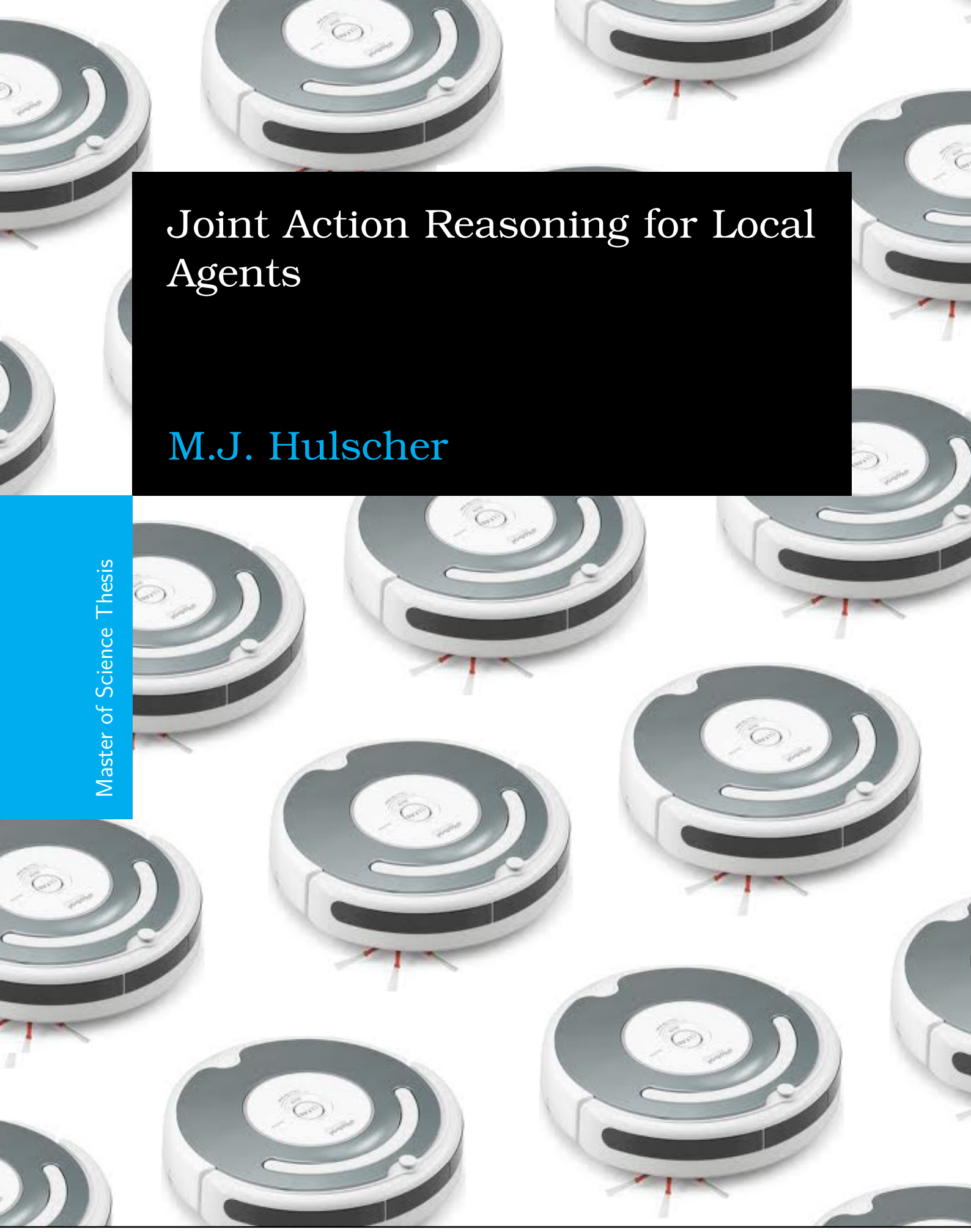


Joint Action Reasoning for Local Agents

M.J. Hulscher

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Joint Action Reasoning for Local Agents

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M.J. Hulscher

June 7, 2013

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Almende B.V. Their cooperation is hereby gratefully acknowledged.

Abstract

Decentralized MDPs provide an extensive framework that can model the interaction between multiple agents making decisions sequentially in a stochastic environment. Finding an optimal policy for problems in this framework however requires a large computational effort. For instance, an infinite-horizon problem, is proven to be undecidable in the worst-case.

In this thesis we show an alternate route to obtain a decentralized policy. Hereby, we rely on the use of optimal global behavior and an altered local belief.

The optimal global behavior is defined by the solution to the underlying centralized MDP problem. We solve the underlying MDP problem to obtain a global policy which determines the optimal action in a given state. The agents in a decentralized decision problem do not have full access to this state however. They use limited information to build a belief about the current state they occupy. A standard belief state update does not inherently include any reasoning about the actions of other agents however. In this thesis we altered the agents' local belief state by adding a term that incorporates explicit reasoning about the actions of other agents.

We test this method on a set of commonly used benchmark problems. We will show that we can positively affect the cooperative behavior of the agents in a decentralized decision problem by altering the local belief states of the individual agents within it.

Table of Contents

Chapter 1

Summary

If we were to look at robots that are currently employed in everyday households, such as vacuum robots, we see that these robots are mainly able to work within bounded environments by themselves. Suppose we would want to clean larger environments with more robots, we would find that this is not so easily done. It would require the decentralized execution of a task in which the robots should cooperate. We depart from the explicit modelling of such a robot problem, and focus on the intricacies involved with decentralized planning and execution for a much broader class of this kind of problem. We assume that the generic form of these problems can be described with a stochastic model in which the robots can perform high-level actions that affect their environment. The robots should all individually decide on these actions without explicit communication among each other to aid them.

There are multiple frameworks available to model this stochastic interaction between agents and their environment. These frameworks model environments in which the agent or agents should pick and execute an action every discrete time step. In this thesis, we further assume discretized state, action and observation spaces in each of the frameworks mentioned. At each time step, the agent(s) receive information pertaining to the current state of the system. Based on this information and other stored information, the agent(s) should pick the following action to take.

In this thesis, we use three separate frameworks, all active in stochastic decision modelling. The first framework is the standard MDP , which is used to model the activity of a single agent. The agent chooses and executes an action every time step, resulting in a stochastic effect on the next state of the system.

If the agent in question, no longer has an accurate or full view of the state, we can model this with the second framework, which is the POMDP. In this framework, the agent only receives a fuzzy signal with some information about the current state. This information is collected into the belief state, which is a probability distribution over all possible states the agent might occupy.

The third framework governs modelling problems with more agents. While separately naming them, we consider these multi-agent extensions of the MDP and POMDP to be very much

like each other. These extensions are the Dec-MDP and Dec-POMDP frameworks. These frameworks both describe problems in which agents must cooperate to achieve their goals. The agents cannot, however, observe the complete current state. They can only perceive their own small part of it. There are algorithms with which we could calculate a policy for such a problem, though these algorithms require a great amount of computational effort while still being limited to planning for a set horizon.

Our goal is to find a more computationally efficient way to build a decentralized policy. Instead of directly solving the decentralized problem, we make use of the underlying MDP which can be built by temporarily assuming full observability of the state by all agents. This way we can solve the MDP and obtain a global policy that links states to a corresponding optimal joint action. We hereby utilize the fact that it is easier to solve a standard MDP problem than its decentralized counterpart. We use this global policy combined with a belief-state and several heuristic functions for each agent to make appropriate action decisions. We also apply the global policy information to the belief-state update rule in order for the agents to reason about the actions of other agents.

Evaluation of this approach is done by testing it on a set of existing benchmark problems called the 'recycling robots' problem, the 'broadcast channel' problem and the 'meeting on a grid' problem. We evaluate our approach by viewing the expected rewards we can obtain in these problems. These expected rewards are obtained by collecting the rewards over multiple simulation runs. We compare our results to currently known best results in both infinite horizon and finite horizon methods for decentralized problems.

Running the simulations showed that we can achieve good results on all provided benchmark problems. In two of the three benchmarks, we did not improve the collective behavior of all agents. This was due to the fact that they already performed well without the added knowledge. On the 'meeting on a grid' problem, we found that the addition of global policy information improved the behavior. This in turn shows that the addition of global action reasoning can improve the behavior of the agent in cooperation. This performance enhancement is not guaranteed however.

Chapter 2

Introduction

2-1 Problem Setting

Executing a task with multiple agents requires all actions made by the individuals to be coordinated with each other. This type of coordination between multiple agents can be done easily if all agents can communicate perfectly with each other. It becomes much more difficult however, when the communication is cut off and the agents must cooperate and base their actions on locally obtainable information only. This is the type of problem setting we envision in this thesis. We wish for multiple agents to cooperate without explicit communication, essentially tackling a cooperative problem decentrally.

We envisioned these types of scenarios in order to aid us in developing cooperative techniques for multi-agent problems. These problems include multi-agent search and rescue operations, (submarine) exploration, distributed resource allocation or cleaning to name a few.
We can illustrate the cleaning robots problem more specifically, as a cooperation effort between multiple roomba robots. Currently, these cleaning robots do not possess the ability to cooperate with one another in any way. They are designed and built to vacuum an average sized living room by themselves. This is expressed by their battery life and behavior. In order to process and clean a larger and more complex environment however, we wish for multiple robots to cooperate. They could take over each others' work when in need and divide a large workload of a large space into individually achievable chunks.

In addition to being able to just cooperate, we would want the robots to do this without explicit communication among each other. In other words, the robots must coordinate their actions without intermittently communicating their states or findings. This desire stems from the real world possibility that communication might not be achievable in some cases. Either robots do not possess communicative hardware (such as most cleaning robots) or the environment is duly large that communication channels cannot span it.

Originally our goal was to model the multi-robot cleaning problem using decentralized markov decision process frameworks. We could then use this model to build a local policy for each robot and apply these in a real world setting. This goal however gradually changed into finding

a new way to approach general decentralized decision making. Specifically, we investigated a novel way to incorporate a centralized policy into the decentralized execution of a problem. The idea here is that we solve an underlying problem that does not take into account the decentralized nature of a problem. We then utilize the formed centralized policy to have individual agents build a belief about the actions of other agents.

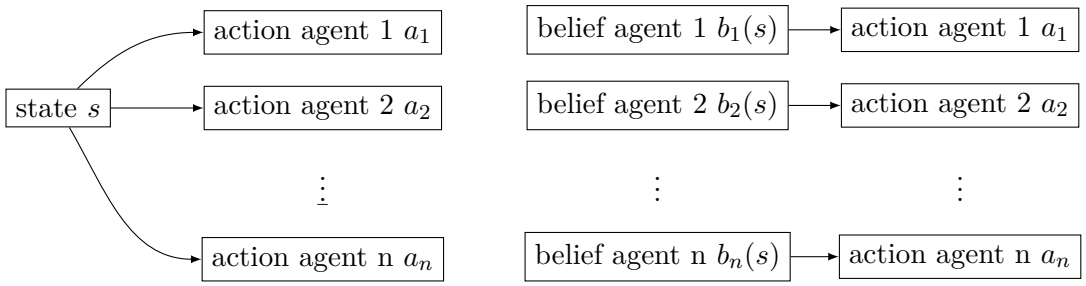
2-2 Decision Theoretic Planning

The standard Markov Decision Process (MDP) framework has been extensively researched over the years and gained much traction in the scientific community as the preferred framework to model sequential decision making in a stochastic environment[1, 2]. The framework models an agent that has the ability to take actions that affect its environment stochastically. The aim of solving, or planning for an MDP problem is to find a strategy for the agent to behave in such a way that it maximizes a given performance criterion. An agent in an MDP setting has a complete view of the state of the system at any given time, which might not be entirely realistic in real world settings where sensor readings can sometimes be false or otherwise inaccurate.

This partial observability of the state can be modelled using the Partially Observable Markov Decision Process framework. In this framework, the agent can no longer view the complete state, but only obtains observations with fuzzy information about the state. The reduced clarity of the state comes at a price, as planning for this particular type of stochastic system requires more computational effort than for a standard MDP. In general POMDP problems, the agent is required to balance the need to optimize its reward gain with the need to gain additional information about the state of the system. To achieve this, the agent needs to reason about multiple possible states it might occupy. This is achieved by keeping track of a *belief-state* which is a probability distribution over states the agent might occupy. The belief state is updated at each time step after the agent obtained its observation.

For some problems we require multiple agents to work together to achieve a certain goal. This too can be modelled, using a Markov Decision Process framework by either extending MDP directly to model multiple agents. Otherwise we could extend the POMDP framework, which in turn will restrict the agents' observation of the state. These multi-agent decision processes model the agents' stochastic interaction with the environment as well as the interactions among the multiple agents. Though multi-agent decision processes have been known to allow for full state observability, most research is done on partially observable subclasses [3, 4, 5, 6]. These partially observable multi-agent decision processes are described by the Dec-MDP and Dec-POMDP frameworks.

In the field of multi-agent stochastic decision processes, we generally separate the execution of the problems from solving them. In other words, we generally first solve the problem in a separate step before we can execute it. The reason behind this is that executing agents should not be able to perceive their performance measure, as this can constitute as an indirect way of communication [2].
In order to solve a multi-agent decision process, we usually assume that the planning algorithm has full access to the interaction mechanics of the environment in addition to the model of stochasticity of the observations. Planning for the problem itself focusses on finding a set



(a) global policy where the complete state is mapped to all agent actions at once. We call refer to the collection of all actions as the joint action or global action.

(b) local policies where each agent has their own mapping from local information to local action.

Figure 2-1: Difference between local and global policies.

of local actions for every agent, based on their sequences of observations within the system. Though much advancements have been made, planning for a multi-agent problem proves to be intractable for most larger problems.

We make use of matters concerning MDP and POMDP to solve their multi-agent extensions. In particular, we use the *underlying MDP* of a given decentralized problem to build a global policy which maps fully observable states to actions for every agent as graphically shown in Figure 2-1a. This as opposed to local policies which only provide mappings from individual agent (local) beliefs to local actions as shown in Figure 2-1b.

2-3 Applications of Decision Theoretic Planning

Previously, we described general problem settings for which multi-agent decision process frameworks can be used. In literature, there are many examples of such problems, some of which we will discuss here.

For example, we can identify a real-world application in robot soccer competitions [7], where a team of robots must cooperate to win a soccer match. The robots must perform high-level actions pertaining to navigation and game strategy, in order to achieve their goal of winning a match. The robots have access to limited and fuzzy information about their direct surroundings, not to mention the information being fed to the other agents. The outcome of the actions taken by the robots is considered stochastic since there is always a chance of failure. An example of this is passing a ball which could be intercepted by the opposing team.

Other examples can include multi-agent navigational tasks [8] and exploration tasks [9]. These tasks require multiple agents to traverse an environment and map it. The agent are required to individually take actions and take measurements of the environment, eventually resynchronising with one another to build a full map. The agents should individually decide in which direction to explore, leaving the rest of the environment to the others.

Quite some research has also been put into using decentralized decision models in order to improve performance of distributed sensor networks [10, 11, 12]. We can illustrate the situation in these models as a set of sensor nodes that need to collectively gather information

about a certain environment aspect. For example, this could be illustrated by a set of nodes that must track a certain target that can pass different nodes. The action of scanning for the target incurs a cost which can only be offset by finding a target and gaining rewards.

Another field of use is Search and Rescue operations [13, 14], where multiple agents are tasked with finding victims in a cluttered environment and extracting them. This task requires the agents to cooperatively search the area and help each other remove obstacles in their path. This paves the way for a planning tasks in which agents with very heterogeneous set of skills, where some are able to clear out debris and others can safely extract victims on site.

2-4 Focus of this Thesis

We focus on the utilization of a centrally solved policy in the decentralized execution of a multi-agent problem. The idea is that we solve an underlying version of a decentralized stochastic problem where we temporarily assume full observability and communication between the agents. This way we have an optimal planning from state to global action.
Adapting this global policy to individual agents can be done through heuristic functions based on the local belief state of a agent. What is novel however, is that we also incorporate the global policy in the local belief update function of every agent. This way, we introduce a way for individual agents to reason about the actions of others. This in turn can lead to an improved performance of the collective behavior of all agents. One thing to note however is that we cannot guarantee optimal behavior with this approach. It serves to investigate the potential of influencing collective behavior through local belief update alterations.

2-5 Research Questions

As previously mentioned, we want to make use of the underlying centralized version of a decentralized problem. The goal is to add useful information to individual agents pertaining to the activity of others. In order to do this we ask ourselves some questions which we would like to have answered by the end of this thesis. The are:

- Can we extend the local belief of agents in a decentralized problem to incorporate a global policy?
- Can we improve the cooperative behavior in a decentralized setting by altering the local belief update for individual agents?
- Can we guarantee performance improvement by altering the local belief update?

2-6 Outline

This thesis is organised as follows. In Chapter 3 we will specify in further detail, the MDP framework and its generalisations, the partially observable POMDP and multi-agent frameworks Dec-MDP and Dec-POMDP. This chapter will also contain content that explains the

current methods used in these frameworks. Chapter 4 will describe the novel approach, devised during the course of our research. This approach utilizes a simpler underlying version of a decentralized problem to aid in building a decentralized executable policy. Experiment details and results are provided in Chapter 5, followed by Chapter 6 with conclusions and recommendations.

Chapter 3

Markov Decision Processes

3-1 Introduction

A problem posed for a multi-agent system can be written in the form of a Decentralised Markov Decision Process (Dec-MDP) or Decentralized-Partially Observable Markov Decision Process (Dec-POMDP). These frameworks model the independent decision making of multiple agents and their limited view of the environment and its states. In this chapter, we will show how the Dec-MDP and Dec-POMDP frameworks are logically built from the well established Markov Decision Process (MDP) which is used in the field of Reinforcement Learning (RL) to emulate the interaction between an agent and its environment. This chapter begins by explaining the standard MDP and continues to expand on that by incorporating partial observability of the state with the POMDP framework. Further on we will introduce the multi-agent markov decision frameworks with Dec-MDP and Dec-POMDP.

