the worst case, or just end up on a slightly less optimal route to its goal. A reasonable valuation of a policy, referred to as the policie's *utility*, is therefore defined as the expected sum of rewards obtained by behaving according to the policy. The solution to an MDP is the *optimal policy*, i.e., the one with the highest utility, which is referred to as *Maximum Expected Utility (MEU).*

[1]Policies can also be stochastic, these will however not be considered here.

Modelling a problem as an MDP makes some tacit simplifying assumptions. Most notably, the current state is assumed to be fully observable, while in reality, there usually is no access to such a ground truth. Referring to the robot example once more, if the robot is not equipped with a GPS module, it would have no external information about its position within the grid. It would therefore be required to make *observations* of its surroundings and deduce a probability distribution over all possible positions[2]. It may, for example, be equipped with a distance meter that can detect adjacent walls, which would however not unambiguously inform the current position, and may itself also be subject to technical imprecision or even malfunction. This uncertainty about the current state is called *partial observability*, and MDPs with this property are referred to as Partially Observable MDPs (POMDPs). Both MDPs and POMDPs can be solved iteratively by algorithms such as Value Iteration.

Another assumption of MDPs is the single-agent setting. There is only one entity making decisions and influencing the state through actions at every time step. Extending an MDP to allow for multiple agents to be active at the same time is called *decentralisation*, and such a decentralised MDP (DecMDP) can also be subject to partial observability, together forming a DecPOMDP.
Building upon the concepts presented here, the next section formally defines DecPOMDPs.

2.2 DecPOMDPs

Bernstein et al. (2002) originally presented DecPOMDPs. The majority of the following definitions is adopted from the work of Braun, Gehrke, et al. (2022), which in turn is based on Oliehoek and Amato (2016), and Russell and Norvig (2020). There exist several definitions with slight deviations, especially with regard to notation. This particular definition uses discrete random variables V, denoted as uppercase letters, with the finite set of values v_i it can take on referred to as its range $ran(V) = \{v_1, \ldots, v_m\}$.

Definition 2.2.1 (DecPOMDP)
A DecPOMDP model M is defined as a Tuple $(\boldsymbol{I}, S, \boldsymbol{A}, T, R, \boldsymbol{O}, \Omega, b_0)$:

- $\boldsymbol{I}$: a set of N agents
- S: a state random variable with a set of states as range
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{I}}$: a set of action random variables, each with a set of individual actions as range, and $ran(\boldsymbol{A}) = \times_{i \in \boldsymbol{I}} ran(A_i)$ the set of *joint actions*
- $T(S', S, \boldsymbol{A}) = P(S'|S, \boldsymbol{A})$: a state transition probability function
- $R(S, \boldsymbol{A})$: a real-valued reward function

[2]This example closely resembles the SLAM problem, see, e.g., Durrant-Whyte and Bailey (2006)

- $\boldsymbol{O} = \{O_i\}_{i \in \boldsymbol{I}}$: a set of observation random variables, each with a set of individual observations as range, and $ran(\boldsymbol{O}) = \times_{i \in \boldsymbol{I}} ran(O_i)$ the set of *joint observations*
- $\Omega(\boldsymbol{O}, S) = P(\boldsymbol{O}|S)$: observation probability function, also referred to as sensor function
- b_0: an initial state probability distribution ('belief') at time $t = 0$

Optional components:

- Discount factor $\gamma \in [0, 1]$
- Finite time horizon $\tau \in \mathbb{N}$

To avoid confusion, note that among the several different notations in literature, most notably, O refers to the sensor function in some cases. The original definition also includes an explicit starting state s_0, which is a special case of b_0, where $b_0(s_0) = 1$ and 0 for all other states.
Regarding the optional components: the discount factor γ can be used to increasingly diminish rewards based on the timesteps that have already passed to encourage agents to quickly reach their goals. If this is not desired, the default value of $\gamma = 1$ will not influence the rewards (see equation 2.4 below). While the definition allows for an infinite time horizon τ, this thesis and most literature only considers the finite case.

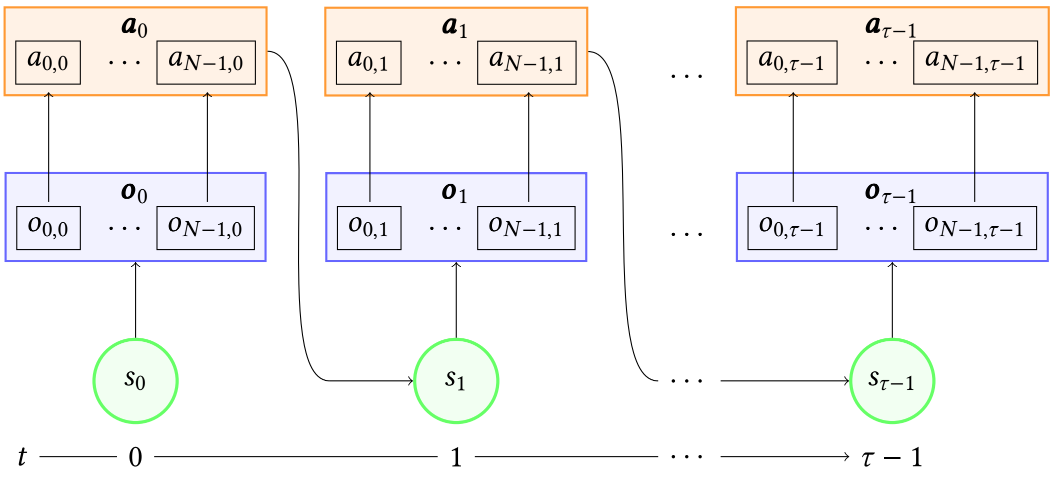

Figure 2.1: Core mechanism of the DecPOMDP state progression (adapted from Oliehoek and Amato, 2016)

Figure 2.1 illustrates the state progression in a DecPOMDP over time. The current state s_t can be viewed as emitting joint observations $\boldsymbol{o}_t$ at each timestep t, which is another way of interpreting observations by the agents. Each agent i observes its individual component $o_{i,t}$ of the joint observation $\boldsymbol{o}_t$, which then informs the decision on this agent's individual action $a_{i,t}$. The joint action $\boldsymbol{a}_0$ is then executed which determines the probability distribution over the next state s_{t+1}. This process runs from $t = 0$ through $t = \tau - 1$, assuming a finite horizon τ. Rewards are omitted here for visual clarity, since they do not influence the underlying mechanism. Agents are unable to observe the obtained rewards

during execution, as that may serve as an oracle-like source of information about the actual current state, however all information gain is supposed to be modeled by the sensor function.
The reward function is defined over joint actions, which implies working together for a shared reward. The more general partially observable stochastic game (POSG), where each agent has an individual reward function, can be defined to enable adversarial behaviour, but will not be considered here.
DecPOMDP models do not include explicit communication between agents, information sharing can however be modeled implicitly using actions and respective observations, e.g., through visual displays or speaker outputs. Including communication explicitly results in a DecPOMDP subclass called DecPOMDP-Com (Goldman and Zilberstein, 2003).

Due to partial observability, policies for a DecPOMDP cannot assign actions to states, since the agents can only guess the actual current state. Therefore, the agents' individual policies $\pi_i : ran(O_{i,0:t}) \longrightarrow ran(A_i)$ each map from their respective observation histories $o_{i,0:t}$ to actions a_i for each time step $t \leq \tau - 1$. The subscript $[\cdot]_{s:e}$ denotes a sequence over a discrete time interval $[s : e]$, in this case the sequence $(o_{i,0}, o_{i,1}, \ldots, o_{i,t})$. All individual policies taken together form a *joint policy* $\boldsymbol{\pi} = (\pi_i)_{i \in I}$. The set of all possible joint policies of a DecPOMDP model M is denoted $\boldsymbol{\Pi}_M$.