3-2 Markov Decision Processes (MDPs)

The Markov Decision Process (MDP) [1, 15] has been widely used in literature as a mathematical framework for sequential decision making in stochastic domains. More specifically, an MDP models an agent's interaction with a stochastic environment. During the course of this thesis we assume we can discretize all aspects of this interaction. In other words, we model the set of agent actions and the set of states it can occupy as finite sets. We also assume that time is divided into discrete steps. At every discrete time step, the agent is given the opportunity to pick and execute an action based on the current system state. This action, combined with the current state, stochastically affect the transition to a new system state. After each transition to a new state, the agent is given a reward, depending on which action it took and which new state it transitioned to. The rewards gained over multiple time steps are combined in a performance measure. It is the goal of the agent to optimize its performance measure over a specified number of steps. In order to do this, we build a *policy* that describes

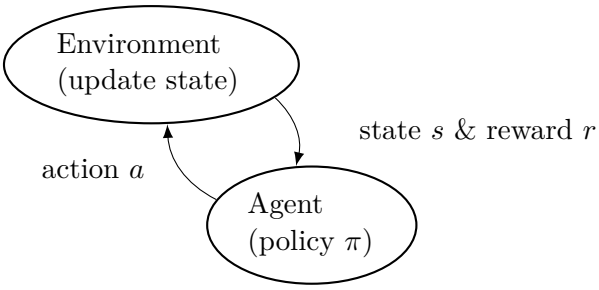


Figure 3-1: MDP: agent interaction with its environment.

which action to take when the system is in any particular state. If the policy is optimal, it describes the optimal actions to take, starting from each state.

Figure 3-1 displays the interaction between an agent and its environment during a single time step. At any time instant k, the agent will be in a certain state $s_k \in S$. At this point, the agent has the option of choosing any action $a_k \in A$. After choosing and executing a particular action, the system transitions to a new state $s' \in S$. At this time, the agent also receives a reward r depending on the action it took and the state it reached. Reaching a particular state is the result of the entire series of previous states and actions. The probability of ending up in a state s_{k+1} and receiving reward r is written as:

$$Pr\left\{s_{k+1} = s', r_{k+1} = r | s_k, a_k, s_{k-1}, a_{k-1}, ..., s_0, a_0\right\} \tag{3-1}$$

where s_k and a_k are the states and actions respectively at time step k.
One important aspect of a Markov Decision Process is the *Markov Property* [1] which states that the entire state-action history up to the last state transition $s_k, a_{k-1}, s_{k-1}, a_{k-2}, ..., a_0, s_0$ can be summarized by just the the current state s_k. Incorporating this into (3-1) results in:

$$Pr\left\{s_{k+1} = s', r_{k+1} = r | s_k, a_k\right\} \tag{3-2}$$

This implies that all relevant information pertaining to the decision at hand is available in the current state description of the system. With this assumption, the probability of reaching a certain state $s_{k+1} = s'$ can be calculated more easily since we only need to take into account the last state s_k and action a_k.

$$T(s, a, s') = Pr\left\{s_{k+1} = s' | s_k = s, a_k = a\right\} \tag{3-3}$$

Where T is the state transition probability function as defined in Definition 3.1. Similarly, given any current state s_k and action a_k, the expected reward can be calculated by:

$$R(s, a, s') = E\left[r_{k+1} | s_k = s, a_k = a, s_{k+1} = s'\right] \tag{3-4}$$

All these aspects of the MDP combined make up the entire framework. We formally define the MDP in Definition 3.1.

Definition 3.1 (MDP)**.** The MDP is defined as a tuple $\langle S, A, T, R \rangle$, where

- S is a discrete, finite set of states
- A is a discrete, finite set of actions
- T is a state transition probability function defined as $T : S \times A \times S \to [0, 1]$
  Functionally denoted by the probability $T(s, a, s') = Pr\{s'|s, a\}$[1] that the system transitions to state s' from state s after using action a.
- R is a reward function defined as $R : S \times A \times S \to \mathbb{R}$
  Functionally denoted by the reward $r_{k+1} = R(s, a, s')$ given to the agent if it reaches state $s_{k+1} = s'$ after taking action a in state s

The reward function R and state transition function T completely describe the system dynamics and the value of all actions, and are used to calculate a policy for the agent.

The MDP-framework heavily relies on the fact that the agent has complete access to the state. In reality however, sensors can be noisy or otherwise inaccurate resulting in an uncertain state measurement. To account for this more realistic setting, a generalised variant of the MDP framework exists to account for partial observability of the state space. This Partially Observable Markov Decision Process (POMDP) will be discussed in Section 3-3. The process and tools to create a policy for an MDP will first be discussed in the following subsections.

3-2-1 Performance Measures

For the agent in the MDP-framework, the objective is to maximize a performance measure over a series of time steps. This performance measure indicates the expected long term reward the agent can obtain. One standard performance measure is the *finite-horizon optimality* [16], through which the agent aims to maximize the expected sum of rewards it receives over the upcoming h time steps, where h is called the horizon of the problem. It represents the number of steps into the future that the agent has to take into account when choosing an action. The finite-horizon optimality is written as:

$$E\left[\sum_{k=1}^{h} r_k\right] \tag{3-5}$$

where r_k is the reward received at time step k.

The general approach of the finite-horizon optimality is very basic. It does pose some problems in its applicability however. The main reason is that any possible reward in the future is valued as much as any reward that can be obtained immediately. This can be counteracted by discounting future rewards, as done in (3-6).

$$E\left[\sum_{k=1}^{h} \gamma^k r_k\right] \tag{3-6}$$

[1] Writing $Pr\{x|y\}$ denotes the probability of x happening, given the occurrence of y.

Where the discount factor falls within the range $0 < \gamma \leq 1$.
Here the immediate rewards are more valuable to the agent, effectively prioritizing immediate reward over long-term cumulative reward. Reducing the discount factor will put more emphasis on the immediate reward, whereas a larger discount factor will allow for a greater effect of possible future rewards.
Due to the nature of a discount factor, rewards far enough into the future are discounted enough to be completely negligible. This means we do not necessarily need to choose a horizon. Instead, we can reason over an indefinite number of time steps. This is called the infinite-horizon discounted model.

$$E\left[\sum_{k=1}^{\infty} \gamma^k r_k\right] \tag{3-7}$$

Although the reward is summed up over an infinite horizon, the discount factor ensures that the sum is still finite. This does mean that the discount factor can no longer be equal to 1 in this case.
Whether to require a finite or infinite horizon optimality is usually dependent on the performance requirements of the designer of a particular MDP problem. If we expect the problem to be executed for a limited time only, we might require a finite horizon performance measure since this might influence decision making in certain instances. Otherwise, if we would want the agent to make informed decisions at any time instant during a longer run, we would use an infinite horizon measure.

3-2-2 Policy

The aim of solving an MDP is to find an optimal policy π. A policy is considered optimal if the action being mapped to, yields the greatest expected cumulative reward.
A policy is typically either one of two options. A *stationary policy* is a direct mapping from the current state to an action $\pi : S \rightarrow A$. In a *nonstationary policy* π_k, the action to be taken is decided on both state and the current time step k. Such an approach can only be used for finite horizon problems since the policy would need to be infinitely large for infinite horizon policies. Due to the nature of finite-horizon optimality, a finite-horizon policy is not necessarily stationary. If an agent chooses an action to take, it is usually going to matter if it is the last action possible or there is still a time to go.

Policies can also be classified as being either deterministic or stochastic. A deterministic policy maps states directly to actions $\pi : S \rightarrow A$. A stochastic policy however maps to actions in a probabilistic manner $\pi : S \times A \rightarrow [0, 1]$. In this thesis, we only use deterministic policies.

To define the performance of a particular policy, we can evaluate it using one of the previously mentioned performance measures. This evaluation results in a value function for the policy. We further explain value functions in the next subsection.

3-2-3 Value Functions

To evaluate the performance of a policy, we use the value function $V^{\pi}(s)$ of a particular policy. The value function describes the expected reward that can be obtained if the agent starts

from any given state and then continually follows the policy. Calculating the value function for a policy is done by finding the expected reward from any state. In the infinite-horizon case, this can be expressed by the equation:

$$V^{\pi}(s) = E^{\pi}\left[r_{k+1}|s_k = s\right] = E^{\pi}\left[\sum_{j=0}^{\infty}\gamma^j r_{k+j+1}|s_k = s\right] \tag{3-8}$$

where E^{π} denotes the expected value given that the agent follows the policy π. Often, the value function and policy are combined into a state-action value function or *Q-function*. The Q-function builds up the expected value of each state-action pair. It describes the expected return starting from state s, taking action a and after that continuing on following policy π. The Q-function for an infinite horizon model is expressed by:

$$Q^{\pi}(s,a) = E^{\pi}\left[r_{k+1}|s_k = s, a_k = a\right] = E^{\pi}\left[\sum_{j=0}^{\infty}\gamma^j r_{k+j+1}|s_k = s, a_k = a\right] \tag{3-9}$$

The value functions are similarly defined for finite-horizon problems.

The value functions are obtained through experience by maintaining an average of all results gained after having been in a certain state, for each state. This is done iteratively by using the *Bellman equation* shown below, as an assignment.

$$\begin{aligned}
V^{\pi}(s) &= E^{\pi}\left[\sum_{j=0}^{\infty}\gamma^j r_{k+j+1}|s_k = s\right] \\
&= E^{\pi}\left[r_{k+1} + \gamma\sum_{j=0}^{\infty}\gamma^j r_{k+j+2}|s_k = s\right] \\
&= \sum_{s'\in S} T(s', \pi(s), s)\left[R(s', a, s) + \gamma E^{\pi}\left[\sum_{j=0}^{\infty}\gamma^j r_{k+j+2}|s_{k+1} = s'\right]\right] \\
&= \sum_{s'\in S} T(s', \pi(s), s)\left[R(s', a, s) + \gamma V^{\pi}(s')\right]
\end{aligned} \tag{3-10}$$

The Q^{π} value function is updated in a similar fashion. The optimal policy π^* is a policy that receives the most reward starting from any initial state. For an agent who has found the optimal policy, the following equations hold:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{3-11}$$

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{3-12}$$

$$\tag{3-13}$$

for all $s \in S$ and $a \in A$, where $V^*(s)$ and $Q^*(s,a)$ are the optimal value functions.

Deriving the actual policy from the value functions is done by choosing the action that maximizes future reward at each state. This is expressed by:

$$\begin{aligned}\pi(s) &= \arg\max_a Q(s,a) \\ &= \arg\max_a E\left[r_{k+1} + \gamma V^\pi(s_{k+1}|s_k = s, a_k = a\right] \\ &= \arg\max_a \sum_{s'} T(s',a,s)\left[R(s',a,s) + \gamma V^\pi(s')\right]\end{aligned} \tag{3-14}$$

where $\arg\max_a$ describes the particular action a that results in the maximization of the remaining expression. We will commonly evaluate a new value function based on the policy defined this way and iteratively improve upon the result. Eventually approaching a maximum. There are some challenges associated with calculating an optimal policy. The main difficulty lies in the so-called *curse of dimensionality*. This means that the problem size increases exponentially with increasing state space size, making it computationally harder to calculate the value function of a given policy.

3-2-4 Dynamic Programming

Dynamic Programming (DP) is a term used to describe a set of algorithms that are able to find an optimal policy for a standard MDP problem, provided the algorithms have access to a complete model of the problem. In this thesis we assume to have access to this complete model for policy generation purposes. DP encompasses two core algorithms called *Policy Iteration* and *Value Iteration*.

Policy Iteration

Policy Iteration describes an algorithm in which the policy is built using two separate steps iteratively. In the first step, the current proposed policy is evaluated using the expression previously defined in (3-10). During this step, the algorithm essentially calculates the value V^π of the current policy. Often the Bellman equation is applied to each state iteratively to build the current value of the policy. An alternative approach is to use Linear Programming [17] to calculate the value.

During the second step, the policy is improved by calculating the expected reward of each state-action combination and adding that to the discounted value function.

$$Q^\pi(s,a) = \sum_{s'} T(s',a,s)\left[R(s',a,s) + \gamma V^\pi(s')\right] \tag{3-15}$$

The new policy can then be found by plugging $Q(s,a)$ into (3-14). A this point, if there exists some action $a \in A$ where $Q^\pi(s,a) > V^\pi(s)$, the policy will be improved in that particular state. If the policy cannot be further improved, that means that its value function already

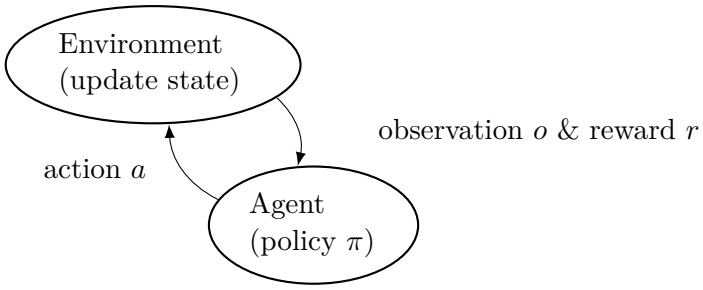


Figure 3-2: POMDP: agent interaction with its environment.

satisfies the Bellman equation and therefore is optimal.
Performing policy iteration on a discrete MDP problem will guarantee it to converge within a finite number of iterations [2]. It is worth noting that a policy iteration algorithm requires an initial policy π_0 as input. Also, the evaluation step is computationally expensive since it requires passing over every state multiple times per iteration.

Value Iteration

In a Value Iteration algorithm, the focus is less on directly interacting with the policy, and more with the value function. Essentially, the value iteration algorithm combines the two steps seen in Policy Iteration, no longer requiring the use of an explicit policy to be calculated during each iteration. Instead, it iteratively computes an approximation of the value function until it converges to the optimal value function. Starting with an initial value V_0 the algorithm iteratively performs the Bellman update according to:

$$V'(s) = \max_a \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V(s') \right] \tag{3-16}$$

Value iteration guarantees an optimal value function V^* in the limit [1]. Subsequently, we can derive an optimal policy from that by applying (3-14).

Despite both algorithms achieving the same goal of finding an optimal policy, they have additional strengths and weaknesses. Policy iteration will generally converge faster but at the cost of expensive value computations at every iteration. Value iteration, on the other hand, may converge slower, especially if the discount factor γ is closer to 1. However, it has a less computationally demanding update process, especially with respect to state space size.

3-3 Partially Observable Markov Decision Processes (POMDPs)

An MDP is very suited for decision making under full certainty of the full state. It is this assumption of full certainty of the current state however that can pose problems in real world settings where sensor signals can be noisy or otherwise inaccurate, effectively leaving the environment only partially observable. Deciding on actions directly based on noisy data can

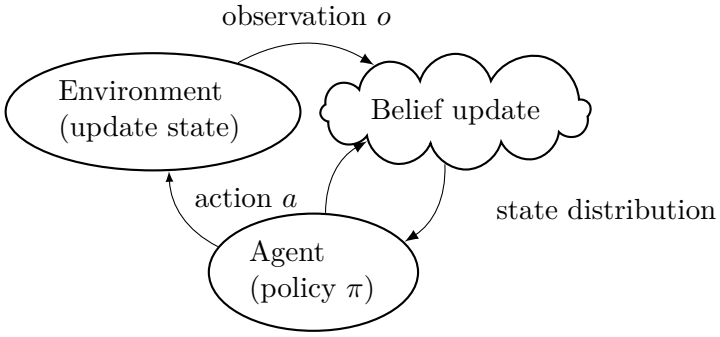


Figure 3-3: Belief State MDP.

have detrimental effects on the performance of an agent. To cope with uncertainty concerning the state, the Partially Observable Markov Decision Process (POMDP) [16] framework was devised. It is an extension of the (fully observable) MDP treated in the previous section that, instead of viewing the state s directly, now only obtains an *observation* $o \in \Omega$. This observation will most likely contain incomplete information, prompting the agent to reason about the uncertainty of occupying a certain state. It might also need to choose actions that forgo possible immediate reward for an increased likelihood of reward later on. The possible observations an agent can see are contained in a finite set Ω. As with the standard MDP, a POMDP can be described by the tuple:

Definition 3.2 (POMDP)**.** A POMDP is defined as $\langle S, A, \Omega, T, R, O \rangle$ where

- S, A, T and R are described by the definition of the standard MDP
- Ω is the discrete finite set of possible observations
- O is an observation function defined as $O : S \times A \times \Omega \to [0, 1]$
  Functionally denoted by the probability function $O(o, a, s') = Pr\{o|a, s'\}$ describing the probability that an agent will observe o after taking action a which resulted in reaching state s'

Figure 3-2 displays a diagram of the interaction between an agent and its environment in the POMDP framework.

As with the standard MDP, the aim of solving a POMDP is to create a policy that maximizes the total future reward. This is now more difficult however, since the agent does not know the state directly. It is usually unwise to blindly base any action decision on noisy or incomplete data. This means that a mapping directly from observations to actions is generally not preferable. By implementing a *belief state*, the agent can build a probability distribution over its possible current state out of multiple observations. The belief state is most often used to plan for POMDP problems by building a mapping from belief to action $\pi : b \to A$.

3-3-1 Belief State MDP

Remember that a POMDP is still a Markov process. This means that the problem is following the Markov property as explained in Section 3-2. In other words, we assume that the

current state is a sufficient summary of the past actions taken and states that were visited. Observations of the state however, do not adhere to the Markov property. This means we should find another way to represent past activity. We can do this by collecting all actions and observations during the process.. This information is often summarized into the belief state. Using a belief state in a POMDP is also described as converting it to a *belief state MDP* [2]. The interaction is graphically represented in Figure 3-3. The belief state itself is a vector $b_k(s)$ that forms a probability distribution over all states.

$$\begin{aligned} b_k(s) &= Pr\left\{s_k = s | o_k, a_{k-1}, o_{k-1}, ..., a_0, o_0\right\} \\ &= Pr\left\{s_k = s | o_k, a_{k-1}, b_{k-1}\right\} \end{aligned} \tag{3-17}$$

where

$$b_0(s) = Pr\left\{s_0 = s\right\}$$

is the initial belief, defining the probability that the system is in state s at time instant $k = 0$. The belief state is updated at every time step, using the previous belief and the most recent action and observation. This belief update is calculated by:

$$b(s') = \frac{Pr\left\{o|s', a\right\}}{Pr\left\{o|b, a\right\}} \sum_{s \in S} Pr\left\{s'|s, a\right\} b(s) \tag{3-18}$$

where $Pr\left\{s'|s, a\right\}$ and $Pr\left\{o|s', a\right\}$ are defined by T and O respectively, and

$$Pr\left\{o|b, a\right\} = \sum_{s' \in S} Pr\left\{o|s', a\right\} \sum_{s \in S} Pr\left\{s'|s, a\right\} b(s) \tag{3-19}$$

is a normalizing constant.