As before, the optimality criterion is that of Maximum Expected Utility (MEU), where the solution to the DecPOMDP is the optimal policy $\boldsymbol{\pi}^*$:

$$MEU(M) = (\boldsymbol{\pi}^*, U_M(\boldsymbol{\pi}^*)) \tag{2.1}$$

$$\boldsymbol{\pi}^* = \operatorname*{argmax}_{\boldsymbol{\pi} \in \boldsymbol{\Pi}_M} U_M(\boldsymbol{\pi}) \tag{2.2}$$

The expected utility $U_M(\boldsymbol{\pi}) \mathrel{\hat{=}} E[\sum_{t=0}^{\tau-1} R(s_t, \boldsymbol{a}_t) | b_0, \boldsymbol{\pi}]$ can be calculated by summing the utilities of each state as starting state, and weighting these utilities by the probability of this state actually being the starting state:

$$U_M(\boldsymbol{\pi}) = \sum_{s_0 \in ran(S)} b_0(s_0) \cdot U_M^{\boldsymbol{\pi}}(s_0, \boldsymbol{o}_{0:0}) \tag{2.3}$$

Note that $\boldsymbol{o}_{0:0} = \emptyset$. The utility of a particular state is then calculated recursively for each time step up to the horizon τ:

$$\begin{aligned} U_M^{\boldsymbol{\pi}}(s_t, \boldsymbol{o}_{0:t}) = R(s_t, \boldsymbol{\pi}(\boldsymbol{o}_{0:t})) + \gamma^t \cdot \sum_{s_{t+1} \in ran(S)} \Big[\, T(s_{t+1}, s_t, \boldsymbol{\pi}(\boldsymbol{o}_{0:t})) \\ \cdot \sum_{\boldsymbol{o}_{t+1} \in ran(\boldsymbol{O})} \Omega(\boldsymbol{o}_{t+1}, s_{t+1}) \; U_M^{\boldsymbol{\pi}}(s_{t+1}, \boldsymbol{o}_{0:t+1}) \Big] \end{aligned} \tag{2.4}$$

In essence, each time steps yields an immediate reward based on the assumed state and the joint action given by the joint policy. Future rewards are then added recursively on top by considering the immediate rewards that each next state would yield in combination with each possible next observation, and weighting both by their likelihood of occuring, which is given by T and Ω, respectively.

The joint observation histories are extended in each time step by concatenating the previous history with the new joint observation: $\boldsymbol{o}_{0:t+1} = \boldsymbol{o}_{0:t} \circ \boldsymbol{o}_{t+1}$. The calculation stops at timestep $t = \tau - 1$, where $U_M^{\boldsymbol{\pi}}(s_{\tau-1}, \boldsymbol{o}_{0:\tau-1}) = R(s_{\tau-1}, \boldsymbol{\pi}(\boldsymbol{o}_{0:\tau-1}))$ only yields the last reward without continuing the recursion.

2.3 Computational Complexity

The core challenge of DecPOMDPs lies in their exponential dependency both on the agent number N and, in some parts, the horizon τ as well. These dependencies emerge in three facets:
Firstly, the function sizes of T, R and O, denoted by $\mathbb{T}, \mathbb{R}$, and $\mathbb{O}$, respectively, all depend exponentially on N. Formally, they lie in

$$\mathbb{T} \in O\left(s^2 a^N\right) \qquad \mathbb{R} \in O\left(s a^N\right) \qquad \mathbb{O} \in O\left(s o^N\right) \tag{2.5}$$

where $s = |ran(S)|$ is the number of states, $a = \max_{i \in \boldsymbol{I}} |ran(A_i)|$ the largest set of actions available to some agent, and $o = \max_{i \in \boldsymbol{I}} |ran(O_i)|$ the largest set of observations available to some agent. Equation 2.5 follows directly from multiplication of the input ranges.
Secondly, the cost $\mathbb{C}$ of evaluating a single joint policy depends exponentially on both N and τ, and thirdly, the total number $\mathbb{P}$ of existing joint policies depends exponentially on N and even doubly exponentially on τ. Formally, they lie in (Oliehoek and Amato, 2016):

$$\mathbb{C} \in O\left(s o^{N\tau}\right) \qquad \mathbb{P} \in O\left(a^{\frac{N(o^\tau - 1)}{o-1}}\right) \tag{2.6}$$

To illustrate this complexity in practice, consider the Dec-Tiger example by Nair, Tambe, et al. (2003). In this problem, there are two agents who have to choose between opening one of two doors, with a treasure behind one and a hostile tiger behind the other. The agents may also choose to listen for noise made by the tiger to inform a belief about which door it is more likely to be behind. With $a = 3$ and $o = 2$, this is close to the least complex problem possible in terms of these problem primitives. Table 2.1 shows the sizes of the Dec-Tiger policy space for different agent numbers N and time horizons τ. While this problem is a two-agent problem by definition, the larger agent numbers are included for demonstration purposes.

	Horizon τ		
Agent no. N	2	4	6
2	729	2.059×10^{14}	1.310×10^{60}
3	19683	2.954×10^{21}	1.499×10^{90}
4	531441	4.239×10^{28}	1.716×10^{120}

Table 2.1: Sizes of the Dec-Tiger policy space $\mathbb{P}$ for different agent numbers N and time horizons τ

With four agents and a horizon of six steps, there are already $1.7x10^{120}$ different policies, even though there are only three actions and two observations in this problem. Numbers like these provide intuition about how hard DecPOMDPs are to solve, especially more realistically modelled problems with some of these primitives in the order of hundreds or thousands.
Formally, finding the optimal solution of a DecPOMDP with finite horizon is NEXP-complete for $N \geq 2$ agents (Bernstein et al., 2002). Unlike MDPs and POMDPs, there is no method like Value Iteration to directly calculate the optimal joint policy $\boldsymbol{\pi}^*$ for a DecPOMDP, since there is no way to know what other agents are doing. Solving a DecPOMDP therefore requires searching through the entire space $\mathbb{P}$ of joint policies for the optimal one. The naive solution of brute force search would involve the complete evaluation of every joint police in $\mathbb{P}$, yielding a total cost of $\mathbb{C} \cdot \mathbb{P}$.

The next section describes a more sophisticated approach to this search procedure.

2.4 Multi-agent A*

The MAA* search algorithm is one way of alleviating some of the aforementioned complexity by early detecting suboptimal solutions, speeding up the search process while still retaining the optimal solution.
The basic principle of MAA* is the same as classic A* search. The general idea of A* is to incrementally construct the optimal solution by expanding the most promising partial solution in each step, and pruning the search space of suboptimal partial solutions every time a solution candidate is expanded to full length. The most promising partial solution is determined by an evaluation function F that combines the exact value of a partial solution with a heuristic estimate for the remainder of the solution. The heuristic function has to be admissible to allow for search space pruning, which means that it has to overestimate the actual value if the goal is maximisation, or underestimate for minimisation. Given an admissible heuristic, pruning works as follows (in the maximisation case): once a solution has been expanded to full length, there is a lower bound for the best overall value attainable. If the evaluation $F(p)$ of a partial solution p is below this lower bound, the admissibility (overestimation) property of the heuristic function guarantees that this partial solution cannot be expanded into a full-length solution with a higher actual value than the current lower bound. Therefore, these partial solutions need not be expanded further and can be discarded from the search procedure.

MAA* applies this general A* principle to joint policies in DecPOMDPs. Each node at depth t in the MAA* search tree represents a partial joint policy (a joint policy of length $t < \tau$). Figure 2.2 shows an exemplary expansion of a horizon-2 partial joint policy node into one of its horizon-3 child nodes. δ^t denotes the vector of individual horizon-t policies, where q_i^t is the individual policy of agent i. In this minimal example there are only two agents, actions, and observations. Each leaf node within δ^2 (note the distinction between outer and inner nodes) gets expanded into two child nodes, one per