Creating a policy, based on the belief state $\pi(b) \rightarrow a$ requires the use of the value function. However, the expectation of future rewards in no longer only dependent on the action choice. It also relies on the likelihood of being in a certain state.
Incorporating the belief state into the value function (3-10), results in the expression:

$$V(b) = E^{\pi}\left[\sum_{k=0}^{h} \gamma^k R(b_k, \pi(b_k)) | b_0 = b\right]$$

where

$$R(b_k, \pi(b_k)) = \sum_{s \in S} R(s, \pi(b_k)) \cdot b_k(s)$$

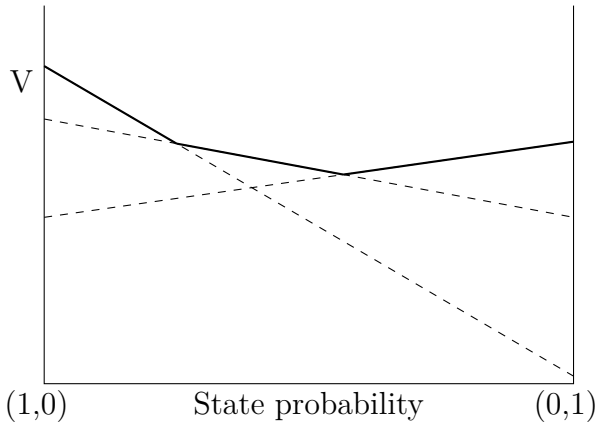


Figure 3-4: Piecewise linear convex value function for a two-state problem. The x-axis displays the belief space between two states ($s_1, state_2$). The y-axis displays the value of the belief.

denotes the reward the agent expects to receive after the next state transition,assuming the agent follows the policy π throughout the execution of the problem.

This value function is no longer defined on the state of the system, but on the belief over the state. Where the possible beliefs are contained in a continuous set of probability distributions over the state space S. Due to this aspect of continuity, we can no longer define the value over discrete instances. It has however been proven that the value function in this case is piece-wise linear and convex [18]. The value function can be comprised of a finite number of vectors on the belief simplex, indicating that the value of accurate knowledge is in general better than insecurity. An example of a value function defined this way is shown in Figure 3-4.

3-3-2 POMDP Planning

In order to plan for POMDP problems, we have to account for the added partial observability in the model. In addition to the POMDP specific value iteration algorithms, there are a two heuristic strategies that are often used to quickly build an approximate solution. By first solving the underlying MDP of the POMDP problem, we can obtain the fully observable policy $\pi_{MDP}(s)$ and its state action value function $Q_{MDP}(s, a)$. We combine the knowledge of optimal actions in every state with the state estimation built in the belief state.
A straightforward heuristic we can use is the *Most Likely State* policy π_{MLS} [19] where we take the state that has the highest probability and use it in combination with the MDP policy to pick an action.

$$\pi_{MLS}(b) = \arg\max_{a \in A} Q(\arg\max_{s \in S} b(s), a) \tag{3-20}$$

$$= \pi_{MDP}(\arg\max_{s \in S} b(s)) \tag{3-21}$$

This policy heuristic is clearly not optimal since it does not sufficiently take into account the inherent inaccuracy of partial observability. It does however provide an indication of the

accuracy of the belief state. This is due to the fact that the MLS policy reacts to the highest probable state and thus, this heuristic policy will perform better if the state estimation is more accurate.

Another heuristic, more often used in literature is the Q_{MDP} heuristic [20, 21].

$$\pi_{QMDP}(b) = \arg\max_{a \in A} \sum_{s \in S} b(s) \cdot Q_{\text{MDP}}(s, a) \tag{3-22}$$

The Q_{MDP} heuristic weighs the probability of of being in a certain state with the potential reward it can gain, if it were to depart from that state. This way the agent can choose an action based on the most reward it stands to gain, given a certain belief state. The Q_{MDP} heuristic can work effectively for some problem domains. It does not work for problems where the agent should collect information repeatedly due to the fact that the Q_{MDP} heuristic assumes that all state uncertainty disappears after taking a single action.

Value Iteration

To optimally plan for a POMDP we need to be able to compute the optimal value function for it. Using value iteration, we can iteratively estimate the value function, which converges to ϵ near the optimum in a finite number of iterations [22]. Similarly to value iteration for MDPs, we iteratively perform Bellman updates until the value function does not change any more. The Bellman update for POMDPs is defined as

$$V'(b) = \max_{a \in A} R(b, a) + \gamma \sum_{o \in \Omega} Pr\left\{o|b, a\right\} V(b') \tag{3-23}$$

$$\tag{3-24}$$

Where the value function itself is defined by a set of α-vectors $V = \langle \alpha_1, \ldots, \alpha_n \rangle$. This representation is graphically shown in Figure 3-5.
The value at a given belief is defined by the expression:

$$V(b) = \max_{\alpha \in V} b \cdot \alpha \tag{3-25}$$

with $b \cdot \alpha = \sum_{s \in S} b(s) \cdot \alpha(s)$ being an inner product.

We can adapt (3-14) to account for a partial observation of the state. This way we can compute the POMDP policy as:

$$\pi(b) = \arg\max_{a \in A} R(b, a) + \gamma \sum_{o \in \Omega} Pr\left\{o|b, a\right\} V(b) \tag{3-26}$$

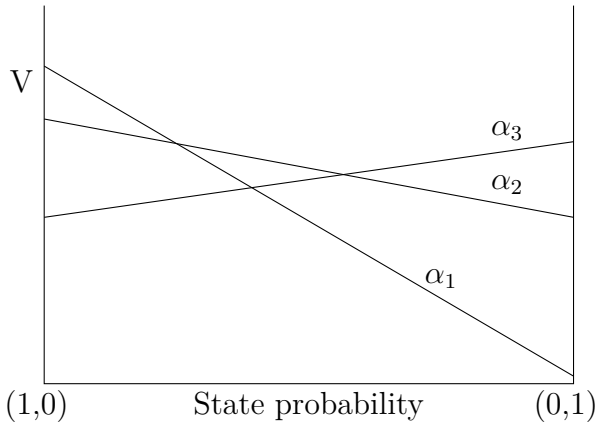


Figure 3-5: α-vector representation of the value of a two-state POMDP problem. The x-axis displays the belief space between two states $(s_1, state_2)$. The y-axis displays the value of the belief.

More often however, the α-vector representation is used directly to compute a policy. Each α-vector in the value function is labelled with the action that is associated with it. This means that the POMDP policy is given by:

$$\pi(b) = a(\arg\max_{\alpha_i \in V} b \cdot \alpha_i) \tag{3-27}$$

where each particular α_i represents a single action in the action space A. It should be noted that a particular α-vector representation does not necessarily contain an α-vector for each action, since it is possible that there exists an action in the action space that is never the optimal choice.

3-4 Multi-Agent Decision Process

The POMDP framework can be expanded in various ways to model problems with multiple agents. The most often used variants are the Dec-MDP and Dec-POMDP. Their aspects, similarities and differences are discussed in this section.

Both MDP and POMDP frameworks are concerned with decision problems for single agents acting in a stochastic environment. If we would want to tackle problems where multiple agents should cooperate however, we would need to expand the frameworks to do this. We need to be able to incorporate the interactions between agents, since building an optimal policy no longer depends purely on the environmental dynamics, but also on the other agents and in turn, their policies.
There are two notable frameworks that provide this expansion into multiple agent setting. These are the Decentralised Markov Decision Process (Dec-MDP) and Decentralized-Partially Observable Markov Decision Process (Dec-POMDP)[23]. In these frameworks, the agents work together to maximize a single globally defined reward function. None of the agents

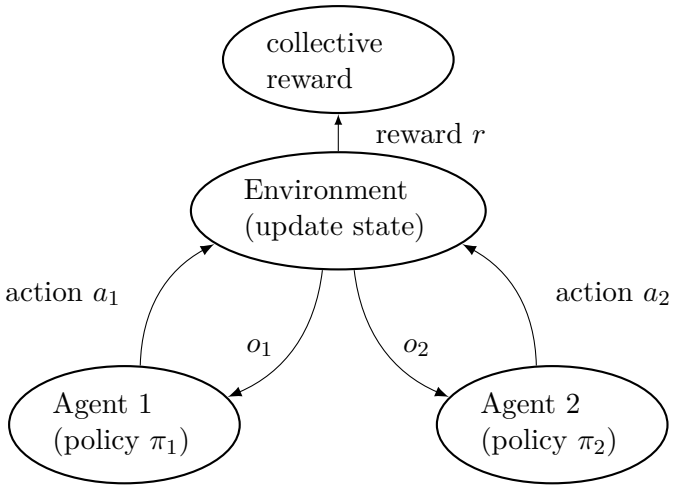


Figure 3-6: The interaction of two agents with their environment in a Dec-MDP or Dec-POMDP.

within a Dec-POMDP problem can explicitly communicate with another, which means they do not possess the ability to share their local observations in between time steps. This adds difficulty since the agents must cooperatively build a joint policy in such a way that the outcome is optimized, despite this lack of communication. The frameworks pose problems in which the individual agents must cooperate to maximize an expected cumulative reward. The individual agents must base their action decisions on *local information*, which they often only themselves can observe.
As with the MDP and POMDP, both the Dec-MDP and Dec-POMDP can be described by a single tuple, here given for the Dec-POMDP framework.

Definition 3.3 (Dec-POMDP)**.** A Dec-POMDP is defined as $\langle \mathcal{D}, S, \mathbf{A}, \mathbf{\Omega}, T, R, O \rangle$, where

- $\mathcal{D} = \langle 1, ..., n \rangle$ is the set of n agents
- S, T, O, R Are defined in the same way as with the POMDP
- $\mathbf{A}$ is set of possible joint actions
- $\mathbf{\Omega}$ is the set of possible joint observations

In the Dec-POMDP framework, the joint action space is defined as the combined set of the action spaces of every agent in the system $\mathbf{A} = \langle A_1, A_2, ..., A_n \rangle$ [2]. Similarly, the *joint observation set* encompasses the observation sets of all agents combined $\mathbf{\Omega} = \langle \Omega_1, \Omega_2, ..., \Omega_n \rangle$.

The difference between the Dec-POMDP and Dec-MDP frameworks lie in their assumptions on the observations of the agents, in particular the joint observation of the agents. For the Dec-MDP, we add the assumption that the combined observations of all agents, uniquely describe the complete state. In other words, should we enable perfect communication of local observations between all agents, they would know the full state of the system from these observations. This is not the case for Dec-POMDP problems however. Should the

[2]The notation used here, indicates the collection of action space of all agents.

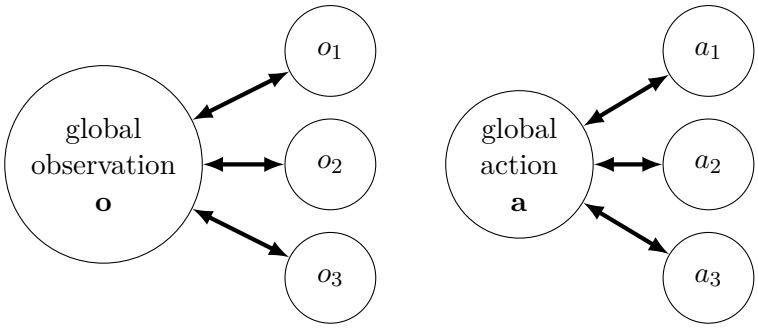


Figure 3-7: Global observation **o** and global action **a** deterministically relate to the combination of their local counterparts, local observations o_i and a_i.

agents communicate all local observations here, they would still be unsure of the complete system state. For the remainder of this chapter, we will mainly write about the Dec-POMDP framework and methods pertaining to it. These methods are just as valid for the Dec-MDP framework due to their close resemblance.

Figure 3-6 displays the interaction between the agents and their environment. This interaction cycles once per time step, which means that at every time step k, a joint action $\mathbf{a}_k = \langle a_1, a_2, ..., a_n \rangle_k$ is taken and a joint observation $\mathbf{o}_k = \langle o_1, o_2, ..., o_n \rangle_k$ is obtained. The state transition probability function depends on the joint action in this case, which adds the assumption that all agents will have chosen an action before collectively progressing to the next state. This is more accurately illustrated in Figure 3-7. Here we show that we can collect all local actions a_i into one global action **a** deterministically.

In the execution of a Dec-POMDP problem, the agents are not able to actually receive rewards or view the rewards they have obtained. Observing the immediate rewards would provide additional information regarding the true state which is not present in the received observations. The actual reward function is only used during a planning stage to form the policy. This means that we generally first find a policy offline after which we can execute it in a fully decentralized setting. At this planning stage, the point is to build a *joint policy* that maximizes the total expected cumulative reward. The joint policy is the combination of all local policies $\boldsymbol{\pi} = \langle \pi_1, \pi_2, ..., \pi_n \rangle$.

3-4-1 Histories

In the process of building a joint policy, planning for a Dec-POMDP, we find similar problems as with POMDP problems. A Dec-POMDP problem has similar issues with partial observability of its state. Problems within the POMDP framework usually deal with partial observability by determining actions on a belief state. Where the belief state can best be described as a summary of past observations and actions. Theoretically, the Dec-POMDP framework could also create a joint belief **b** by using knowledge of the state transition and observation models. However, this would require the knowledge of the joint actions and joint observation, which are not available to all agents during the execution of a real run.

If the agents are not able to see the full state, or build a belief state, there is little more left than to base the actions on the history of local actions and observations. We form the history

of the process from information found at each time step k, where the system is in a state s_k, observes a joint observation $\mathbf{o}_k$ and performs a joint action $\mathbf{a}_k$. The history of a system after h time steps will be a sequence of states, joint observations and joint actions in the form:

$$\langle s_0, \mathbf{o}_0, \mathbf{a}_0, s_1, \mathbf{o}_1, \mathbf{a}_1, ..., s_{h-1}, \mathbf{o}_{h-1} \rangle \tag{3-28}$$

where s_0 is the initial state and the initial joint observation $\mathbf{o}_0$ is assumed to be an empty joint observation: $\mathbf{o}_0 = \emptyset$.

Due to partial observability however, we are generally unable to observe the complete state at any given time step. An agent might also not be able to see the complete joint action being performed. Consequently, we might need to build the history in a different way to allow for these missing pieces of information.

We can identify three types of histories that are being used in Dec-POMDP problems.

The Action Observation History (AOH) for agent i, denoted by $\vec{\theta}_{i,k}$ is defined as the sequence of actions taken, combined with the sequence of observations received by agent i. At a time instant k

$$\vec{\theta}_{i,k} = \langle o_0, a_0, o_1, ..., a_{k-1}, o_k \rangle_i \tag{3-29}$$

Agent i's set of possible AOHs at time k is $\vec{\Theta}_{i,k} = \times^k (\Omega_i, A_i)$.

The Observation History (OH) for agent i, $\vec{o}_{i,k}$, is defined as the sequence of observations that the agent has received up until a time step k

$$\vec{o}_{i,k} = \langle o_0, o_1, ..., o_k \rangle_i \tag{3-30}$$

The set of observation histories for agent i at time k is denoted $\vec{\Omega}_{i,k} = \times_k \Omega_i$.

The Action History (AH) for agent i, $\vec{a}_{i,k}$, is defined as the sequence of actions an agent has performed at a time step k

$$\vec{a}_{i,k} = \langle a_0, a_1, ..., a_{k-1} \rangle_i \tag{3-31}$$

It is important to note that the history is only necessary due to the fact that the agents have no perception of the complete state. Should an agent perceive the full state at any time step, it can ignore its built up history prior to that point. due to the markov property, the current state is a sufficient summary of all past actions and observations.

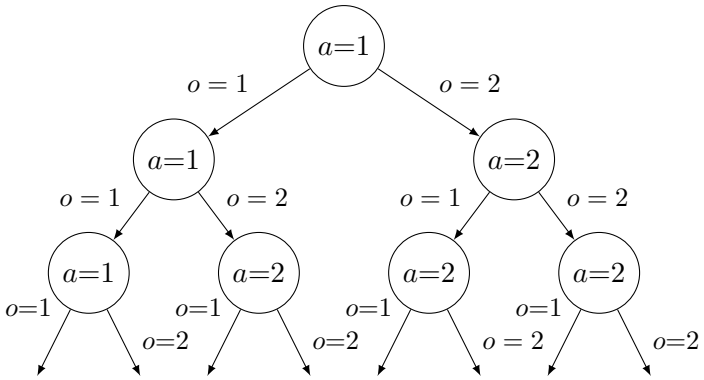


Figure 3-8: Policy tree for a single agent.

3-4-2 Policies

We can shape the policies for the Dec-POMDPs in multiple ways. Determining the action either deterministically or stochastically based on the local history. Most often, the policy is generated as a set of policy trees, one policy tree for each agent. En example of such a policy is shown in Figure 3-8.

If an agent i in a Dec-POMDP decides on its actions by a direct mapping from its observation history, its policy is called a *pure* or *deterministic policy*, π_i^p. The mapping $\pi_i^p : \vec{\Omega}_i \rightarrow A_i$ accounts for the observation action cycle of a single agent.

We define a *mixed policy* π_i^m for an agent i as a set of pure policies along with a probability distribution over this set. By doing this, the policy is will not always specify the same actions for the same situation.

Another way of obtaining this stochastic behavior is to use a *stochastic policy* π_i^s, which is a mapping from action-observation histories to probability distributions over actions. This mapping will take the form: $\pi_i^s : \vec{\Theta}_i \rightarrow Pr\{A_i\}$.

The decision to use a particular policy type depends greatly on the decision problem at hand and the desired performance. Most methods use a deterministic policy to avoid additional complexity associated with accounting for stochastic behavior.

3-4-3 Value Function

To determine whether or not a policy is improving its behavior or not, we refer to the value associated with that policy. The value for a joint policy $\boldsymbol{\pi}$ is defined as

$$V(\boldsymbol{\pi}) = E\left[\sum_{k=0}^{h-1} R(s_k, \mathbf{a}_k) | \boldsymbol{\pi}, Pr\{s_0\}\right] \tag{3-32}$$

Which is essentially the expected cumulative reward over the next h steps, given a global policy $\boldsymbol{\pi}$ and an initial state distribution $Pr\{s_0\}$.

Since we usually build the value over multiple runs of a decision problem, it might be best to build the value iteratively. This leads us to build the recursive expression

$$V(\boldsymbol{\pi}) = \sum_{s_0 \in S} V_{\boldsymbol{\pi}}(s_0, \vec{\theta} = \emptyset) Pr\{s_0\} \tag{3-33}$$

If the value of a certain policy cannot be computed recursively, it can be computed by averaging the obtained reward over a great number of trials. This is not favourable however, since it is very time consuming.

3-4-4 Related Frameworks

There are other multi-agent frameworks closely related to the Dec-POMDP that need mentioning. These frameworks are often mentioned alongside Dec-POMDPs in research papers due to their similar workings.

The Multi-Agent Markov Decision Process (MMDP) [24] is an extension of the standard MDP allowing for the interaction between multiple agents. At each time step, the agents perform an action based on the current state of the entire system. The actions of all agents combined is called the joint action $\mathbf{a}$. All agents have full observability of the current state, meaning that solving an MMDP can be done using existing MDP solvers.

The Partially Observable Stochastic Game (POSG) [25] is a game theoretic generalisation of the Dec-POMDP. The extension this framework provides is that the reward function is now defined separately for each agent. This means that agents in a POSG problem have the ability of being self-interested and aiming to purely maximize their own reward. This also enables the modelling of groups of agents that are competitive in nature. The downside to individual rewards is however that it is difficult to define jointly optimal policy. An optimal policy for this type of problem is most likely a Nash equilibrium (NE), however this could very likely be a local optimum. An NE is a policy equilibrium in which individual agents cannot unilaterally improve their performance by altering their policy.

3-4-5 Dynamic Programming for Dec-POMDPs

Due to it being a generalised version of the Dec-POMDP, any solution method for POSG problems can also be applied to Dec-POMDP problems. Such is the case with the DP solution approach [25]. To solve the Dec-POMDP problem through dynamic programming, the authors have generalised the DP operator used often in POMDP problems. This operator uses a two-step system during solving. It first performs a backup to create new policy trees and accompanying vectors. It then proceeds to prune those policy trees that are dominated, meaning their values are lower than all other policy trees. This is iteratively repeated until there is no more pruning possible. All steps taken during an iteration are shown in algorithm 1.

The DP approach starts at the end of the horizon, taking the final actions in the policy and backs them up. The increasing number of subtree policies created this way, will grow exponentially as the algorithm goes along. This is why dominated subtree policies are pruned. The *exhaustive backup* used in this method is a very computationally expensive operation, generating all $k+1$-step policies from a given set of k-step policies.

Algorithm 1 Dynamic Programming

Input: : Sets of depth-k policy trees Q_i^k and corresponding value vectors V_i^k for each agent i

Output: : Sets of depth-$k+1$ policy trees Q_i^{k+1} and corresponding value vectors V_i^{k+1} for each agent i

1: **for all** agents i **do**
2: Perform exhaustive backups to get Q_i^{k+1}
3: Recursively compute V_i^{k+1}
4: **repeat**
5: choose an agent i, and find a policy tree $q_j \in Q_i^{k+1}$ that satisfies: $\forall b \in \Delta(S \times Q_{-i}^{k+1}, \exists v_k \in V_i^{k+1} \setminus v_j$ such that $b \cdot v_k \geq b \cdot v_k$
6: $Q_i^{k+1} \leftarrow Q_i^{k+1} \setminus q_j$
7: $V_i^{k+1} \leftarrow V_i^{k+1} \setminus v_j$
8: **until** no more pruning is possible

3-4-6 Complexity

A major hardship in solving Dec-POMDP and even Dec-MDP problems, comes from the complexity associated with them. In an analysis of the complexity of a Dec-POMDP [23], the computational efforts of general MDP, POMDP and multi-agent generalisations problems were compared. This illustrated the inherent difference between a decentralized framework and a centralized one, essentially validating the intuitive need for a separate framework. The complexity of an MDP was previously proven to be P, whilst a POMDP is PSPACE-complete. This means that the problems can be solved using amounts of resources equal to polynomial functions of their size. Problems in the framework of a Dec-POMDP however are worst-case NEXP-complete, indicating that solving a Dec-POMDP problem will require a $2^{poly(n)}$ complex algorithm. In essence this means that solving a Dec-POMDP is quickly becomes intractable with increasing problem size.

3-5 Summary

This chapter provided an overview of the most notable stochastic decision process frameworks. The most basic form of these is the MDP where an agent is able to take actions at every time step. These actions change the state of the environment, and based on this altered state, the agent is given a reward. The goal of solving an MDP is to find an optimal policy that dictates the agent's actions in every state. Optimality is defined as the maximum expected cumulative reward the agent should obtain by adhering to the policy.
In the POMDP framework, the agent is no longer able to directly observe the system state and must build a probability distribution governing the probability of occupying a certain state. This distribution, called a belief-state, can then be used by the agent to base its action decisions on.

Multi-agent decision processes are used to model stochastic problems for multiple agents. Both the Dec-MDP and the Dec-POMDP provide general frameworks that are able to model multi-agent decision problems where the agents are only able to partially observe their envi-

ronment. The downside of these frameworks however are their complexity ratings, indicating the great computational effort required to solve such a decentralized problem.

Chapter 4

Centralized Solutions

The way in which we usually calculate decentralized policies is by building a set of local policy trees based on local observations and actions. This requires an extensive search over all possible policies, trading off the potential utility with potentially gained information[1]. All local policies also slowly build up a steady belief of the actions of other agents. After all, any local action is only as good as its cooperative counterpart performed by another agent. The focus in finding these local policies has been largely put on the decentral nature of the system. In other words, we mostly plan decentrally for decentralised problems. This has produced great, and often times optimal results. It does however require for the policy to be built from scratch without any prior information that may be helpful.

In this thesis, we wish to explore the use of prior information in the form of a policy that has been built on a global underlying version of the decentralized decision problem. We utilize this information by incorporating it into the belief update of an individual agent. This allows a single agent to reason about the actions of other agents in the decentralized problem. We obtain the global problem by first adding a few assumptions to the knowledge and capabilities of the agents. For instance, we can assume that the local agents in a decentralised decision problem have the ability to freely communicate with each other or have the ability to observe the current state. This way, we can reduce a decentralized problem into an MDP, MMDP or POMDP problem, which were previously defined in Chapter 3.

Of course these underlying problems differ from their decentralized counterparts. MDPs hardly handle partial observability well and both MDP and POMDP do not allow the agents to build a belief of the actions of other agents. They are useful however, for finding optimal actions in certain states or belief states. In addition, solving an MDP is computationally much easier to solve than a decentralised decision problem and there is a plethora of existing algorithms that can do this optimally.

This thesis researches the possibility of using these underlying models and resulting policies, to aid in building decentralized policies.

[1] We define utility as the expected reward gained by taking a particular action. It serves to illustrate the usefulness of a particular action.

4-1 Other Centralized Approaches

The notion of using the underlying models in decentralized planning is not new. It has been used in literature, particularly to aid communication decisions among agents [26, 27, 28]. These methods assume the agents have the ability to communicate to each other at execution time, however this communication is to be minimized. The underlying MDP problem is solved to find the optimal action in each state while communication serves to enhance local knowledge of the state to construct the global state from combined observations. The local belief is this scenario serves the purpose of finding specific instances where communication would enhance the decision process. If the local belief, combined with a global policy would result in a single action option, there would be no need to communicate. If the local agent has multiple potential actions to take, it can choose to communicate to other agents in order to gain more information and thus, make a better decision. These methods focus on aspects of communication while leaving the action decision policy to be determined by the underlying MDP. While this certainly helps the performance of a multi-agent decision problem, it does not adhere to our assumption that the agents cannot communicate.

The use of action decision heuristics in decentralized settings is also not new. The Q_{MDP} as previously explained in Chapter 3 has been used in research for decentralized policies [29]. In these cases however, the Q_{MDP} is used in finite-horizon MDP problems. Using this heuristic method in combination with an infinite horizon utility approach has not been done to the best of my knowledge.

4-2 Planning Centrally

In order to find a centralized solution, we reduce the decentralized problem. As previously mentioned, this requires us to add a few assumptions on the decentralized problem. In this thesis we focus on the two most common uses of decentralized decision problems, the Dec-MDP and the Dec-POMDP.

If we assume that in either of these problem types, the agents have full observability of the global state, we can eliminate the observation function and set from their respective tuples. Reducing their tuples:

$$\mathcal{D}, S, \mathbf{A}, \mathbf{\Omega}, T, R, O \rightarrow \mathcal{D}, S, \mathbf{A}, T, R$$

Which is an MMDP tuple. As previously mentioned, any MMDP problem can be solved using MDP solving methods, thanks to the full observability of the global state for each agent. It is worth noting that, although MMDP problems are of lower complexity, the underlying MMDP of a Dec-POMDP or Dec-MDP still grows exponentially with the number of agents.

If we were to assume full communication between all agents, this would have a different effect between the two decentralized decision frameworks. A Dec-MDP problem is by design, fully observable if we have perfect communication. This means that we can reconstruct the global state by combining the local observations of all agents. This means that the agents can

observe the full global state if they communicate their local observations. Again, having full observability of the global state will reduce any Dec-MDP to an MMDP.

Assuming full communication in a Dec-POMDP problem will provide different results however. By definition, the agents are not able to see the state of the system. Even if they communicate all their observations, they will only be able to see the global state partially. Communicating agents will however all have the same view of the system. This means that they can build a belief governing the global state and all act on that belief. This reduces the complexity of a Dec-POMDP to that of a POMDP [27]. One thing to note though, is that this simplification only holds if all agents start out at the same joint belief state.

We solve the underlying MDP/MMDP by using the existing *Value Iteration* algorithm. This algorithm may converge slightly slower than the conventional *Policy Iteration* algorithm, but is less influenced by the size of the state space which is still susceptible to take on great proportions when derived from a decentralized decision problem. The general outline of the value iteration algorithm is written is Section 3-2.
In the end, we obtain a deterministic policy $\pi : S \rightarrow A$ and the Q-value function $Q(s, a)$ associated with it .

In addition to solving the underlying MDP, we will solve the underlying POMDP. To do this we reduce the decentralized problem in a different way. Instead of assuming full observability of the state, we assume the agents merely communicate their observations during each time step. This means that we construct a problem where we combine all local observations into a single global one. This effectively changes the tuple we use to:

$$\mathcal{D}, S, \mathbf{A}, \mathbf{\Omega}, T, R, O \rightarrow S, A, \Omega, T, R, O$$

Similarly to the underlying MDP, we use an existing solver to solve the underlying POMDP and obtain an α-vector policy. Instead of using a normal Value Iteration algorithm to obtain this α-vector policy, we use an approximate method called Point Based Value Iteration (PBVI) [30]. The idea behind PBVI, is that we do not iterate over the full belief spectrum, but find a small reachable set for which we find an optimal value function. This reachable set can be obtained by applying the belief update rule iteratively during several test runs of the POMDP problem. We can build an extensive reachable set by picking a random action at each time step of simulation.

4-3 Building a Local Belief

We intend to try to improve on the belief of the local agents, by allowing them to reason about the actions and state occupation of the other agents. Using the global policy, we can build a local belief state for the agents. This local belief state does not only incorporate the system dynamics into its belief update, but also the assumption of the other agents' behavior. We first need to find an expression for the local belief update:

We start out by defining our belief for an agent i as:

$$b(s') = Pr\{s'|a_i, o_i\} \tag{4-1}$$

by applying Bayes' rule, we can write this as:

$$Pr\{s'|a_i, o_i\} = \frac{Pr\{o_i, a_i, s'\}}{Pr\{o_i, a_i\}} \tag{4-2}$$

The denominator term is a normalizing constant that can formulated in terms of the numerator. This means:

$$Pr\{o_i, a_i\} = \sum_{s' \in S} Pr\{o_i, a_i, s'\} \tag{4-3}$$

We will focus on expanding the numerator term further since the denominator term is directly linked to that. In order to build a proper update, we need to be able to incorporate the state transition function and the observation function. In order to add these, we need to reason about the global action $\mathbf{a}$ and the global observation $\mathbf{o}$. These are added by the law of total probability

$$\begin{aligned} Pr\{o_i, a_i, s'\} &= \sum_{\mathbf{o} \in \mathbf{\Omega}} Pr\{o_i, a_i, s', \mathbf{o}\} && (4\text{-}4)\\ &= \sum_{\mathbf{o} \in \mathbf{\Omega}} \sum_{\mathbf{a} \in \mathbf{A}} Pr\{o_i, a_i, s', \mathbf{o}, \mathbf{a}\} && (4\text{-}5)\\ &= \sum_{\mathbf{o} \in \mathbf{\Omega}} \sum_{\mathbf{a} \in \mathbf{A}} \sum_{s \in S} Pr\{o_i, a_i, s', \mathbf{o}, \mathbf{a}, s\} && (4\text{-}6) \end{aligned}$$

By use of the chain rule, we can expand this to:

$$\begin{aligned} &= \sum_{\mathbf{o} \in \mathbf{\Omega}} \sum_{\mathbf{a} \in \mathbf{A}} \sum_{s \in S} Pr\{o_i|a_i, s', \mathbf{o}, \mathbf{a}, s\} && (4\text{-}7)\\ &\quad \times Pr\{\mathbf{o}|a_i, s', \mathbf{a}, s\} && (4\text{-}8)\\ &\quad \times Pr\{s'|a_i, \mathbf{a}, s\} && (4\text{-}9)\\ &\quad \times Pr\{\mathbf{a}|a_i, s\} && (4\text{-}10)\\ &\quad \times Pr\{a_i, s\} && (4\text{-}11) \end{aligned}$$

We can reduce most of these terms citing sufficient statistics. For instance, term (4-7) can be reduced to

$$Pr\{o_i|\mathbf{o}\} \tag{4-12}$$

due to the fact that the local observation spaces Ω_i are subsets of the global observation space $\mathbf{\Omega}$. The local observations o_i are determined by the global observation $\mathbf{o}$ which in turn is determined by the other variables such as state. This dependence is expressed in term (4-8).

By noting that all information of the local action a_i is contained within the global action $\mathbf{a}$ and that the current state does not contribute to the observation by definition in Section 3-3. We can reduce term (4-8) to:

$$Pr\left\{\mathbf{o}|s',\mathbf{a}\right\} \tag{4-13}$$

This leaves the term to be perfectly described by the observation function O.

Term (4-9) is reduced by similar reasoning. All information of the local action is again contained within the global action, reducing the term to:

$$Pr\left\{s'|\mathbf{a},s\right\} \tag{4-14}$$

This term is now perfectly represented by the state transition function T.

The last term to be reduced, is term (4-11). This turns to:

$$Pr\left\{s\right\} \tag{4-15}$$

which is essentially the belief state $b(s)$. The probability over the state is not dependent on the action taken after it. This action is also known, eliminating the need of forming a probability distribution $Pr\left\{a_i\right\}$ over it.

After switching some sum terms, this renders the numerator part of the belief update to be:

$$\sum_{\mathbf{o}\in\mathbf{\Omega}} Pr\left\{o_i|\mathbf{o}\right\} \sum_{\mathbf{a}\in\mathbf{A}} Pr\left\{\mathbf{o}|s',\mathbf{a}\right\} \sum_{s\in S} Pr\left\{s'|\mathbf{a},s\right\} Pr\left\{\mathbf{a}|a_i,s\right\} b(s) \tag{4-16}$$

resulting in the full local belief update:

$$\frac{\sum_{\mathbf{o}\in\mathbf{\Omega}} Pr\left\{o_i|\mathbf{o}\right\} \sum_{\mathbf{a}\in\mathbf{A}} Pr\left\{\mathbf{o}|s',\mathbf{a}\right\} \sum_{s\in S} Pr\left\{s'|\mathbf{a},s\right\} Pr\left\{\mathbf{a}|a_i,s\right\} b(s)}{\sum_{s'\in S}\sum_{\mathbf{o}\in\mathbf{\Omega}} Pr\left\{o_i|\mathbf{o}\right\} \sum_{\mathbf{a}\in\mathbf{A}} Pr\left\{\mathbf{o}|s',\mathbf{a}\right\} \sum_{s\in S} Pr\left\{s'|\mathbf{a},s\right\} Pr\left\{\mathbf{a}|a_i,s\right\} b(s)} \tag{4-17}$$

At this point we can start to insert global knowledge into the system. There are two points at which we reason about the global counterparts of local information. Those being the global action $\mathbf{a}$, and the global observation $\mathbf{o}$. Reasoning about the observations of others might not be desirable however since these observations are stochastic by design. Instead we focus on the global action and the probabilities thereof.