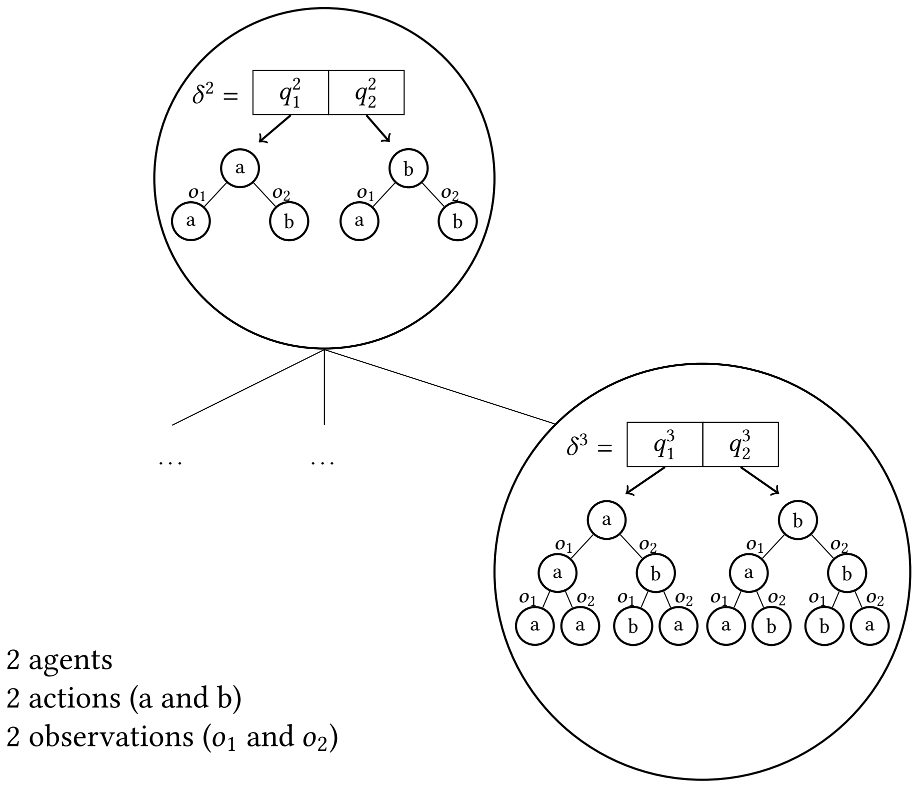
$\delta^2 = q_1^2 \; q_2^2$
$\delta^3 = q_1^3 \; q_2^3$
a
b
o_1
o_2
…
2 agents
2 actions (a and b)
2 observations (o_1 and o_2)

Figure 2.2: Expansion of a horizon-2 node into one of its horizon-3 child nodes in the MAA* search tree (adapted from Szer et al., 2005)

observation, yielding eight new leaf nodes in δ^3. Each distinct assignment of the two actions a and b to these eight nodes forms a new horizon-3 joint policy, of which there are $2^8 = 256$ possible assignments. Therefore, the (outer) horizon-2 partial joint policy node gets expanded into 256 horizon-3 nodes, further illustrating the exponential nature of the problem as well as the impact of early detecting suboptimal solutions.
Admissible heuristics for partial joint policies can be defined, for instance, by disregarding some properties of the DecPOMDP such as decentralisation and partial observability, thus using the evaluation of the underlying MDP or POMDP as so-called Q-value heuristic (Oliehoek, Spaan, et al., 2008). Szer et al. (2005) describe an entire class of admissible heuristics.

The complexity improvement of MAA* cannot be stated in general, as this depends on the problem and the quality of the heuristic. For the specific case of the Dec-Tiger problem with the underlying MDP as heuristic, the numbers of evaluated policies for horizon 2 and 4 are reduced from 729 and 2.059×10^{14} (cf. Table 2.1) to 252 and 9.445×10^8, respectively (Szer et al., 2005).

MAA* is guaranteed to find the optimal solution. There are also approximate solution methods like Joint Equilibrium-based Search for Policies (JESP, Nair, Tambe, et al., 2003) and Direct Cross-Entropy Policy Search (DICEPS, Oliehoek, Kooij, et al., 2008), which trade in guaranteed optimality for greater speedup potential. For the purpose of this thesis, however, an exact solution method is crucial.

3 Related Work

The concept of lifting has been applied in various fields, including probabilistic inference (Kersting et al., 2009; Nath and Domingos, 2010; Van den Broeck et al., 2011; Braun and Möller, 2018; Holtzen et al., 2019), online decision making (Nath and Domingos, 2009; Apsel and Brafman, 2011; Gehrke, Braun, Möller, et al., 2018; 2019) [both as cited by Braun, Gehrke, et al., 2022], and with particular relevance to this thesis, in offline decision making, with specific regard to (PO)MDPs. Several approaches target different scalability challenges concerning space and time complexity.
Some of the following works do not relate to first-order representation or lifting, however, they do relate to this thesis in a more general sense of complexity reduction and space compression in DecPOMDPs.

State space and actions. First-order MDPs (FOMDPs, Boutilier et al., 2001) are specified using a variant of the situation calculus (McCarthy, 1963), and are solved without explicitly enumerating the states and actions, which are instead represented by a set of first-order formulas. Factorised FOMDPs (Sanner and Boutilier, 2007) extend FOMDPs by a factorised representation of the state space to enable factored transition models.

Policy space. Sanner and Kersting (2010) use lifting for dominance analysis to prune suboptimal policies in First-order POMDPs (FOPOMDPs). First-order Open Universe POMDPs (OUPOMDPs) (Srivastava et al., 2014) additionally allow uncertainty over the existence and identity of objects in the first-order representation. Boularias and Chaib-draa (2008) use lossless compression on the policy space to increase efficiency in Dynamic Programming for DecPOMDPs.

Horizon. Memory-Bounded Dynamic Programming (MBDP, Seuken and Zilberstein, 2007b) is an approximate solution method for DecPOMDPs that surrenders the guarantee for the exact solution in exchange for linear dependence on the horizon τ instead of doubly exponential.

Observations. Improved MBDP (IMBDP) (Seuken and Zilberstein, 2007a) additionally tackles the exponential dependency on the number of observations by excluding unlikely or even impossible observations. MBDP with observation compression (MBDP-OC, Carlin and Zilberstein, 2008) merges sets of observations into categories, guided by the resulting value loss, and considers observations within a category as indistinguishable.

Agents. Braun, Gehrke, et al. (2022) claim to be the first to consider lifting for the agent set. The majority of this thesis is based on their work.

Independent choice logic (Poole, 1997) is an alternative semantic framework inspired by MDPs using first-order representations, which is also able to represent multiple agents.

4 Approach

This chapter describes a new optimisation problem that arises from the work of Braun, Gehrke, et al. (2022) about symmetric DecPOMDPs, in particular the isomorphic variant. These are introduced in the first section, followed by a formal definition of the new optimisation problem and query type, which asks for the minimum number of agents that is required to obtain a given expected utility in an Isomorphic DecPOMDP. Finally, the last section contains details about the implementation.

4.1 Symmetric DecPOMDPs

The fundamental observation which motivates symmetric DecPOMDPs is that models can contain formally indistinguishable agents, i.e., agents that can choose from the same range of actions, make the same observations, and behave the same. Two agents show identical behaviour if T, R, and Ω map to the same value when swapping the agents in the input, which is where the notion of symmetry originates from. To be clear, the same behaviour does not imply always doing the exact same thing, only given the same situation, i.e., it means adhering to the same policy. Such a situation may occur, for instance, when modelling problems with distinct agent types.
The next section formalises the partitioning of the agent set into subsets of indistinguishable agents, and adapts the original definition of DecPOMDPs to incorporate this partitioning, yielding Partitioned DecPOMDPs as a framework for symmetric DecPOMDPs.

4.1.1 Partitioned DecPOMDPs

The agent set $\boldsymbol{I}$ is partitioned into K pairwise disjoint, nonempty subsets $\mathfrak{I}_k$, formally:

$$\boldsymbol{I} = \dot{\bigcup}_{k=1}^{K} \mathfrak{I}_k$$
$$\forall k \in \{1, \dots, K\} : \mathfrak{I}_k \neq \emptyset$$
$$\forall k, l \in \{1, \dots, K\}, k \neq l : \mathfrak{I}_k \cap \mathfrak{I}_l = \emptyset$$

Additionally, agents within the same partition are required to have the same sets of individual actions and observations available:

$$\begin{aligned} \forall k \in \{1, \dots, K\} : \ \forall i, j \in \mathfrak{I}_k : \\ ran(A_i) = ran(A_j) \\ ran(O_i) = ran(O_j) \end{aligned} \tag{4.1}$$

The previous definition of DecPOMDPs is then adapted to incorporate this partitioning and *partition* actions and observations, satisfying Equation 4.1. In this context the overline notation sets function symbols and sets apart from non-partitioned DecPOMDPs.