To do this, we look at one term in the local belief update formula.

$$Pr\left\{\mathbf{a}|a_i,s\right\} \tag{4-18}$$

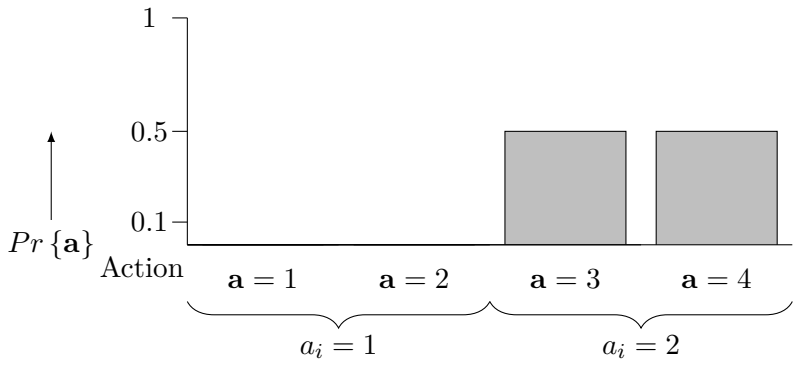


Figure 4-1: Probability of a global action given only a particular local action

fully local belief

If an agent i were to have no information about the possible actions or behavior of the others, this term would be reduced to

$$Pr\{\mathbf{a}|a_i\} \tag{4-19}$$

due to the fact that state information is only effective in determining the actions of other agents. We call a belief update with this action reasoning, an *ignorant local belief update*. This corresponds to finding the possible global actions that can result from one local contribution and equally distributing the probability of those taking place. An example for a local agent i is given in Figure 4-1. In this example we imagine a two-agent problem where each agent can perform one of two possible actions. The combination of two local actions determine the global action. If the local agent has used action 2, this corresponds to two possible global actions which are equally probable.

local belief with added policy information

We could also reason over the actions of the other agents by using the state-action value function which was computed as a global solution. If we were to have a single agent i reason about the actions of other agents by means of the Q-function, we can incorporate this into the global action probability. In this case, we deduce a probability from the Q-function by normalizing the expected rewards within and deciding that an action with a higher reward is more likely to take place than an action with a lower value. We first assume that instances of a_i and s are independent. This way we can write.

$$Pr\{\mathbf{a}|a_i, s\} = Pr\{\mathbf{a}|a_i\} \cdot Pr\{\mathbf{a}|s\} \tag{4-20}$$

where $Pr\{\mathbf{a}|a_i\}$ is the probability of $\mathbf{a}$ occurring, given only a_i and $Pr\{\mathbf{a}|s\}$ is the probability of $\mathbf{a}$ given s. Using the Q-function, we define:

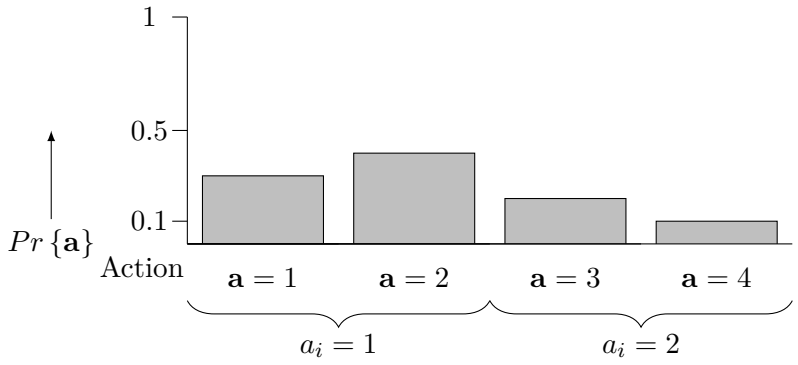


Figure 4-2: Probability of a global action given state information and global values at those states

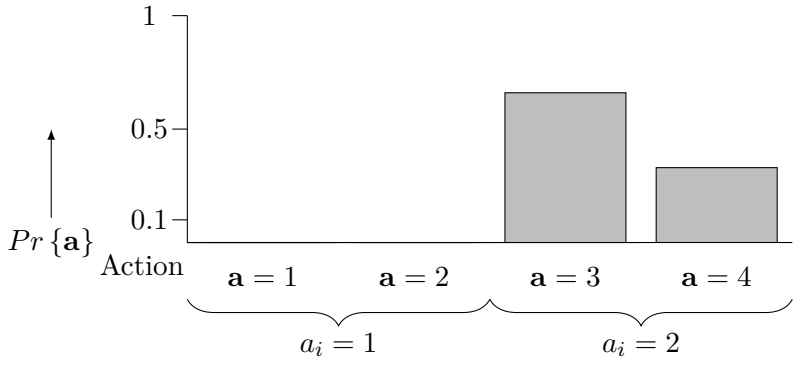


Figure 4-3: (Normalized) probability of a global action given both local action and state information

$$Pr\{\mathbf{a}|s\} = \frac{Q(s,\mathbf{a})}{\sum_{\mathbf{a}\in\mathbf{A}} Q(s,\mathbf{a})} \tag{4-21}$$

4-4 Local Decision Making

Using the solutions to the underlying MDP problem, we try to find local actions for the agents to act independently in the system. In essence, we are trying to convert aspects of the underlying policy to make them useful in a decentralized setting. The most straightforward heuristics we can test are those also used in POMDP problems.
The MLS and Q_{MDP} heuristics can be used to pick an action from the local belief state. Since the local belief state reasons over the same states as a normal belief state, we deem it trivial to adapt these for a local agent i. In essence we seek the best global action to take using these heuristics, then identifying the local component corresponding to agent i. In case of the MLS policy we seek the global action as such

$$\mathbf{a} = \arg\max_{\mathbf{a}\in\mathbf{A}} Q(\arg\max_{s\in S} b(s), \mathbf{a}) \tag{4-22}$$

Then identifying the local action a_i corresponding to the found global action $\mathbf{a}$.
Similarly, we use the Q_{MDP} heuristic to identify the local action for agent i by first picking the best global action

$$\mathbf{a} = \arg\max_{\mathbf{a}\in\mathbf{A}} \sum_{s\in S} b(s) \cdot Q_{\text{MDP}}(s,\mathbf{a}) \tag{4-23}$$

As shown in Section 3-3, these heuristics stem from the field of POMDP where the only possible underlying model is the MDP. In the multi-agent scenario, we also have the option of using the underlying POMDP model and its solution to aid in decentral decision making. We use the α-vector representation resulting from solving a POMDP problem. Again, we use the belief of the current state to pick a global action $\mathbf{a}$ from which we can deduce the corresponding local action a_i. This is done by altering 3-27 to account for global actions.

$$\mathbf{a} = \mathbf{a}(\arg\max_{\alpha_i\in V} b \cdot \alpha_i) \tag{4-24}$$

Here we essentially find the best global action $\mathbf{a}$ corresponding to a certain current belief b. Again. this global actions uniquely corresponds to a local action a_i for each agent i.

4-5 Evaluating Decentralized Policies

Since the local policies, derived from a central policy, are not directly tied to a value function, we can only build an expected value by taking the results of multiple test runs.

The nature of the policies is based on infinite horizon execution. Thus we try to build an expected value from any starting point, taking into account all possible future outcomes.

We compare the results in two ways

$$E\left[\sum_{k=1}^{h} r^k\right] \tag{4-25}$$

where r^k is the reward obtained at time step k.

$$E\left[\sum_{k=1}^{\infty} \gamma^k r^k\right] \tag{4-26}$$

In order to actually compute the latter expected reward, we will simply choose a horizon long enough to find an expectation that does not change if we further increased it, i.e. we expect the discounted reward to reach a limit which will indicate its infinite horizon discounted value. We also expect the extensive nature of the belief update to have an effect on the simulation time of a given number of time steps. The local belief update needs to take into account multiple different possibilities in order to make an informed state estimate.

4-6 Summary

In this chapter has shown how we intend to utilize the global underlying policy to aid in executing a decentralized problem. There currently do exist methods that make use of this same idea, these however mostly focus on adding communication. The reasoning behind of this is that the agents would communicate only when they were unsure of the current state, leading to a separate policy being formed to tie down communication rules.

In order to build global policy, we reduce the decentralized problem to a form of MDP by assuming full perfect communication between all agents and full observability of the state. We solve this reduced form of the problem to obtain the global policy and its state-action value function.

We make use of the global policy in two ways. We use it to alter the local belief update for an individual agent and we also use it to base local actions on. The local belief update contains a term that allows the agent to reason about the actions taken by other agents. We use this term in combination with the value function to build a distribution over possible actions that were taken.

In order for local agents to be able to base their actions on the altered local belief, we make use of heuristics. These heuristics originate from their use in POMDP frameworks where we

also use them to forge a link between the fully observable MDP and POMDP. We set out to evaluate the utility of this method by calculating the expected value for a given decentralized stochastic decision problem. There is however, no direct link we can put between this method and a corresponding value. This means we have to evaluate by performing multiple runs of the problem and using the resulting values to construct a value.

Chapter 5

Simulations and Experimental Results

In this chapter, we compare the results obtained from simulations on benchmark problems with results from other algorithms and approaches.

5-1 Benchmark Problems

In order to test the effect of the our approach on a decentralized problem, we test it on a few benchmark problems found in literature. The problems we use to test are the 'recycling robots' problem, the 'broadcast channel' problem and the 'meeting on a grid' problem. In this section we will briefly explain the thought process behind these benchmark problems. The exact state transition probabilities are defined in so-called '.dpomdp' files that can be found at a Dec-POMDP problem domain website [1]. We chose these benchmarks because they do not require repeated information gathering. This is desirable since the Q_{MDP} heuristic is unable to handle these types of problems.

We compare the performance of our approach with existing optimal performance. We also compare to the performance of the heuristic action decision method in combination with other belief update rules. These other update rules include instances where the local agents have artificial access to all observations or actions. We do this to allow ourselves more insight into the effect of our approach.

Recycling Robots Problem

The recycling robot problem [31] revolves around the two robots trying to clean up an unspecified environment. The agent set in this problem is defined by $\mathcal{D} = \langle robot_1, robot_2 \rangle$. Each robot has the option of taking one of three high-level actions. They can choose to pick up a small can, pick up a large can or recharge their battery. $A_{robot_1} = A_{robot_2} = \langle recharge, pickup\_small, pickup\_large \rangle$. If a robot chooses to pick up a small can, it will

[1] http://users.isr.ist.utl.pt/~mtjspaan/decpomdp/index_en.html

receive a small reward, if it chooses to recharge, it will receive nothing. A robot is unable to pick up a large can by itself, thus it requires the other robot to help. The robots will only receive a reward for picking up a large can if they do so simultaneously. The state space of this system is comprised of the battery levels of the robots which is discretized into two values. A 'high' and 'low' battery value for each robot resulting in the state space description $S = \langle high\_high, high\_low, low\_high, low\_low \rangle$. Picking up a small can will lower the battery state with a probability of 0.7. Otherwise, the battery state will remain the same. Trying to pick up a large can will have the same effect, only then, probability that the battery state is reduced is 0.5. If any one of the two robots has a low battery state and further depletes it, a strong negative reward is issued. Both robots can perfectly observe their own battery state, they cannot however observe the battery state of each other. This makes it difficult for the robots to appropriately determine when they should best try to pick up the large can. Summarizing, the Recycling robots problem is a problem with 4 states, 9 actions and 4 observations.

Broadcast Channel Problem

In the broadcast channel problem [32], we look at the interaction between two agents who need to broadcast messages over a single channel. Every time step the agents can send a message over the channel, however only one message can be sent over the channel at a time, otherwise the messages collide. To avoid this, the agents to need to coordinate their actions in such a way that they are able to send their messages with minimal chance of collision. The agents themselves can take two possible actions. They can either *send* a message or *wait* and do nothing. If an agent successfully sends a message, it receives a reward of 1. If the agent waits, or unsuccessfully sends a message, it receives no reward. At the end of a time step, both agents receive a stochastic observation of whether or not a message was successfully sent or not. Both agents have a message queue to pick their messages from. The most basic version of the broadcast problem only has a queue length of 1. After each time step, the queue has a probability p_i of having a message added to it per agent. This probability differs per agent since it would be trivial to build a policy if they had the same probability of receiving a message (basically taking turns sending and waiting). The standard 2-agent problem uses $p_1 = 0.9$ and $p_1 = 0.1$. Overall, the system has 4 states, 4 observations and 4 actions where the observations are 2 per agent, as are the actions per agent.

Meeting on a Grid

Another benchmark problem is the 'meeting on a grid' problem where we place two agents in a square environment subdivided in a pattern of 3x3 squares. [33, 34]. The agents can navigate the environment by taking small steps every time step in the directions 'up', 'down', 'right' or 'left'. They can also opt to stay in the same spot. Any attempted move will only succeed with limited probability, failing the actions results in moving to another spot than intended or standing still. An attempt to move through walls will result in the agent in staying in the same square. The robots have the ability to occupy the same square simultaneously but cannot sense each others' presence. The agents can only make out their own position in the grid. The goal of the robots is to find each other in one of two spots on the grid, either in the top left

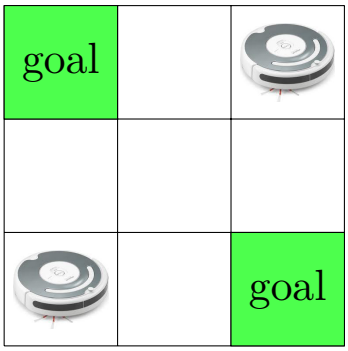


Figure 5-1: meeting on a grid problem. Two agents need to find each other in one of two goal squares to receive a reward.

or bottom right corner. In other words, the agents need to occupy the same specific square in order to receive a reward. The agents obtain a reward of 1 for each time step they stay in the same spot. Any other configuration of positions results in a reward of 0. The difficulty in coordinating their actions lies in the stochastic nature of action results and the inability for the agents to know about the location of each other. The problem is initialised with both agents starting in opposite ends of the field. This initial setup is graphically represented in Figure 5-1 This leaves the problem with a state space of 81 states. The total number of actions is 25, comprised of five possible actions per agent. The total number of observations is 81, comprised of nine observations per agent.

5-2 Global Policies

As stated in the previous chapter, we first need to find the global policy for the proposed problems. This is easily done by using an existing MDP value iteration algorithm [2]. During the optimization of the problems we have set the discount factor to 0.9 for all problems. These global policies are used later on to define actions decisions for the individual agents in each problem.

The global policy for the recycling robot problem very accurately displays the difficulty in planning actions for the problem. This is depicted in Table 5-1. It is most advantageous for the robots to pick up a large can if they both have a full battery. In the decentralized case however, they do not know the battery state of the other robot at any time. This requires them to reason about the battery state of the other to be able to choose whether or not to pick up a large can. We can also already see an advantage in integrating the global policy into a local one. The individual robots no longer need to reason over multiple actions if their own battery is low, since the optimal action in any state combination is to recharge its battery. Associated with this policy, we also have the Q-function which we will be using in combination with the local beliefs of the agents. This Q-function is described in full in Table 5-2.

[2]The MDP toolbox that contains this algorithm can be found at http://www.mathworks.nl/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox

Table 5-1: Optimal global policy π for recycling robot problem. The robots will only require knowledge of one another if they have a full battery themselves. If their own battery is empty, the only optimal action is to recharge.

Agents #1 \ #2	Battery Full	Battery Empty
Battery Full	$a_1 = 3$, $a_2 = 3$	$a_1 = 2$, $a_2 = 1$
Battery Empty	$a_1 = 1$, $a_2 = 2$	$a_1 = 1$, $a_2 = 1$

actions: **1**=recharge, **2**=search small can, **3**=search large can

Table 5-2: $Q_{\text{recycling robots}}(s, \mathbf{a})$. The highest values for a state correspond to the optimal action in that state.

State	$\mathbf{a} = 1$	$\mathbf{a} = 2$	$\mathbf{a} = 3$	$\mathbf{a} = 4$	$\mathbf{a} = 5$	$\mathbf{a} = 6$	$\mathbf{a} = 7$	$\mathbf{a} = 8$	$\mathbf{a} = 9$
$s = 1$	30.463	31.951	29.609	31.951	33.472	31.153	29.609	31.153	33.848
$s = 2$	30.463	28.697	26.268	31.951	29.873	27.233	29.609	27.991	29.043
$s = 3$	30.463	31.951	29.609	28.697	29.873	27.991	26.268	27.233	29.043
$s = 3$	30.463	28.697	26.268	28.697	26.527	24.228	26.268	24.228	24.703

States: 1= full_full , 2=full_empty , 3=empty_full , 4=empty_empty
The global action numbers here, are based on a ternary numbering system using the local action numbers in Table 5-1. This means for example: $\mathbf{a}$=1 $\rightarrow$ a_1=1, a_2=1 and $\mathbf{a}$=4 $\rightarrow$ a_1=2, a_2=1.

The global policy associated with the broadcast problem is described in Table 5-3. Similarly to the global policy for the recycling robot problem, we find some basic rules that local agents can always adhere to. In this case, agent #1 will always send a message if it has one, no matter the message state of agent #2. This is due to the fact that agent #1 is much more likely to have a message in queue than agent #2 and would thus more likely be able to safely send it. The global policy also defines actions for the state in which neither agent has a message in queue. This is of no effect to the performance of the agents since they cannot clash any message they cannot send.The Q-Function of the broadcast problem is shown in Table 5-4.

Table 5-3: Optimal global policy π for broadcast channel problem. Agent #1 will always send its message if it has one in queue. What happens when neither agent has a message in queue is of no relevance since there can be no clash.

Agents #1 \ #2	Message in local queue	No message in local queue
Message in queue	$a_1 = 1$ / $a_2 = 2$	$a_1 = 1$ / $a_2 = 2$
No message in queue	$a_1 = 2$ / $a_2 = 1$	$a_1 = 1$ / $a_2 = 1$

actions: **1**=send message, **2**=wait

Table 5-4: $Q_{\text{broadcast channel}}(s, \mathbf{a})$

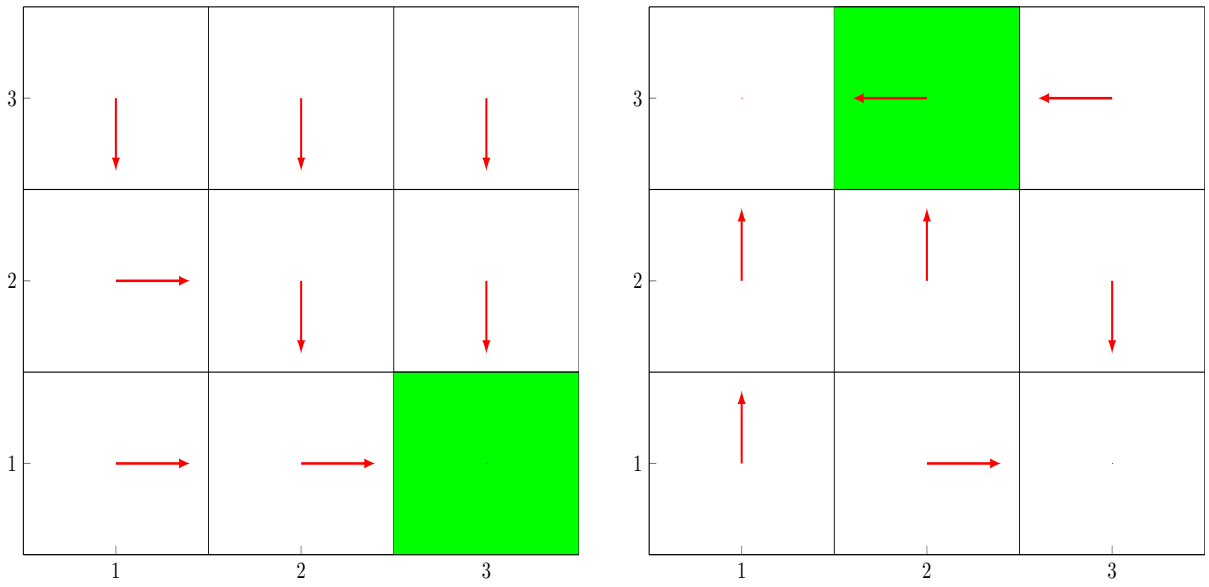
State	$\mathbf{a} = 1$	$\mathbf{a} = 2$	$\mathbf{a} = 3$	$\mathbf{a} = 4$
$s = 1$	8.4321	8.4321	8.4321	8.4321
$s = 2$	8.4321	8.7310	9.4321	8.7310
$s = 3$	8.4321	9.4321	8.5158	8.5158
$s = 4$	8.4321	9.7310	9.5158	8.7579

where $\mathbf{A} = \langle 1$=send_send, 2=send_wait, 3=wait_send, 4=wait_wait$\rangle$
and $S =$
$\langle 1$=message_message, 2=message_nomessage, 3=nomessage_message, 4=nomessage_nomessage$\rangle$

The global policy for the meeting on a grid problem is quite extensive since it defines global actions for each state and the state space is vastly larger than the other two problems. In fact it has 81 possible states whereas the other two problems only have 4. In essence, the global policy dictates that the agents should move to the goal space nearest to them. If one agent is already at a goal space, the other should move to meet it there, as can be seen in Figure 5-2a. This is an example of avoiding equilibrium conditions. Suppose both agents are at opposite goal spots, one of the two should move to the other goal space in order to receive a reward. However in it will not matter which agent stays and which one moves since the total reward remains the same. This introduces a choice as to which agent should move and which one should stay. This choice has now been made by solving the underlying problem, easing the actual decentralized decision making later on when such a situation occurs.