Definition 4.1.1 (Partitioned DecPOMDP)
A partitioned DecPOMDP model $\overline{M}$ is defined as a Tuple $(\bar{\boldsymbol{I}}, \bar{S}, \bar{\boldsymbol{A}}, \bar{T}, \bar{R}, \bar{\boldsymbol{O}}, \bar{\Omega}, \bar{b}_0)$:

- $\bar{\boldsymbol{I}}$: a *partitioning* $\{\mathfrak{I}_k\}_{k=1}^K$ of N agents, where $n_k = |\mathfrak{I}_k|$ and $|\bar{\boldsymbol{I}}| = \sum_k n_k = N$
- $\bar{S}$: a state random variable with a set of states as range
- $\bar{\boldsymbol{A}} = \{\bar{A}_k\}_{k=1}^K$: a set of action random variables, each with a set of *partition actions* as range, and $ran(\bar{\boldsymbol{A}}) = \times_{k=1}^K ran(\bar{A}_k)$ the set of *joint actions*
- $\bar{T}(\bar{S}', \bar{S}, \bar{\boldsymbol{A}}) = P(\bar{S}'|\bar{S}, \bar{\boldsymbol{A}})$: a state transition probability function
- $\bar{R}(\bar{S}, \bar{\boldsymbol{A}})$: a real-valued reward function
- $\bar{\boldsymbol{O}} = \{\bar{O}_k\}_{k=1}^K$: a set of observation random variables, each with a set of *partition observations* as range, and $ran(\bar{\boldsymbol{O}}) = \times_{k=1}^K ran(\bar{O}_k)$ the set of *joint observations*
- $\bar{\Omega}(\bar{\boldsymbol{O}}, \bar{S}) = P(\bar{\boldsymbol{O}}|\bar{S})$: observation probability function, also referred to as sensor function
- $\bar{b}_0$: an initial state probability distribution ('belief') at time $t = 0$

Optional components:

- Discount factor $\gamma \in [0, 1]$:
- Finite time horizon $\tau \in \mathbb{N}$

In Partitioned DecPOMDPs, as opposed to each agent, each partition $\mathfrak{I}_k$ has its its own partition policy $\bar{\pi}_k : ran(\bar{O}_k) \longrightarrow ran(\bar{A}_k)$, mapping partition observation histories $\bar{o}_{k,0:t}$ to partition actions $\bar{a}_k$ for each time step $t \leq \tau - 1$. Joint partition policies $\bar{\boldsymbol{\pi}} = (\bar{\pi}_k)_{k=1}^K$ consist of one policy for each partition, with $\boldsymbol{\Pi}_{\overline{M}}$ the set of all joint partition policies.

Partitioned DecPOMDPs follow the same principles as standard DecPOMDPs. Replacing the elements of model M with their respective counterparts in $\overline{M}$ yields the same equations as 2.1 through 2.4 for Partitioned DecPOMDPs. In fact, if every partition contains exactly one agent, i.e. $N = K$, then the partitioned model $\overline{M}$ is equivalent to M (or 'ground'). For every other $\overline{M}$ with $K < N$, however, the partitioned model reduces the number of required action and observation random variables, which has direct implications on the computational complexity, especially if $K \ll N$:

$$\bar{\mathbb{T}} \in O(\bar{s}^2 \bar{a}^K) \qquad \bar{\mathbb{R}} \in O(\bar{s}\bar{a}^K) \qquad \bar{\mathbb{O}} \in O(\bar{s}\bar{o}^K) \tag{4.2}$$

$$\bar{\mathbb{C}} \in O\big(\bar{s}\bar{o}^{K\tau}\big) \qquad\qquad \bar{\mathbb{P}} \in O\Big(\bar{a}^{\frac{K(\bar{o}^{\tau}-1)}{\bar{o}-1}}\Big) \tag{4.3}$$

Analogously to Equations 2.5 and 2.6, $\bar{s} = |ran(\overline{S})|$, $\bar{a} = \max_k |ran(\overline{A}_k)|$, and $\bar{o} = \max_k |ran(\overline{O}_k)|$, with $k \in \{1, \ldots, K\}$. Assuming $\overline{M}$ is a partitioned version of the underlying model M, or at least models the same state space, then $\bar{s} = s$.

Partitioned DecPOMDPs serve as the framework for symmetric DecPOMDPs, but they do not themselves enforce agent indistinguishability within partitions. The model is defined over an abstract concept of partition actions and observations, without fully specifying what they consist of and how they interact with $\overline{T}, \overline{R}$, and $\overline{\Omega}$ to achieve the aforementioned symmetry. To enable lifting, this framework has to be instantiated with specific forms of symmetry and the according syntactic constructs. The next section specifies one of such instantiations, referred to as Isomorphic Symmetry.

4.1.2 Isomorphic Symmetry

Poole (2003) presents the idea of a Parameterised Random Variable (PRV) to represent a set of random variables. In the following, X_k is a logical variable with the domain $dom(X_k) = \mathfrak{I}_k$ being the set of agents in partition k, and the PRV $A_k(X_k)$ represents all $A_k(i_k)$ with $i_k \in dom(X_k)$, where $A_k(i_k)$ is simply the action random variable of the i-th agent in partition k. Analogously, the observation random variables are represented by the PRVs $O_k(X_k)$. Defining operations on such a first-order representation, where individual agents no longer appear explicitly, enables capturing symmetries in the transition, reward, and observation functions in a compact encoding, which will be explained in this section.

Several authors have built upon the idea of using PRVs and adapted the PRV syntax in the process. For a detailed definition see, e.g., Taghipour (2013), who uses PRVs of the form $P(\boldsymbol{X})|C$, where a constraint C specifies which value assignments to the set of logical variables $\boldsymbol{X}$ are allowed. This form translates to the form used here by setting $\boldsymbol{X}$ to X_k and C to $dom(X_k)$, yielding the form $A_k(X_k)|dom(X_k)$. Since C allows for the entire domain of X_k for each k, the C is omitted for simplicity.

A Partitioned DecPOMDP model $\overline{M}$ can now be instantiated with PRVs, i.e., the partition actions and observations $\overline{A}_k$ and $\overline{O}_k$ are now PRVs $A_k(X_k)$ and $O_k(X_k)$, respectively. The transistion, reward, and sensor function are adapted accordingly to be defined over these PRVs:

$$\begin{aligned}
&\overline{T}(\bar{S}', \bar{S}, A_1(X_1), \ldots, A_k(X_k)) \\
&\overline{R}(\bar{S}, A_1(X_1), \ldots, A_k(X_k)) \\
&\overline{\Omega}(O_1(X_1), \ldots, O_k(X_k), \bar{S})
\end{aligned}$$

The resulting model $\overline{M}_i$ is referred to as Isomorphic DecPOMDP.

At this point, the symmetries in these functions are not yet apparent. Rather than viewing these definitions as capturing symmetries that already exist, they instead induce the symmetries themselves. In other words, the definitions came first, and the implied symmetries now have to be inferred from them. The symmetries become evident by defining how these functions each imply a full joint distribution over all represented random variables, and how to resolve them, making each agent appear explicitly again. In lifted probabilistic inference, this process is referred to as *grounding*, as the inverse term to lifting.

Grounding the PRV versions of $\bar{T}, \bar{R}$, and $\bar{\Omega}$ is based on two assumptions, with the first being agent indistinguishability within each partition as mentioned previously, meaning every agent behaves exactly like each other agent of the same partition. This indistinguishability allows for a singular definition of a certain agent behaviour (in terms of these functions), which can then be applied to all agents at once. $\bar{T}, \bar{R}$, and $\bar{\Omega}$ can thus be viewed as "template" functions for agent behaviour, as they do not specify the number of agents they apply to, only the number of partitions, which can each contain any number of agents. Consequently, these functions imply not just a single full joint distributions, but one for each distinct assignment of agent numbers to the partitions.

The second assumption that is made in Isomorphic DecPOMDPs is intra-partition agent independence, i.e., every agent operates completely independent of each other agent in its respective partition. This strong assumption allows the implied full joint distributions to be decomposed into products, where the factors are each possible grounding $d \in D = \times_{k=1}^{K} dom(X_k)$ inserted into $\bar{T}, \bar{R}$, and $\bar{\Omega}$, respectively. The values of $\bar{T}$ and $\bar{\Omega}$ result from multiplication over all groundings $d = (x_{1,i_1}, \ldots, x_{K,i_K})$, while the additive rewards $\bar{R}$ result from summation:

$$\begin{aligned} T(S', S, A_1, \ldots, A_N) &= \prod_{d \in D} \bar{T}(\bar{S}', \bar{S}, A_1(x_{1,i_1}), \ldots, A_K(x_{K,i_K})) \\ R(S, A_1, \ldots, A_N) &= \sum_{d \in D} \bar{R}(\bar{S}, A_1(x_{1,i_1}), \ldots, A_K(x_{K,i_K})) \\ \Omega(O_1, \ldots, O_N, S) &= \prod_{d \in D} \bar{\Omega}(O_1(x_{1,i_1}), \ldots, O_K(x_{K,i_K}), \bar{S}) \end{aligned} \tag{4.4}$$

The tables 4.1 show an example of this grounding for $\bar{T}$ in a simple case with two partitions containing two agents each, two states $\{s^0, s^1\}$, and a shared range of actions $\{a^0, a^1\}$. Table 4.1 (a) shows the function $\bar{T}$, instantiated with PRVs $A_1(X_1)$ and $A_2(X_2)$, and Table 4.1 (b) shows the implied full joint distribution for all agents on the ground level. The example omits the $\bar{S}'$ for visual simplicity, which does not change the calculation in principle.
The example works the same for $\bar{R}$ (with mappings r_i): one would only need to change all exponents in the function values to factors, e.g., the first entry $R(s^0, a^0, a^0, a^0, a^0)$ in the full joint table would map to $4 \cdot r_1$ instead of p_1^4.

The exact calculation of the full joint values is exemplified below for the second entry in Table 4.1

$\bar{S}$	$A_1(X_1)$	$A_2(X_2)$	$\bar{T}$
s^0	a^0	a^0	p_1
s^0	a^0	a^1	p_2
s^0	a^1	a^0	p_3
s^0	a^1	a^1	p_4
s^1	a^0	a^0	p_5
s^1	a^0	a^1	p_6
s^1	a^1	a^0	p_7
s^1	a^1	a^1	p_8

(a) PRV version of $\bar{T}$

S	$A_1(x_{1,1})$	$A_1(x_{1,2})$	$A_2(x_{2,1})$	$A_2(x_{2,2})$	T
s^0	a^0	a^0	a^0	a^0	p_1^4
s^0	a^0	a^0	a^0	a^1	$p_1^2p_2^2$
s^0	a^0	a^0	a^1	a^0	$p_1^2p_2^2$
s^0	a^0	a^0	a^1	a^1	p_2^4
s^0	a^0	a^1	a^0	a^0	$p_1^2p_3^2$
s^0	a^0	a^1	a^0	a^1	$p_1p_2p_3p_4$
s^0	a^0	a^1	a^1	a^0	$p_1p_2p_3p_4$
s^0	a^0	a^1	a^1	a^1	$p_2^2p_4^2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s^1	a^1	a^1	a^1	a^1	p_8^4

(b) Full joint distribution

Table 4.1: Grounding of the PRV version of $\bar{T}$ into the implied full joint distribution for two partitions with two agents each, and a shared range of actions $\{a^0, a^1\}$. This example shows only one state $\bar{S}$ for simplicity.

(b):

$$\begin{aligned} T(s^0, a^0, a^0, a^0, a^1) &= \bar{T}(s^0, A_1(x_{1,1}), A_2(x_{2,1})) \cdot \bar{T}(s^0, A_1(x_{1,2}), A_2(x_{2,1})) \\ &\quad \cdot \bar{T}(s^0, A_1(x_{1,1}), A_2(x_{2,2})) \cdot \bar{T}(s^0, A_1(x_{1,2}), A_2(x_{2,1})) \\ &= p_1 \cdot p_1 \cdot p_2 \cdot p_2 \end{aligned}$$

Another way of looking at the PRV definitions of $\bar{T}$, $\bar{R}$, and $\bar{\Omega}$ is that they in fact compactly define multiple identical functions at once. Viewed in the context of probabilistic graphical models, these identical functions form isomorphic subgraphs of the larger function they compose, which is the origin of the term isomorphic symmetry.

4.1.3 Complexity Improvements

Braun, Gehrke, et al. (2022) show that in order to rank joint policies by utility in an Isomorphic DecPOMDP, it suffices to only consider one representative agent for each partition $\mathfrak{I}_k$ during the calculation. Formally, their proof shows that partition policies can be defined over $ran(A_k)$ and $ran(O_k)$. Once an optimal policy has been found, each agent adopts the individual policy of its partition representative. The resulting joint policy is then identical to the one that would have been found in the underlying DecPOMDP that does not consider isomorphic symmetries, given that an exact solution method like MAA* was used. Since the policy calculation only considers representatives but not partition sizes, the expected utility values do not equal the true values of the full policies, although they are ranked the same. To obtain the true MEU, the optimal joint policy therefore needs to be evaluated once with all agents considered.

Defining partition policies over $ran(A_k)$ and $ran(O_k)$ means that the values $\bar{a} = \max_k |ran(A_k)|$ and $\bar{o} = \max_k |ran(O_k)|$ with $k \in \{1, \ldots, K\}$ in Equations 4.2 and 4.3 coincide with a and o in Equations 2.5 and 2.6. Therefore, the function sizes $\overline{\mathbb{T}}_i$, $\overline{\mathbb{R}}_i$, $\overline{\mathbb{O}}_i$, and the size of the policy space $\overline{\mathbb{P}}_i$ also coincide with $\mathbb{T}$, $\mathbb{R}$, $\mathbb{O}$, and $\mathbb{P}$, respectively, but with K replacing N and $s = |ran(\bar{S})|$:

$$\overline{\mathbb{T}}_i \in O\big(s^2 a^K\big) \qquad \overline{\mathbb{R}}_i \in O\big(sa^K\big) \qquad \overline{\mathbb{O}}_i \in O\big(so^K\big) \tag{4.5}$$

$$\overline{\mathbb{P}}_i \in O\Big(a^{\frac{K(o^\tau - 1)}{o-1}}\Big) \tag{4.6}$$

The result for the evaluation cost $\overline{\mathbb{C}}_i$ of a policy, however, is not as straightforward. Braun, Gehrke, et al. state the complexity as

$$\overline{\mathbb{C}}_i \in O\big(log_2(n) so^{K\tau}\big)$$

with $n = \max_k n_k$. The partition sizes n_k are assumed to appear as exponents, and the logarithm emerges from the complexity of binary exponentiation, also known as Square-and-Multiply.

There need to be further concessions for this result to hold, which will be discussed in more detail in section 4.2.1.

4.2 New Optimisation Problem and Query Type

From the first part of the previous section follows a more general corollary: While different partition sizes do not influence agent behaviour, i.e., result in the same policy, they do influence the expected utility. The optimal policy $\bar{\boldsymbol{\pi}}^*$ that was found using representatives can be expanded to any number of agents in each partition, and each distinct agent distribution over the partitions yields its own expected utility. This insight gives rise to a new kind of query:

> Given an Isomorphic DecPOMDP model $\overline{M}_i$ and a desired utility U_{min}, how many agents are at least required in $\overline{M}_i$ to obtain an expected utility of U_{min}?

Technically, the agent number N, as part of the partitioning $\bar{\boldsymbol{I}}$, is a model parameter of $\overline{M}_i$, which means the query should in fact ask for a model $\overline{M}'_i$ which obtains U_{min} and is identical to $\overline{M}_i$ except for $\bar{\boldsymbol{I}}'$, which is a partitioning of the optimal agent number. This only has syntactical implications, however, and for the sake of simplicity, the agent number of a model $\overline{M}_i$ is assumed to be modifiable to a number $n \geq K$ for the purpose of utility calculation, denoted by $\overline{M}_i(n)$. Note that this leaves the partitioning open to the optimisation, which defines the optimal agent distribution itself, as the query only asks for the total agent number. The same principle applies to the representatives' policy, where

$\bar{\boldsymbol{\pi}}^*(n)$ denotes the extension of the policy $\bar{\boldsymbol{\pi}}^*$ to all n agents.
The optimisation problem underlying this query is thus defined as finding a number n^* of agents, s.t.

$$n^* = \min\{n \in \mathbb{N}^{\geq K} \mid LU_{\overline{M}_i(n)}(\bar{\boldsymbol{\pi}}^*(n)) \geq U_{min}\} \tag{4.7}$$

where LU refers to a *Lifted Utility* function, which is yet to be defined. Solving this problem requires gradually increasing the agent number and reevaluating $\bar{\boldsymbol{\pi}}^*(n)$ in $\overline{M}_i(n)$, which will be discussed next.