5-3 Simulation Results

The data shown in this chapter is the result of multiple runs of each benchmark problem, varying over belief update type and method of action selection. The expected value of a particular combination is the average over 150 runs. The simulations were run over a period of 100 time steps. In the case of the broadcast channel problem, this was extended to 900 time steps to be able to compare the results to existing finite horizon optimal results.



(a) Agent #2 should move to #1, even if it is already in a goal spot.

(b) Both agents should move to the nearest goal spot on the grid.

Figure 5-2: Parts of the global policy for the meeting on a grid problem. If agent #1 occupies the color filled square, the optimal actions for agent #2 are defined by the arrows for each state from which the arrow originates. The lack of an arrow in a square indicates that the optimal action is to not move.

The different belief update types we evaluated are:

- *local belief + policy*, where each agent performs the local belief update defined in the previous chapter. The belief update includes the global policy information defined by the global policy.
- *local belief*, where each agent performs the local belief update without any knowledge of the global policy.
- *local belief + global action*, where we allow all individual agents to perfectly know the actions performed by all other agents during the last step.
- *local belief + global observation*, where we allow all individual agents to perfectly know the observations seen by all other agents.
- *global belief*, where all agents know all actions and observations of the others. This is essentially how the underlying POMDP behaves and thus will use the POMDP belief update defined in Chapter 3.
- *global knowledge*, where all agents have access to the full state. This is essentially how the underlying MDP behaves.

The belief updates for the 'local belief + global action' and 'local belief + global observation' are completely described in Appendices A and B respectively.

The performance of our approach is evaluated in two ways. First we build the infinite horizon value of the policies. This is done by calculating the discounted reward over the full length of a simulation run and averaging them over all runs. We assume the this averaged discounted reward adequately represents the infinite horizon utility. Which is reinforced by the fact that the discounted reward will asymptotically approach its limit.

The other way in which we view our results is by limiting the number of time steps. Even though our approach is mainly directed at the infinite horizon case, we also use its gained reward at any time instant and compare it to existing optimal results from finite horizon approaches. This way we can see if our approach is performs comparably to finite horizon policies, which can be proven to be optimal. These comparisons forgo the use of discounting to more accurately view the progress in gaining rewards.

5-3-1 Infinite Horizon Utility

The infinite horizon utility represents the expected discounted value that the agents could obtain in their foreseeable future. This number is a representation of the performance of the agents when we do not specifically plan ahead for a set number of time steps. In the calculation of these infinite horizon values, we used a standard discount value of γ=0.9. We compare the behavior of three action decision heuristics in light of varying states of knowledge about the current global state. This way we can identify which factors play a role in the action decision process and of what importance these are.

Since we cannot actually measure an infinite number of rewards over time, we calculate the infinite horizon utility using a limited number of reward measurements. The infinite horizon utility is calculated as the expected discounted reward to be obtained over an infinite number of steps. In order for this number to be finite, it should reach a limit value asymptotically. We can ascertain this asymptotical approach to a limit value from a finite number of time steps and determine by extension the infinite horizon utility.

These benchmark problems have already been used in relation to infinite horizon policy methods, meaning we can compare existing results in this field with ours. The highest known discounted infinite horizon value for the recycling robot problem is 31.92865 [31].
The highest known discounted infinite horizon value for the broadcast problem is 9.1 [31]. The meeting on a grid problem does not yet have an expected infinite horizon utility to compare to. [3]

Recycling Robots Problem

Applying a local belief method in combination with action decision heuristics to the recycling robots problem will result in a very good performance when compared to the currently known best. Which is, as stated, currently at an expected value of about 31.9. Accurate values of the infinite horizon utility obtained for this problem are depicted in Table 5-5.

There are a few notable occurrences happening in the system however. For instance, the performance of the system under all action decision types can be clearly split into a couple

[3] In instances where the paper was not accurate enough, the value was obtained from http://rbr.cs.umass.edu/camato/decpomdp/download.html

categories. Any policy that relies on a belief with full information on the global observation will outperform most others. This is due to that fact that the recycling robots problem is fully observable once all local observations can be combined. In other words, the recycling robots problem is a Dec-MDP problem.

Another notable occurrence is the seemingly worse performance of both Q_{MDP} and α-vector policies under a belief in which all actions can be seen. The reason this is happening is because the robots will more often charge simultaneously to ensure a full battery for both in the next time step. This next step, the robots can cooperatively pick up a large can to receive a large reward. Charging however does not bring with it any reward and thus both robots miss the opportunity to gain extra reward.

The addition of policy information to the belief state does not seem affect the performance of the system under any action decision heuristic. This most likely means that the policy information provided during the belief update does not add enough certainty to change action decisions in similar situations. On the other side, it also does not alter the behavior negatively so as to reduce performance of the system.

As stated before, the expected values calculated for the infinite horizon utility are done so using a finite number of steps and finding the limit value to which the system values go. This limiting behavior is shown in Figure 5-3 for each belief update, and action heuristic combination.

Table 5-5: Infinite Horizon utility of the MLS,Q_{MDP}, and α-vector action decision heuristics on the recycling robots benchmark problem while varying the information content fed to the individual agents. Adding global policy information seems to have no added effect on the performance.

Belief Type	**MLS**	**QMDP**	**α-vector**
local belief + policy	22.0625	30.737	30.8338
local belief	21.2149	30.7886	31.0509
local belief + global action	23.0378	24.2295	25.4078
local belief + global observation	32.5229	32.9455	30.5005
global belief	32.6472	33.3284	29.486
global knowledge	32.8396	32.5739	31.3307

Broadcast Channel Problem

The results obtained when simulating the broadcast channel problem closely resemble those obtained for the recycling robots problem. The expected infinite horizon utility lies very close to the currently know highest value of 9.1. Accurate values of the infinite horizon utility obtained for this problem are depicted in Table 5-6

Even more so than with the recycling robots problem, the expected infinite horizon values for the broadcast channel problem lie very close together. Even the difference between the MLS and other action heuristics has become very small.

The only combinations of belief and action heuristic to properly differentiate themselves from the rest are those that involve global knowledge, which is understandable since using this

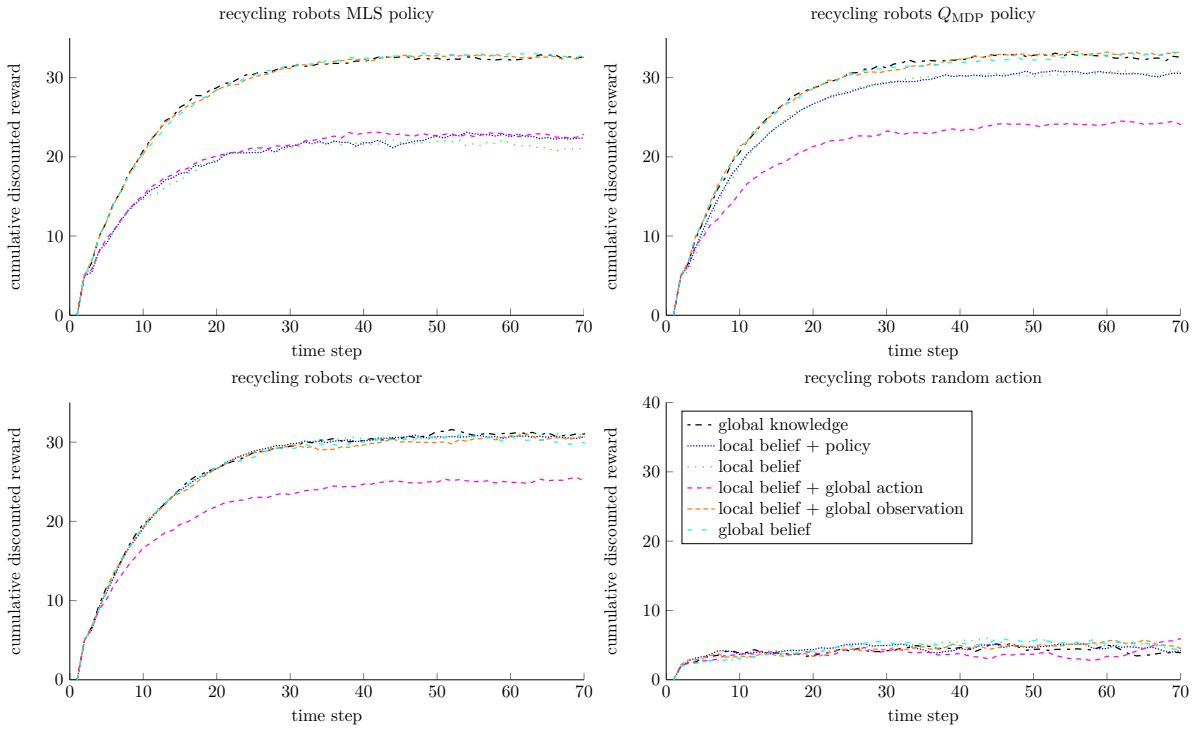


Figure 5-3: Discounted reward for the recycling robots problem over time. The expected discounted reward build up over time shows asymptotic approach to a limit.

belief type is essentially the same as simplifying the decentral problem to an MDP problem. In this more basic form, the agents have basically coordinated their actions perfectly.

Again we see that adding global policy information to the local belief will not improve the behavior in any noticeable way. However, for this problem the reason behind this is more obvious. We might not be able to reasonably expect an agent operating on a local belief with global policy information to outperform one operation on a global belief. In other words, we can expect the global belief to serve as an upper bound on the performance.
Figure 5-4 shows that we can safely say these values will suffice as an infinite horizon utility indication. The expected value has stayed the same over a long period of time.

Table 5-6: Infinite Horizon utility of the MLS,Q_{MDP}, and α-vector action decision heuristics on the broadcast channel Benchmark Problem while varying the information content fed to the individual agents. There is little to no effect if we add (perfect) information to the belief update.

Belief Type	**MLS**	**QMDP**	**α-vector**
local belief + policy	8.93235	8.97647	9.05809
local belief	8.98791	9.06342	9.03451
local belief + global action	8.96727	9.01656	9.00001
local belief + global observation	8.96582	8.99466	8.99371
global belief	8.99888	9.0031	9.02885
global knowledge	9.57903	9.47377	8.95733

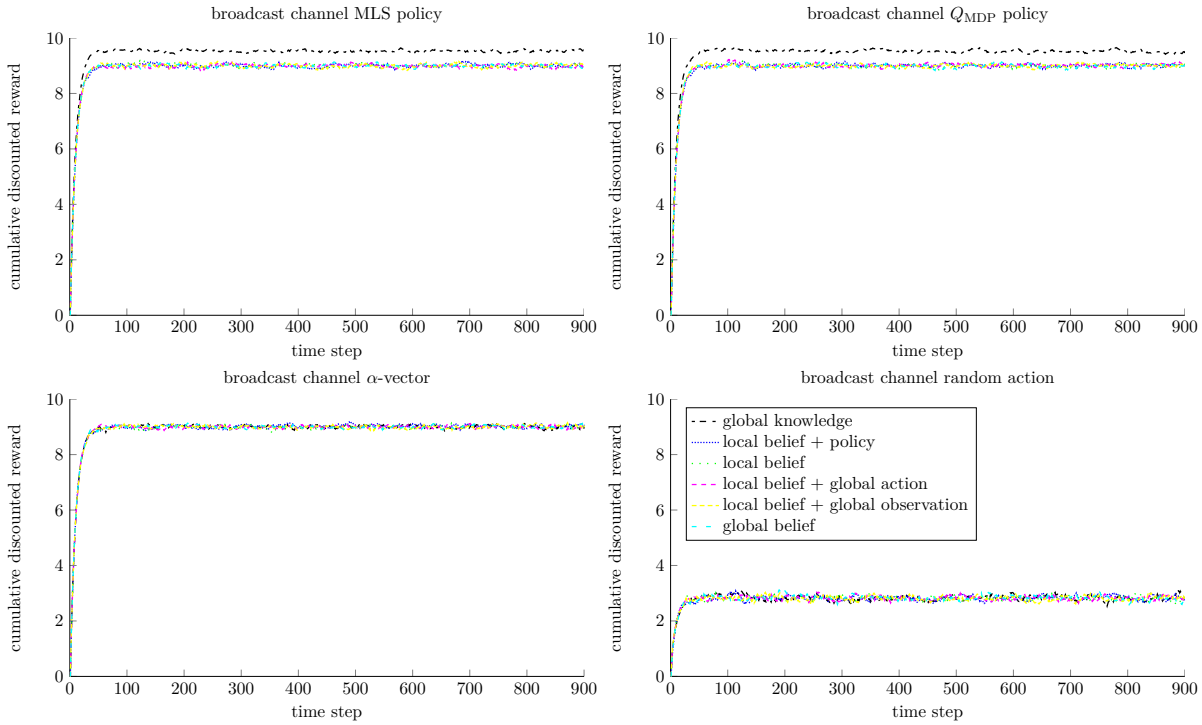


Figure 5-4: Discounted reward for the broadcast channel problem over time. The expected discounted reward build up over time shows asymptotic approach to a limit.

Meeting on a Grid

Compared to the other two benchmark problems, we see a little more variety in results with the meeting on a grid benchmark. There are distinct stages visible in the results that are attributable to the information content fed to the agents.

First off, looking at the MLS policy heuristic, we can clearly see that all forms of belief where the agent is given more information, will result in an improved performance. The belief types where any kind of accurate global information is provided show the best behavior. This would indicate that knowing the action of another agent would be just as advantageous as knowing their observation. We can also see that the inclusion of global policy information to the belief of a local agent will improve the behavior of the system.

Direction our attention to the results involving the Q_{MDP} action decision heuristic. We can identify three distinct categories of utility. The first of which is composed of the two fully local belief types. Using the Q_{MDP} seems to negate any information added to the system, opting to generalise over values of the possibilities. The second category is composed of only the local belief with perfect knowledge of the global action. It shows an improvement over the fully local belief types, indicating that knowledge of the global action can indeed be used (more) effectively to improve performance. The third category shows the best performance. Allowing use of the global observation in this case will result in the best results. This is due to the fact that again, this problem is a Dec-MDP problem, which is fully observable if all local observations are combined.

All belief types perform oddly in combination with the α-vector heuristic. This is most likely

caused by the way in which the α-vector policy is constructed. This is done by sampling the belief space of the agents. This sampled set may not include any high probability beliefs such as perfect knowledge of the state. This discrepancy between sampled set and attainable set results in an altered behavior of the system.

Table 5-7: Infinite Horizon utility of the MLS,Q_{MDP}, and α-vector action decision heuristics on the meeting on a grid benchmark problem while varying the information content fed to the individual agents. Adding information about the global policy affects the performance of the agents.

Belief Type	**MLS**	**QMDP**	**α-vector**
local belief + policy	7.6813	6.75105	7.73261
local belief	7.91842	5.48652	7.91868
local belief + global action	9.93741	7.84034	7.82598
local belief + global observation	9.95044	9.95293	6.58839
global belief	9.94755	9.95878	6.59114
global knowledge	9.95359	9.95443	6.62977

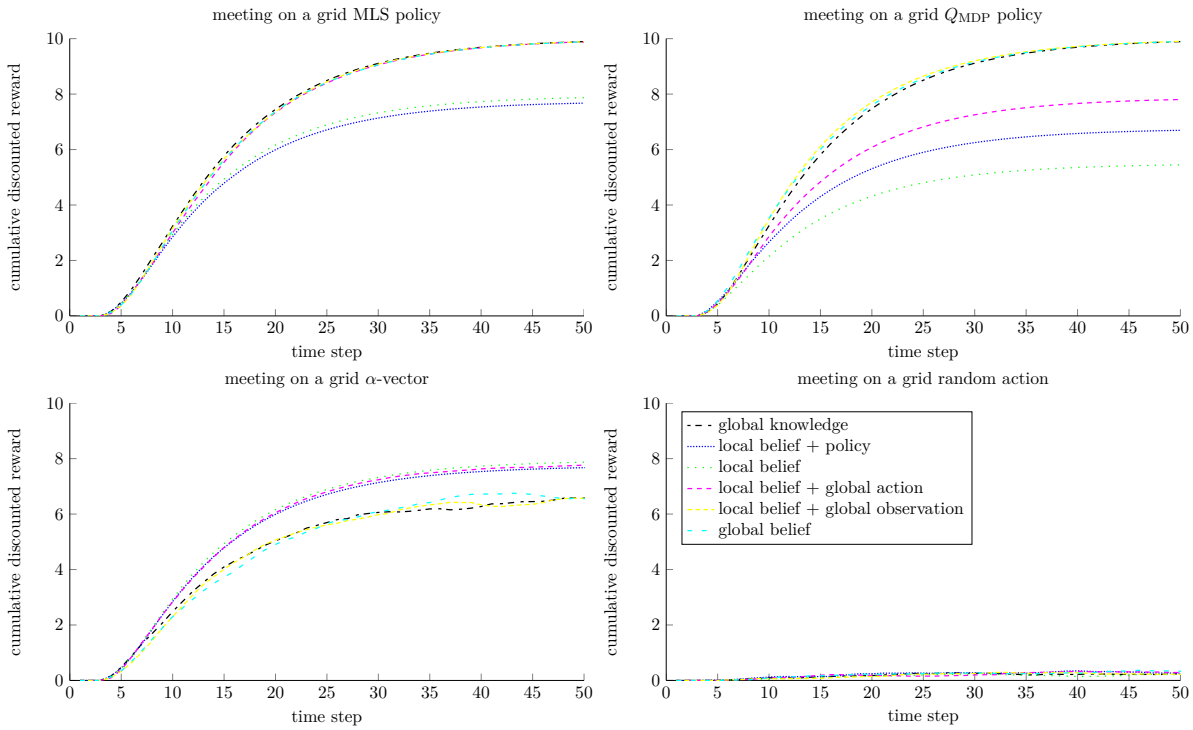


Figure 5-5: Discounted reward for the meeting on a grid problem over time. The expected discounted reward build-up over time shows an asymptotic approach to a limit.