4.2.1 Isomorphic Utility Calculation

Equations 4.5 and 4.6 show that the sizes of $\overline{\mathbb{T}}_i$, $\overline{\mathbb{R}}_i$, $\overline{\mathbb{O}}_i$, and $\overline{\mathbb{P}}_i$ no longer depend exponentially on N, but K instead. This may lead to the assumption that the same holds for the policy evaluation cost $\overline{\mathbb{C}}_i$, but this is not the case. Equation 2.4 shows that the utility calculation for the full joint policy requires summing both over all states and joint observations $\boldsymbol{o} \in ran(\boldsymbol{O})$, rather than $ran(\overline{\boldsymbol{O}})$. This coincides with the size $\mathbb{O}$ of the (full) sensor function, which is still exponential in N, as given by Equation 2.5. Therefore, the previous result of $\overline{\mathbb{C}}_i$ cannot hold without further adjustments. Three possible approaches to this issue are discussed in this section.

Counting. Next to Isomorphic DecPOMDPs, Braun, Gehrke, et al. (2022) also present Counting DecPOMDPs, which is another form of symmetric DecPOMDPs, where the partition actions $\overline{A}_k$ and observations $\overline{O}_k$ are encoded as histograms in so-called Counting Random Variables (CRVs, Milch et al., 2008). Based on the insight that, for indistinguishable agents, it does not matter which particular agent does or observes something particular, only how many of them, these histograms contain the *counts* of agents who choose each action and make each observation in the respective partitions. This idea could be transferred to the problem at hand: every Isomorphic DecPOMDP is count-convertible, i.e., can be turned into an equivalent Counting DecPOMDP. The reason for not using Counting DecPOMDPs in the first place lies in the size $\overline{\mathbb{P}}_c$ of the policy space, which is given as:

$$\overline{\mathbb{P}}_c \in O(n^{a^{\frac{K(n^{\tau o}-1)}{n^o-1}}}) \tag{4.8}$$

with s, n, K, and o as before in section 4.1.3. The cost $\overline{\mathbb{C}}_c$ of policy evaluation, on the other hand, is only polynomially dependent on the agent number:

$$\overline{\mathbb{C}}_c \in O(sn^{K\tau o}) \tag{4.9}$$

Count conversion, however, requires instantiations of the entire grounded functions T, R, and Ω, which means the compact isomorphic representation would be lost, as the grounded function sizes $\mathbb{T}, \mathbb{R}$, and $\mathbb{O}$ are still exponential in N (Equation 2.5). This may be feasible as a one-time calculation, but

since every additional agent constitutes a new isomorphic model, each of these new models needs to be converted again, for an unbounded number of iterations and with continuously growing agent numbers N.
In anticipation of the following section, these full function instantiations are also required for practical reasons. The implementation builds upon a preexisting software framework that cannot trivially adopt such first-order model specifications, although it does work implicitly for isomorphic models. More precisely, the framework internally operates on indices for agents, actions, and observations, both individual and joint, which are automatically generated based on the full function definitions.
The integration of CRVs into the utility calculation of isomorphic models is not fully understood yet, and not in the scope of this thesis, but may still present an interesting avenue for future research.

Additional Independence Assumption. Another possible approach that does not include CRVs is to introduce an additional independence requirement. Isomorphic DecPOMDPs imply independence among agents of the same partition. This could be extended to other partitions as well, i.e., model specifications could require both inter and intra-partition independence in the agent set, given the state. This additional independence enables a second decomposition of $\overline{T}$, $\overline{R}$, and $\overline{\Omega}$:

$$\begin{aligned}
\overline{T}(\bar{S}', \bar{S}, A_1(X_1), \dots, A_K(X_K)) &= \prod_{k=1}^{K} \overline{T}(\bar{S}', \bar{S}, A_k(X_k)) \\
\overline{R}(\bar{S}, A_1(X_1), \dots, A_K(X_K)) &= \sum_{k=1}^{K} \overline{R}(\bar{S}, A_k(X_k)) \\
\overline{\Omega}(O_1(X_1), \dots, O_K(X_K), \bar{S}) &= \prod_{k=1}^{K} \overline{\Omega}(O_k(X_k), \bar{S})
\end{aligned}$$

As before, this change has practical implications, as the differently shaped inputs on the right-hand side of the equation need to be combined into the form on the left-hand side first in order to match the model parser's syntax. On the model level, this additional independence leads to the following lemma:

Lemma 4.2.1
Given inter-partition independence, an Isomorphic DecPOMDP decomposes into K POMDPs.

Proof. Firstly, the Isomorphic DecPOMDP decomposes into K smaller DecPOMDPs, which are only connected through the state: Since all partitions are independent (given the state), i.e., cannot influence each other, each partition forms its own DecPOMDP. Furthermore, since agents within the same partition k are also assumed to be independent, each partition DecPOMDP can be decomposed once more into n_k identical single-agent models, which are in fact POMDPs by definition. As all these POMDPs within the same partition are identical, it suffices to solve only one representative POMDP per partition, multiply the results by the respective partition size n_k, and sum over all partitions for the total utility. $\square$

The downside of this approach, other than the previous, is not one of technical nature or computational complexity issues, but rather semantic. Requiring inter-partition agent independence in addition to the already existing intra-partition independence heavily reduces the model expressiveness to problems in which agents cannot interfere, interact, or cooperate with each other. Essentially, they each have to exist in a solitary environment that only they can influence, ultimately circumventing the decentralisation property. Consequently, one may view this issue as outside of the DecPOMDP domain. With individual reward functions for each partition, this modelling can also be seen as a shift towards POSGs on the partition level.
The idea of exploiting additional independence assumptions has been considered before, e.g., in the *Network Distributed* POMDP (ND-POMDP, Nair, Varakantham, et al., 2005) subclass of DecPOMDPs (Kim et al., 2006), and in transition-independent DecMDPs (Becker et al., 2004; Wu and Durfee, 2006).

Representative Observations. Another approach, following naturally from the policy calculation on representatives, is to base the utility calculation on representatives as well. To this end, utilities are calculated by summing over partition observations $ran(\overline{\boldsymbol{O}})$ instead of the full joint observations $ran(\boldsymbol{O})$, where each agent adopts the observation of its respective representative. Since all agents of the same partition also employ the same individual policy, this also means each agent adopts the same action as the partition representative. Essentially, utilities are calculated as if each partition acted holistically as singular entity. Under this assumption, any two groundings $d = (x_{1,i_1}, \dots, x_{K,i_K})$ and $d' = (x'_{1,i_1}, \dots, x'_{K,i_K})$ are mapped the same as of $\overline{T}$, and analogously $\overline{R}$ and $\overline{\Omega}$:

$$\overline{T}(\overline{S}', \overline{S}, A_1(x_{1,i_1}), \dots, A_K(x_{K,i_K})) = \overline{T}(\overline{S}', \overline{S}, A_1(x'_{1,i_1}), \dots, A_K(x'_{K,i_K}))$$

Obtaining the grounded values of $\overline{T}, \overline{R}$, and $\overline{\Omega}$, as specified in Equation 4.4, thus simplifies to

$$\begin{aligned}
&\prod_{d \in D} \overline{T}(\overline{S}', \overline{S}, A_1(x_{1,i_1}), \dots, A_K(x_{K,i_K})) = \overline{T}(\overline{S}', \overline{S}, A_1(x_{1,i_1}), \dots, A_K(x_{K,i_K}))^{|D|} \\
&\sum_{d \in D} \overline{R}(\overline{S}, A_1(x_{1,i_1}), \dots, A_K(x_{K,i_K})) = \overline{R}(\overline{S}, A_1(x_{1,i_1}), \dots, A_K(x_{K,i_K})) \cdot |D| \\
&\prod_{d \in D} \overline{\Omega}(O_1(x_{1,i_1}), \dots, O_K(x_{K,i_K}), \overline{S}) = \overline{\Omega}(O_1(x_{1,i_1}), \dots, O_K(x_{K,i_K}), \overline{S})^{|D|}
\end{aligned} \tag{4.10}$$

with $D = \times_{k=1}^{K} dom(X_k)$ the set of all groundings. The functions themselves are left unchanged, retaining the complexity improvements of the isomorphic symmetries. Moreover, assuming similar observations for entire partitions is arguably a less strong constraint than inter-partition independence, which are the reasons why this approach is ultimately chosen.

The utility calculation of Equations 2.3 and 2.4 can now be redefined to incorporate Equation 4.10 into the *lifted* utility function LU, where $L = |D| = \prod_{k=1}^{K} n_k$ is called the lifting constant:

$$LU_{\overline{M}_i}(\overline{\boldsymbol{\pi}}) = \sum_{s_0 \in ran(\overline{S})} \overline{b}_0(s_0) \cdot LU_{\overline{M}_i}^{\overline{\boldsymbol{\pi}}}(s_0, \boldsymbol{o}_{0:0}) \tag{4.11}$$

$$\begin{aligned} LU_{\overline{M}_i}^{\overline{\boldsymbol{\pi}}}(s_t, \boldsymbol{o}_{0:t}) = L \cdot \overline{R}(s_t, \overline{\boldsymbol{\pi}}(\boldsymbol{o}_{0:t})) + \gamma^t \cdot \sum_{s_{t+1} \in ran(\overline{S})} \Big[\, \overline{T}(s_{t+1}, s_t, \overline{\boldsymbol{\pi}}(\boldsymbol{o}_{0:t}))^L \\ \cdot \sum_{\boldsymbol{o}_{t+1} \in ran(\overline{\boldsymbol{O}})} \overline{\Omega}(\boldsymbol{o}_{t+1}, s_{t+1})^L \, LU_{\overline{M}_i}^{\overline{\boldsymbol{\pi}}}(s_{t+1}, \boldsymbol{o}_{0:t+1}) \Big] \end{aligned} \tag{4.12}$$

Note that these definitions syntactically match the function LU in Equation 4.7, since the extensions $M_i(n)$ and $\boldsymbol{\pi}(n)$ still yield a new isomorphic model and joint policy, respectively. To exemplify this approach, refer back to Table 4.1: since every partition agent picks the same action, every grounding d can be viewed as picking the same entry in Table (a), which always translates to one of the eight entries with a value of p_i^4 in Table (b), which are exactly the entries where $A_1(x_{1,1}) = A_1(x_{1,2})$ and $A_2(x_{2,1}) = A_2(x_{2,2})$. Every other entry remains implicit and is never used.

Complexity-wise, the evaluation cost $\overline{\mathbb{C}}_i$ now lies in

$$\overline{\mathbb{C}}_i \in O\big(log_2(L) so^{K\tau}\big) \tag{4.13}$$

Every agent that is added to the model during optimisation needs to be assigned to one of the partitions. This assignment is not arbitrary as the partition sizes determine the lifting constant L. From Equation 4.12 it becomes clear that maximising L also maximises the obtained rewards $\overline{R}$. Since there are no further constraints other than maximal utility, the type of agent(s) to add should be equally distributed across all partitions in order to maximise L. As beneficial side effect, this also leads to a steady increase in the growth rate of L for each additional agent.

4.2.2 Implementation

The Multi-Agent Decision Process (MADP) Toolbox[1] provides a software framework for decision-theoretic planning tasks in multi-agent systems. Although general in design, it is mainly focused around DecPOMDPs. Particularly relevant framework components that are used in this implementation are the DecPOMDP models, the parser for DecPOMDP input files (*.dpomdp*), and the *Generalised* MAA* implementation (GMAA*, Oliehoek, Spaan, et al., 2008), which includes the fundamental MAA* algorithm as described in Section 2.4.

With all the necessary components in place, the optimisation procedure is relatively straightforward, summarised in Algorithm 1. Both the desired utility U_{min} and the DecPOMDP model M_i in the form of a .dpomdp input file are passed via command-line interface. The input DecPOMDP is assumed to be

[1]https://github.com/MADPToolbox/MADP

Algorithm 1: Optimisation of the agent number

```
Data: Isomorphic DecPOMDP model $\overline{M}_i$, desired utility $U_{min}$
Result: Minimal agent number $n^*$ that achieves $U_{min}$
$n^* \leftarrow N$
$\bar{\boldsymbol{\pi}}^* \leftarrow MAA^*(\overline{\Pi}_{\overline{M}_i})$
$U \leftarrow U_{\overline{M}_i}(\bar{\boldsymbol{\pi}}^*)$
while $U < U_{min}$ do
    $n^* \leftarrow n^* + 1$
    $U \leftarrow LU_{\overline{M}_i(n^*)}(\bar{\boldsymbol{\pi}}^*(n^*))$
end
return $n^*$
```

an isomorphic model M_i, i.e., the given transition, reward, and sensor functions are interpreted as the isomorphic instantiations of $\overline{T}, \overline{R}$, and $\overline{\Omega}$, and each agent is regarded as one partition representative. Viewing the input DecPOMDP this way allows Isomorphic DecPOMDPs to be implicitly defined without any changes to the *.domdp* file format, parser, and model specification.
The optimal policy $\bar{\boldsymbol{\pi}}^*$ and its utility $U_{\overline{M}_i}(\bar{\boldsymbol{\pi}}^*)$ are then determined using MADP's (G)MAA* implementation. As long as the obtained utility falls short of U_{min}, the agent number n is increased by one, and the extended policy $\bar{\boldsymbol{\pi}}^*(n)$ and model $M_i(n)$ are reevaluated using the lifted utility function LU as defined in Equation 4.11. Finally, the optimal agent number n^* is returned.

This implementation is publicly available[2], including an exemplary DecPOMDP in the *.dpompd* file format.

[2]https://github.com/cccastan/LiftedMAAStar

5 Discussion

With isomorphic symmetries, a DecPOMDPs complexity can be reduced to exponential dependency on K instead of N, both in terms of representation and policy space size, with particular significance if $K \ll N$. However, solving the new optimisation problem involves repeated reevaluations of the policy with rising agent numbers, which is still exponential in N for each exact valuation. For the sake of completeness, it should be mentioned that the (doubly) exponential dependency on the time horizon τ still persists, but this is a separate challenge altogether. To overcome the exponential dependency on N, Section 4.2.1 discussed three possible changes to the utility calculation:
Introducing CRVs shifts the complexity back towards fully instantiating the transition, reward, and sensor functions, and entails practical challenges for the implementation, but retains an exact valuation. Requiring inter-partition independence to enable a second decomposition of $\overline{T}$, $\overline{R}$, and $\overline{\Omega}$ yields an exact valuation as well, with exponential dependency only on K, but incurs a severe penalty in expressiveness. Since isomorphic models already imply intra-partition independence, this restricts agents entirely to their unique environment.
Representative observations surrender exact valuations, aiming at a middle ground between the drawbacks of approximate valuations and benefits of exponential dependency only on K, without imposing further constraints on the model. Such an approximation may be reasonable for problems where partition agents are confined to sufficiently small environments, where there is a high likelihood of every agent making the same observation, or where the effects of slightly different observation hardly deviate from each other as of $\overline{T}$, $\overline{R}$, and $\overline{\Omega}$. In the case of a model in which partitions do indeed receive the same observations (i.e, not just as a simplifying assumption), the valuations are in fact exact.
All three of these approaches have the assumption of intra-partition independence in common, as implied by isomorphic symmetry, which already is a strong assumption by itself. In general, any independence may in particular be restrictive for modelling agent movements around shared spaces in which they can collide, e.g., multiple robots in the same grid.

In summary, none of these approaches solves the complexity problem of utility calculation perfectly, i.e., with guaranteed exact valuations, exponential dependence only on K instead of N, and without making further assumptions. An extension to the current implementation could choose between these approaches, based on which of the additional assumptions hold, covering more eventualities than with one fixed approach. If none holds, the latter two approaches may still serve as approximations of the actual valuation, although with rising agent numbers, the inaccuracy increases as well.

With representative observations, the agent distribution across partitions is currently limited to be uniform in order to maximise the lifting constant L, and thus the overall utility. From a practical standpoint, it may be desirable to differentiate partitions by their contribution to the overall utility, and optimise the utility gained per agent of a certain type against, for instance, its deployment cost. This is not possible with representative observations.

The current implementation assumes the given DecPOMDP to be isomorphic. In fact, any DecPOMDP model can be regarded as an isomorphic representation of another implied ground DecPOMDP model. Determining whether a ground model exhibits isomorphic symmetry is a different, non-trivial challenge that may be tackled in future works (refer to Braun, Gehrke, et al. (2022) on this discussion as well).