5-3-2 Finite Horizon Utility

The complexity issues involved with infinite horizon policies are so severe that we cannot be sure that any infinite horizon policy is optimal. In other words, infinite horizon policies are undecidable [23]. To still be able to evaluate the performance in some way, we will compare the average expected reward of execution over a limited horizon. This is similarly done as the infinite horizon cases. We average the obtained rewards over 150 runs of simulation and deduce an expected reward to be obtained within certain time step limits. In order to compare accurately to existing optimal results, we forgo any discounting of the reward in the calculation of these values.

Comparing results of our approach to existing finite horizon methods does however require some things to take into account. The heuristic action decision method we use can be classified as a stationary policy. In other words, the action decisions do not depend on the time instant. For finite horizon decision problems however, the optimal policy is typically not a stationary one [16]. This can be more intuitively explained by realising that the agents can afford to take more risks near the final steps, realizing it will not reach certain negative states before reaching the final step.

To further elaborate, comparing finite horizon utility in this case is done merely to show performance statistics in smaller cases and view it in light of optimal decentralized finite horizon behavior. Because of this, we will only elaborate on some notable instances. The full set of finite horizon utility results is shown in Appendix C.

Recycling Robots Problem

In a similar effect to the infinite horizon utility in the previous section, the finite horizon utility for almost all belief update-action heuristic combinations lie very close to the finite horizon optimum [35]. Figure C-2 shows the results for the Q_{MDP} policy and its proximity with the finite horizon optimum.

The difference between the finite horizon optimum and our local belief methods is small, especially when used in combination with the Q_{MDP} policy heuristic as can be seen in Figure C-2. This distance seems to be a fixed length away form the optimal decentral case since it does not seem to increas over longer horizons, as can be seen in Figure 5-7.

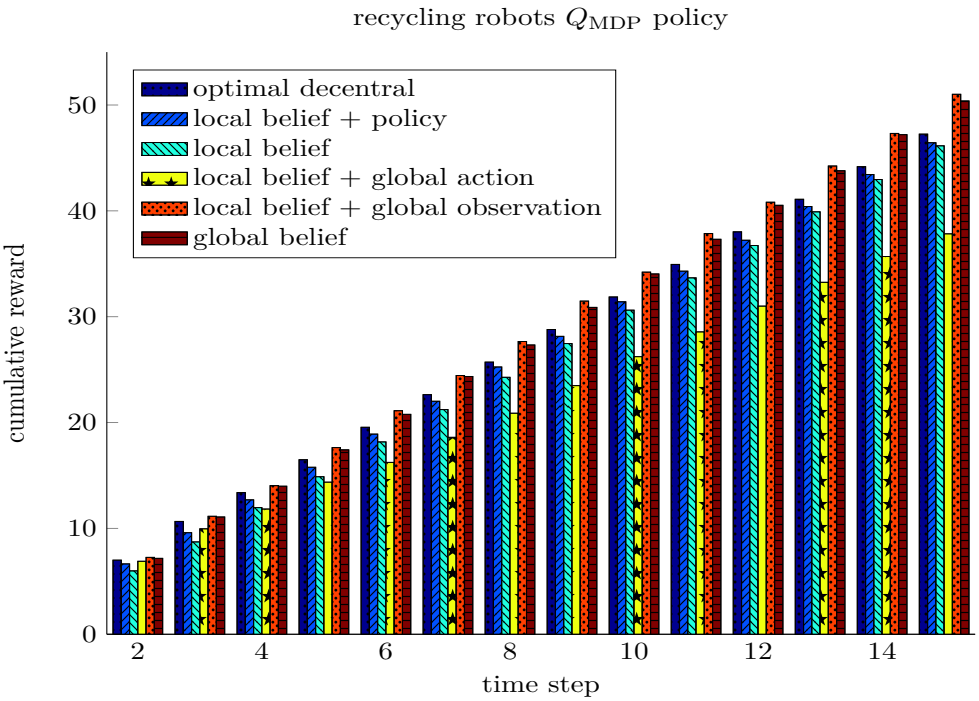


Figure 5-6: Average cumulative reward obtained at specific time steps for the recycling robot problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules. Local belief methods perform close to the optimal decentral values.

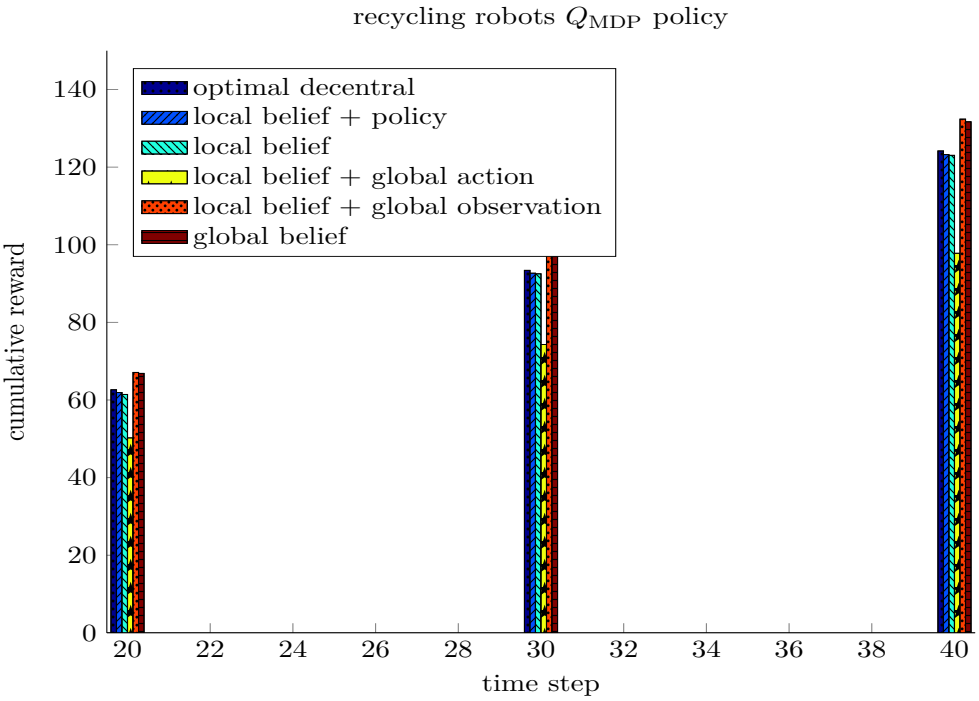


Figure 5-7: Average cumulative reward obtained at specific time steps for the recycling robot problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules. Local belief methods stay close to the optimal decentral values.

Broadcast Channel Problem

The results found for the broadcast channel problem resemble the results found for the recycling robots problem in that the values associated with local belief methods lie closely near the optimal decentral values, as can be seen in Figure C-6. This proximity remains if we look at the higher counts of time in Figure 5-9.

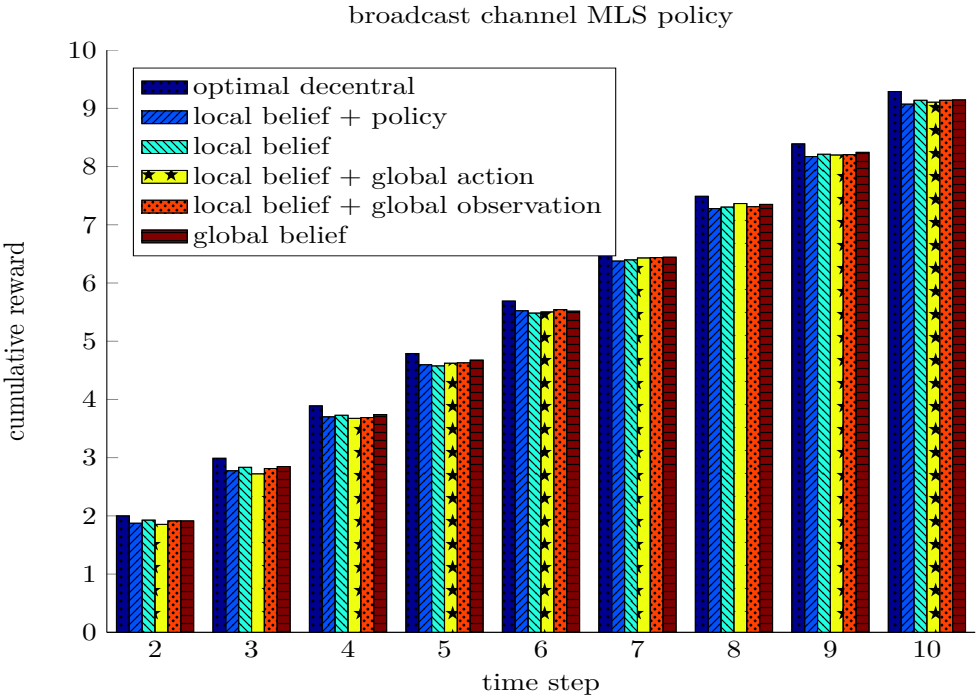


Figure 5-8: Average cumulative reward obtained at each time step for the broadcast channel problem using the MLS policy heuristic in combination with multiple belief update rules.

Meeting on a Grid

Since we did not have an infinite horizon utility for the meeting on a grid problem, we could only compare the behavior of the system to results we obtained ourselves. This is where the finite horizon utility comparison comes in handy, since it indicates the level of performance we can expect with respect to an ϵ optimal performance criterium.

What we can see from the comparison is that the local belief together with the MLS heuristic performs as well as the finite horizon optimum for lower horizons. This is shown in Figure C-11. In fact, up to a horizon of 6, it slightly outperforms the finite horizon optimum indicating that the provided ϵ optimum is at the lower end of its error bound. For higher horizons, the optimal finite horizon policy will again start to outperform the MLS policy with local beliefs.

With the Q_{MDP} policy we see about the same performance as with the MLS policy in that it lies close to the optimum for the small horizon cases, but loses that proximity for larger horizons. The difference in performance between a fully local belief and the altered belief is reduced though. This due to the fact that the QMDP provides a better tradeoff mechanism for dealing with state uncertainty.

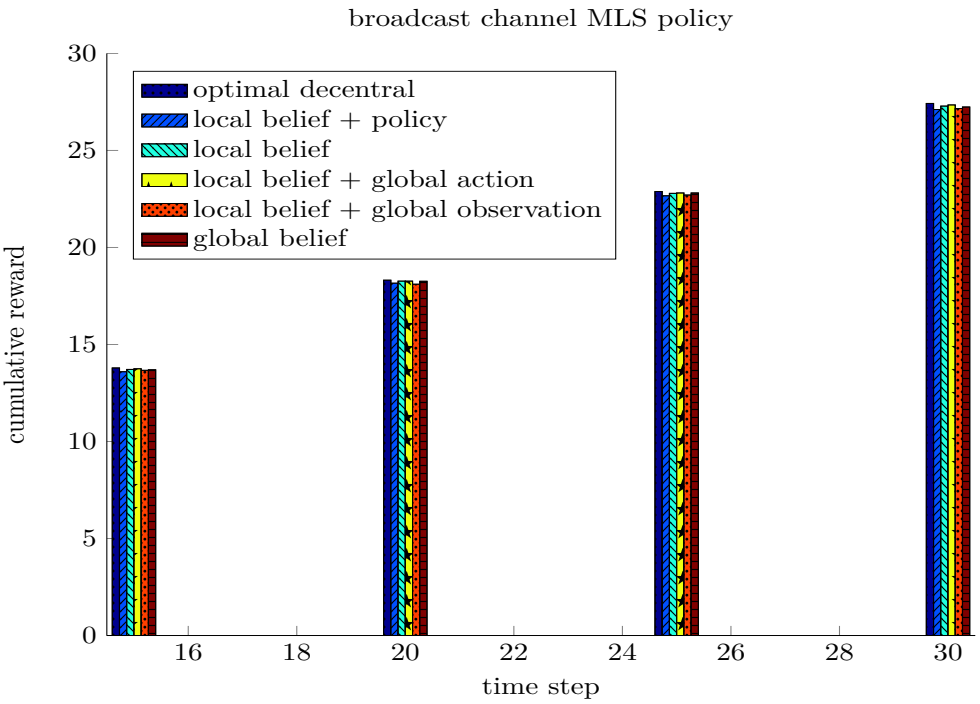


Figure 5-9: Average cumulative reward obtained at each time step for the broadcast channel problem using the MLS policy heuristic in combination with multiple belief update rules.

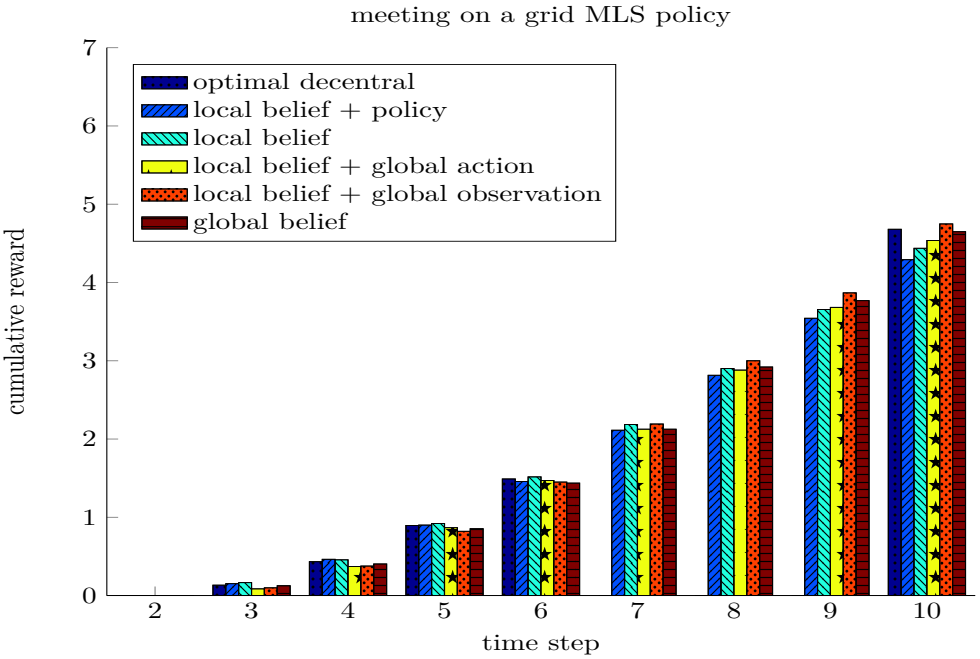


Figure 5-10: Average cumulative reward obtained at each time step for the meeting on a grid problem using the MLS policy heuristic in combination with multiple belief update rules. Local belief methods stay very close to the known optimal values.

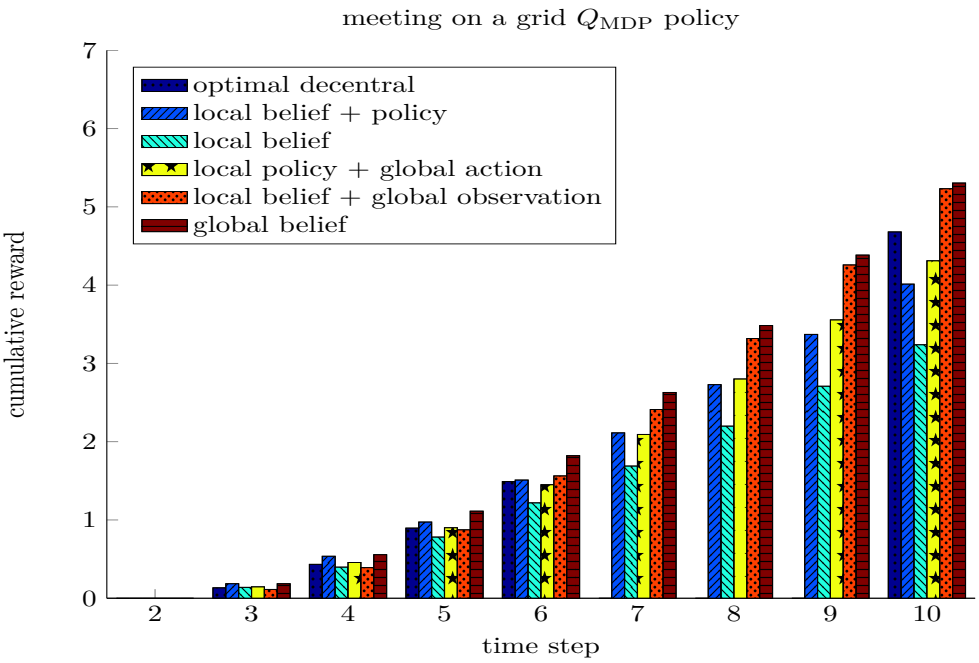


Figure 5-11: Average cumulative reward obtained at each time step for the meeting on a grid problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules. In lower time step count, the local belief methods slightly outperform the 'optimal' values, indicating that these are at the lower end of their ϵ optimal values.

5-4 Summary

In order to find the effect of an altered belief, we compared the behavior of three heuristic action selection methods in combination with six different forms of (partial) knowledge about the system. The aim was to find a positive effect of improved knowledge on the performance of the system. While it is certainly true that perfect knowledge of the state results in a better cooperative behavior among the agents, this is not necessarily true for other 'enhanced' types of belief where the agent would magically know the accurate global action or global observation. This fact is most notably seen in the broadcast channel problem, where anything but perfect knowledge of the current state results in about the same expected reward for all action selection methods.

While we cannot say with certainty that we can benefit from adding global information to a local agent belief update, we can say that it is possible. This leads us to say that we have found a way to improve cooperative behavior by altering the local belief update, we just can't guarantee improvement.

Chapter 6

Conclusions and Recommendations

In this thesis, we studied the broad field of Markov Decision Processes with its extensions into partial observability and in particular, multi-agent decentralized execution. It is this decentralized execution around which the focus of this thesis lies. We set out to find out whether or not we could improve the behavior of a multi-agent decentralised process by making use of an underlying centralized system. We solve the underlying systems problem to find appropriate actions to take in any given situation. The resulting policy is adapted for use by individual agents in a decentralized setting using heuristic functions. These heuristics are be based on the policy derived from underlying fully observable model.

The local agents know the global optimal policy and should therefore be able to perform optimally if they would perfectly know the global state. Since this is not the case, we aim to improve the local knowledge of the global state. This is done by altering the belief update function of the local agents to incorporate information on the global policy by way of the global value function. This allows local agents to reason over the likely actions performed by other agents.