6 Conclusion and Outlook

This thesis discusses strengths and shortcomings of several approaches to the implementation of a new kind of query and the underlying optimization problem in Isomorphic DecPOMDPs.
In this special type of a Partitioned DecPOMDP, both the policy space and the representation sizes of the transition, reward, and sensor functions no longer scale exponentially with the number of agents, but with the number of agent types instead, i.e., the number of partitions. The optimal joint policy can be calculated by considering partition representatives only, and subsequently extending this policy to any partition size. These partition sizes do not change the agent behaviour, but they do influence the expected utility.

This enables a new query type that asks for the optimal number of agents needed to obtain a certain utility value in a given Isomorphic DecPOMDP. In contrast to the policy space and function representations, however, the policy evaluation cost does still depend exponentially on the agent number. To overcome this problem, three amendments are considered: the use of Counting Random Variables, an additional independence assumption between partitions, and representative observations. None of these solves the problem in general, particularly the latter two require strong assumptions to hold. Representative observations are ultimately chosen to be implemented as best compromising solution. Further work concerning the combination of isomorphic symmetries and Counting Random Variables may lead to a general solution.

This thesis is the first to implement this new kind of query and optimisation problem in Isomorphic DecPOMDPs. Consequently, there exist no benchmarks or similar implementations for comparison. As new approaches may be devised in future works, or one of the other aforementioned approaches gets implemented, a formal comparative evaluation can yield further insights into different solution qualities.

List of Figures

List of Tables

References

[1] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N. Kemal Üre, and Mykel J. Kochenderfer. “Decentralized Control of Partially Observable Markov Decision Processes”. In: *Proceedings of the 52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2398–2405.

[2] Udi Apsel and Ronen I. Brafman. “Extended Lifted Inference with Joint Formulas”. In: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2011, pp. 11–18.

[3] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. “Solving Transition Independent Decentralized Markov Decision Processes”. In: *Journal of Artificial Intelligence Research* 22 (2004), pp. 423–455.

[4] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. “The Complexity of Decentralized Control of Markov Decision Processes”. In: *Mathematics of operations research* 27.4 (2002), pp. 819–840.

[5] Abdeslam Boularias and Brahim Chaib-draa. “Exact Dynamic Programming for Decentralized POMDPs with Lossless Policy Compression”. In: *Proceedings of the 18th International Conference on Automated Planning and Scheduling*. AAAI Press, 2008, pp. 20–27.

[6] Craig Boutilier, Raymond Reiter, and Bob Price. “Symbolic Dynamic Programming for First-Order MDPs”. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence*. IJCAI-01. 2001, pp. 690–697.

[7] Tanya Braun, Marcel Gehrke, Florian Lau, and Ralf Möller. “Lifting in Multi-agent Systems under Uncertainty”. In: *The 38th Conference on Uncertainty in Artificial Intelligence*. 2022.

[8] Tanya Braun and Ralf Möller. “Parameterised Queries and Lifted Query Answering”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI Organization, 2018, pp. 4980–4986.

[9] Alan Carlin and Shlomo Zilberstein. “Value-Based Observation Compression for DEC-POMDPs”. In: *7th International Joint Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 2008, pp. 501–508.

[10] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous Localization and Mapping: Part I”. In: *IEEE Robotics & Automation Magazine* 13.2 (2006), pp. 99–110.

[11] Marcel Gehrke, Tanya Braun, and Ralf Möller. “Lifted Temporal Maximum Expected Utility”. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*. Springer, 2019, pp. 380–386.

[12] Marcel Gehrke, Tanya Braun, Ralf Möller, Alexander Waschkau, Christoph Strumann, and Jost Steinhäuser. “Lifted Maximum Expected Utility”. In: *Artificial Intelligence in Health*. Springer, 2018, pp. 131–141.

[13] Claudia V. Goldman and Shlomo Zilberstein. “Optimizing Information Exchange in Cooperative Multi-Agent Systems”. In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS ’03. Association for Computing Machinery, 2003, pp. 137–144.

[14] Steven Holtzen, Todd D. Millstein, and Guy Van den Broeck. “Generating and Sampling Orbits for Lifted Probabilistic Inference”. In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2019, pp. 985–994.

[15] Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. “Counting Belief Propagation”. In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 277–284.

[16] Yoonheui Kim, Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. “Exploiting Locality of Interaction in Networked Distributed POMDPs”. In: *AAAI Spring Symposium: Distributed Plan and Schedule Management*. AAAI Press, 2006, pp. 41–48.

[17] Florian-Lennert Adrian Lau, Florian Büther, Regine Geyer, and Stefan Fischer. “Computation of Decision Problems Within Messages in DNA-tile-based Molecular Nanonetworks”. In: *Nano Communication Networks* 21 (2019).

[18] John McCarthy. *Situations, Actions, and Causal Laws*. Tech. rep. Stanford University, 1963.

[19] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. “Lifted Probabilistic Inference with Counting Formulas”. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. AAAI Press, 2008, pp. 1062–1068.

[20] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. “Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings”. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. Vol. 3. 2003, pp. 705–711.

[21] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. “Networked Distributed POMDPs: A Synthesis of Distributed Constraint Optimization and POMDPs”. In: *Proceedings of the 20th National Conference on Artificial Intelligence*. AAAI Press, 2005, pp. 133–139.

[22] Aniruddh Nath and Pedro M. Domingos. “A Language for Relational Decision Theory”. In: *Proceedings of the 6th International Workshop on Statistical Relational Learning*. 2009.

[23] Aniruddh Nath and Pedro M. Domingos. “Efficient Lifting for Online Probabilistic Inference”. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[24] Mathias Niepert and Guy Van den Broeck. “Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference”. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI Press, 2014, pp. 2467–2475.

[25] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[26] Frans A. Oliehoek, Julian Kooij, and Nikos Vlassis. “The Cross-Entropy Method for Policy Search in Decentralized POMDPs”. In: *Informatica* 32 (Jan. 2008), pp. 341–357.

[27] Frans A. Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. “Optimal and Approximate Q-Value Functions for Decentralized POMDPs”. In: *Journal of Artificial Intelligence Research* 32 (2008), pp. 289–353.

[28] David Poole. “First-order Probabilistic Inference”. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI Organization, 2003, pp. 985–991.

[29] David Poole. “The Independent Choice Logic for Modelling Multiple Agents Under Uncertainty”. In: *Artificial Intelligence* 94.1-2 (1997), pp. 7–56.

[30] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

[31] Scott Sanner and Craig Boutilier. “Approximate Solution Techniques for Factored First-Order MDPs”. In: *Proceedings of the 17th International Conference on Automated Planning and Scheduling*. AAAI Press, 2007, pp. 288–295.

[32] Scott Sanner and Kristian Kersting. “Symbolic Dynamic Programming for First-order POMDPs”. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 2010, pp. 1140–1146.

[33] Sven Seuken and Shlomo Zilberstein. “Improved Memory-Bounded Dynamic Programming for Decentralized POMDPs”. In: *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2007, pp. 344–351.

[34] Sven Seuken and Shlomo Zilberstein. “Memory-Bounded Dynamic Programming for DEC-POMDPs”. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. IJCAI Organization, 2007, pp. 2009–2015.

[35] Siddharth Srivastava, Stuart Russell, Paul Ruan, and Xiang Cheng. “First-Order Open-Universe POMDPs”. In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2014, pp. 742–751.

[36] Daniel Szer, François Charpillet, and Shlomo Zilberstein. “MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs”. In: *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*. AUAI Press, 2005, pp. 576–590.

[37] Nima Taghipour. “Lifted Probabilistic Inference by Variable Elimination”. PhD thesis. KU Leuven, 2013.

[38] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. “Lifted Probabilistic Inference by First-Order Knowledge Compilation”. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. IJCAI Organization, 2011, pp. 2178–2185.

[39] Jianhui Wu and Edmund H. Durfee. “Mixed-Integer Linear Programming for Transition-Independent Decentralized MDPs”. In: *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems.* ACM, 2006, pp. 1058–1060.

Declaration of Academic Integrity

I hereby confirm that this thesis on

Lifting Multi-agent A -*
Implementing a New Optimisation Problem in Isomorphic DecPOMDPs:
How Many Agents Do We Need?

is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

Constantin Castan, Münster, November 3, 2022

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

Constantin Castan, Münster, November 3, 2022