The altered local belief update was tested in combination with multiple action decision heuristics on a set of three benchmark problems. These benchmark problems all present a stochastic environment in which the agents must fulfil a cooperative task without explicit communication or observability of the system state. The agents are assumed to know about the stochastic behavior of system transitions and observations. Results are obtained over multiple runs of each problem and averaged to form an expected value for each action heuristic in combination with a type of belief update.

While thoroughly grounded in theory, this approach could also be applied to a real world system where multiple agents must work together to accomplish a task. Due to the broadness with which the decentralized frameworks are defined, the amount of different systems that can be modelled are endless.
As previously stated, our original intention was to test these decentralized decision methods on a group of cleaning robots in order for them to tackle a larger, more complex environment than they could ever handle by themselves.

Conclusions

Results from simulations have shown that we can expect our approach to perform well in a decentralized setting. In fact, we have empirically shown that the use of local beliefs combined with heuristic functions can closely follow verified optimal results.

Furthermore, these tests show the possible advantage of altering local beliefs to improve cooperative behavior. This is most notable in cases where there is a distinct difference between the performance of a pure local belief and a belief where the global action is perfectly known. This leaves room to improve local reasoning about global actions and by extension, the cooperative performance of the agents in the system.

The added benefit of altering a local belief state is not guaranteed however. Whether or not it improves the cooperative behavior depends on the problem at hand and which type of heuristic is used. A most obvious case where this altered belief will not result in an improvement, is the one where even accurate global information, such as the global observation, does not cause an improvement. This can be seen as an upper bound on the value of the performance of the agents.

By using and specifically altering the local belief update in a decentralized problem, we can look in a new way at approaching decentralized problems and obtaining good results in the process.

We essentially replace a very heavy planning stage to a simplified global planning stage, and requiring a little more effort during execution by way of a heavier belief update.

Future Work

Future works can follow suit by more accurately defining and choosing the way in which global action information is fed into the belief update. In this thesis we used the Q-function to accomplish this since it defines a desirability of particular global actions in every state. This in turn lead to a distribution over possible global actions. This could be improved by building a new type of Q-function that holds the decentralized nature into account or weighs the current Q-function in such a way that it improves the cooperative behavior. Further study should show what kinds of methods are applicable and will result in improvement.

Other extensions of this work could focus on problems in which repeated information gathering is more important. If multiple agents should cooperatively build certainty about the state this can also be reflected in the belief update function. An individual agent can then rely more on the proper behavior of other agents and focus on acquiring their own information.

A major aspect of our research hinges on the assumption that we initially have access to the full system and its interactions. This might not always be the case however. In such a case we could possibly extend out method into online planning where the agents first learn concurrently (with full communication) but need to diminish this communication use over time. Eventually reaching a point where all communication is cut off and the agent must cooperate based on their local information only.

Appendix A

Derivation of belief update with knowledge of global action

As an intermediate measure, we derive a type of local belief update where an agent i is assumed to miraculously know the global action taken. This belief update will serve to help in understand the workings of the local beliefs of the agents.

$$b(s') = Pr\left\{s' | \mathbf{a}, o_i\right\} \tag{A-1}$$

Applying Bayes' rule, we can write this as:

$$b(s') = \frac{Pr\left\{s', \mathbf{a}, o_i\right\}}{Pr\left\{\mathbf{a}, o_i\right\}} \tag{A-2}$$

Again, we focus on the numerator term since the denominator can be written as a function of the numerator by:

$$Pr\left\{\mathbf{a}, o_i\right\} = \sum_{s' \in S} Pr\left\{s', \mathbf{a}, o_i\right\} \tag{A-3}$$

We can expand the numerator term to include the global observation $\mathbf{o}$ and the current state s as follows

$$Pr\left\{s', \mathbf{a}, o_i\right\} = \sum_{\mathbf{o} \in \mathbf{\Omega}} Pr\left\{s', \mathbf{a}, o_i, \mathbf{o}\right\} \tag{A-4}$$

$$= \sum_{\mathbf{o} \in \mathbf{\Omega}} \sum_{s \in S} Pr\left\{s', \mathbf{a}, o_i, \mathbf{o}, s\right\} \tag{A-5}$$

We use the chain rule to expand this expression

$$= \sum_{\mathbf{o}\in\mathbf{\Omega}} \sum_{s\in S} \cdot Pr\left\{o_i|\mathbf{o},\mathbf{a},s',s\right\} \tag{A-6}$$
$$\cdot Pr\left\{\mathbf{o}|\mathbf{a},s',s\right\} \tag{A-7}$$
$$\cdot Pr\left\{s'|\mathbf{a},s\right\} \tag{A-8}$$
$$\cdot Pr\left\{\mathbf{a},s\right\} \tag{A-9}$$

Using only sufficient statistics, this expression transforms to:

$$= \sum_{\mathbf{o}\in\mathbf{\Omega}} \sum_{s\in S} \cdot Pr\left\{o_i|\mathbf{o}\right\} \tag{A-10}$$
$$\cdot Pr\left\{\mathbf{o}|\mathbf{a},s'\right\} \tag{A-11}$$
$$\cdot Pr\left\{s'|\mathbf{a},s\right\} \tag{A-12}$$
$$\cdot b(s) \tag{A-13}$$

After shifting around the sum terms, the total belief update becomes:

$$b(s') = \frac{\sum_{\mathbf{o}\in\mathbf{\Omega}} \cdot Pr\left\{o_i|\mathbf{o}\right\} \cdot Pr\left\{\mathbf{o}|\mathbf{a},s'\right\} \sum_{s\in S} \cdot Pr\left\{s'|\mathbf{a},s\right\} \cdot b(s)}{\sum_{s'\in S} \sum_{\mathbf{o}\in\mathbf{\Omega}} \cdot Pr\left\{o_i|\mathbf{o}\right\} \cdot Pr\left\{\mathbf{o}|\mathbf{a},s'\right\} \sum_{s\in S} \cdot Pr\left\{s'|\mathbf{a},s\right\} \cdot b(s)} \tag{A-14}$$

Appendix B

Derivation of belief update with knowledge of global observation

As an intermediate measure, we derive a type of local belief update where an agent i is assumed to miraculously know the full global observation. This belief update will serve to help in part to understand the workings of the local beliefs of the agents.

$$b(s') = Pr\left\{s' | a_i, \mathbf{o}\right\} \tag{B-1}$$

Applying Bayes' rule, we can write this as:

$$b(s') = \frac{Pr\left\{s', a_i, \mathbf{o}\right\}}{Pr\left\{a_i, \mathbf{o}\right\}} \tag{B-2}$$

Again, we focus on the numerator term since the denominator can be written as a function of the numerator by:

$$Pr\left\{a_i, \mathbf{o}\right\} = \sum_{s' \in S} Pr\left\{s', a_i, \mathbf{o}\right\} \tag{B-3}$$

We can expand the numerator term to include a reasoning over the global action $\mathbf{a}$ and current state s as follows:

$$Pr\left\{s', a_i, \mathbf{o}\right\} = \sum_{\mathbf{a} \in \mathbf{A}} Pr\left\{s', a_i, \mathbf{o}, \mathbf{a}\right\} \tag{B-4}$$

$$= \sum_{\mathbf{a} \in \mathbf{A}} \sum_{s \in S} Pr\left\{s', a_i, \mathbf{o}, \mathbf{a}, s\right\} \tag{B-5}$$

We use the chain rule to expand this expression as follows:

$$= \sum_{\mathbf{a}\in\mathbf{A}} \sum_{s\in S} \cdot Pr\left\{\mathbf{o}|s', a_i, \mathbf{a}, s\right\} \tag{B-6}$$

$$\cdot Pr\left\{s'|a_i, \mathbf{a}, s\right\} \tag{B-7}$$

$$\cdot Pr\left\{\mathbf{a}|a_i, s\right\} \tag{B-8}$$

$$\cdot Pr\left\{a_i, s\right\} \tag{B-9}$$

Using only sufficient statistics, this expression transforms to:

$$= \sum_{\mathbf{a}\in\mathbf{A}} \sum_{s\in S} \cdot Pr\left\{\mathbf{o}|s', \mathbf{a}\right\} \tag{B-10}$$

$$\cdot Pr\left\{s'|\mathbf{a}, s\right\} \tag{B-11}$$

$$\cdot Pr\left\{\mathbf{a}|a_i, s\right\} \tag{B-12}$$

$$\cdot Pr\left\{a_i, s\right\} \tag{B-13}$$

After shifting around the sum terms, the total belief update becomes:

$$b(s') = \frac{\sum_{\mathbf{a}\in\mathbf{A}} Pr\left\{\mathbf{o}|s', \mathbf{a}\right\} Pr\left\{s'|\mathbf{a}, s\right\} \sum_{s\in S} Pr\left\{\mathbf{a}|a_i, s\right\} Pr\left\{a_i, s\right\}}{\sum_{s'\in S} \sum_{\mathbf{a}\in\mathbf{A}} Pr\left\{\mathbf{o}|s', \mathbf{a}\right\} Pr\left\{s'|\mathbf{a}, s\right\} \sum_{s\in S} Pr\left\{\mathbf{a}|a_i, s\right\} Pr\left\{a_i, s\right\}} \tag{B-14}$$

Appendix C

Finite Horizon Utility Graphs

The figures presented in this Appendix show the expected rewards we can obtain from multiple approaches, applied to the three benchmark problems used in this thesis. The figures are arranged per benchmark problem and per action decision type used on them.

Recycling Robots Problem

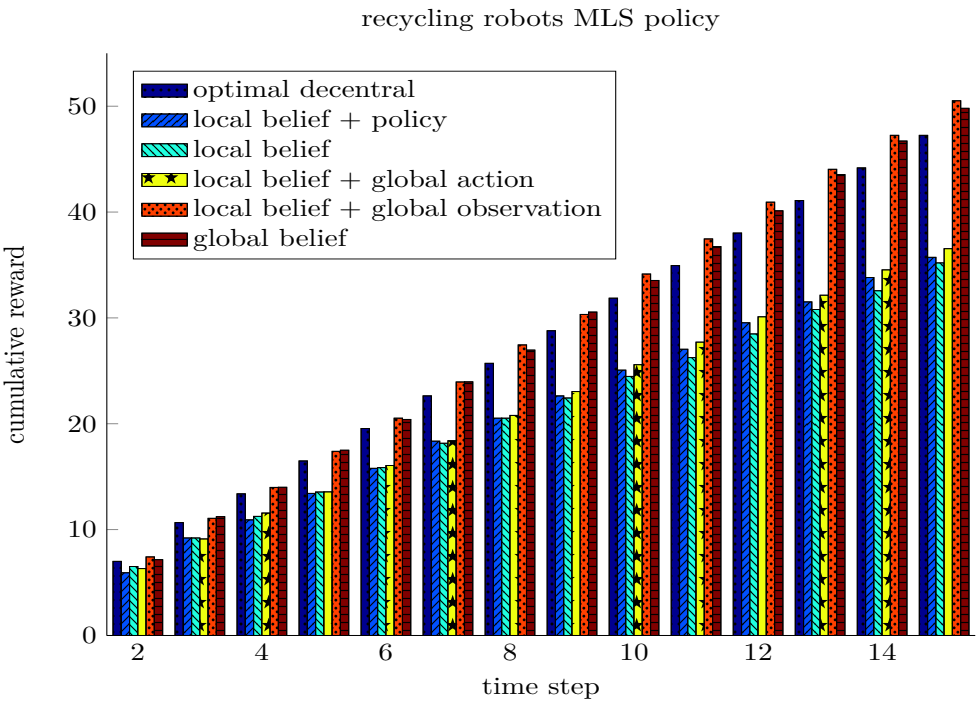


Figure C-1: Average cumulative reward obtained at each time step for the Recycling Robot problem using the MLS policy heuristic in combination with multiple belief update rules

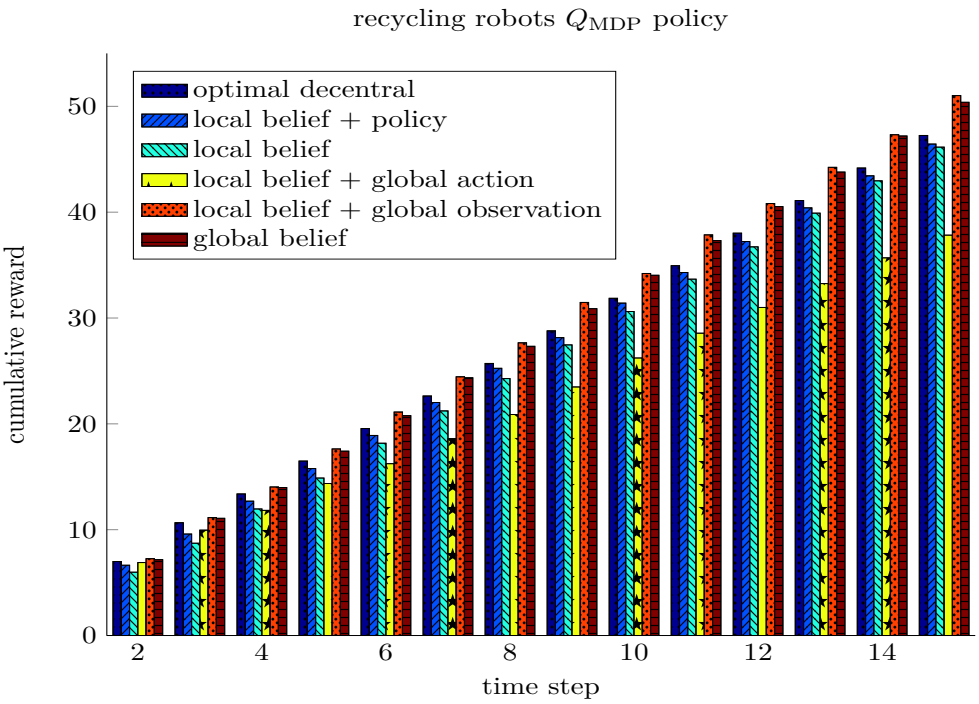


Figure C-2: Average cumulative reward obtained at each time step for the Recycling Robot problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules

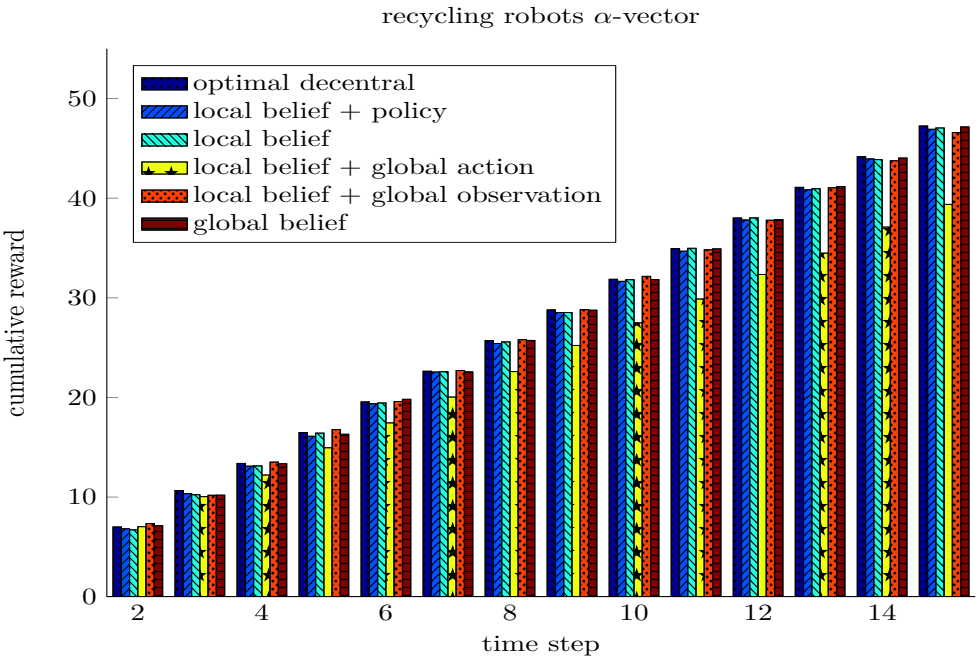


Figure C-3: Average cumulative reward obtained at each time step for the Recycling Robot problem using the α-vector heuristic in combination with multiple belief update rules

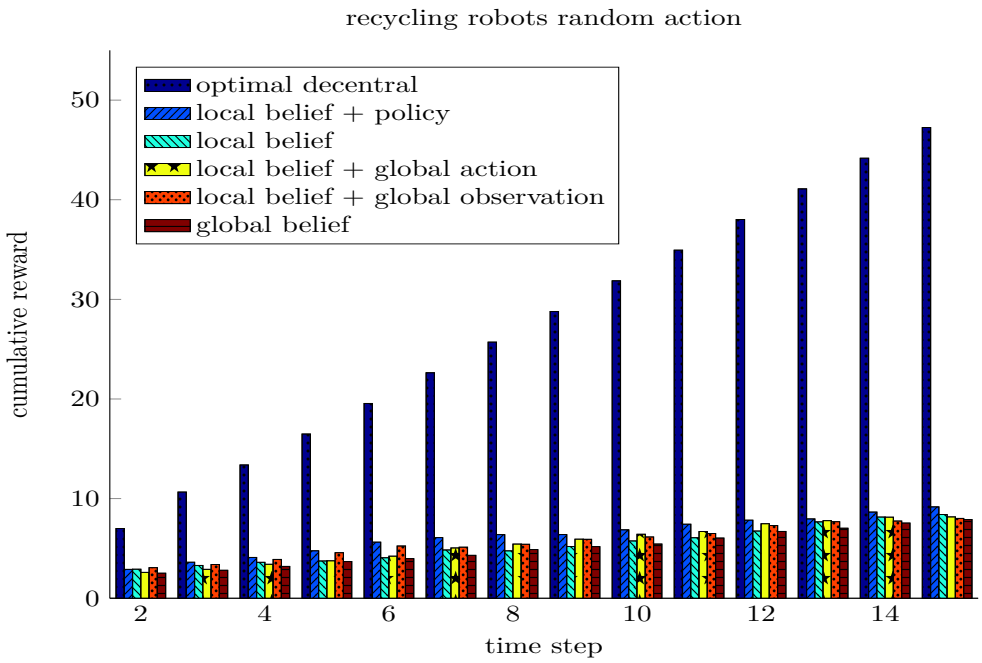


Figure C-4: Average cumulative reward obtained at each time step for the Recycling Robot problem using random uninformed actions

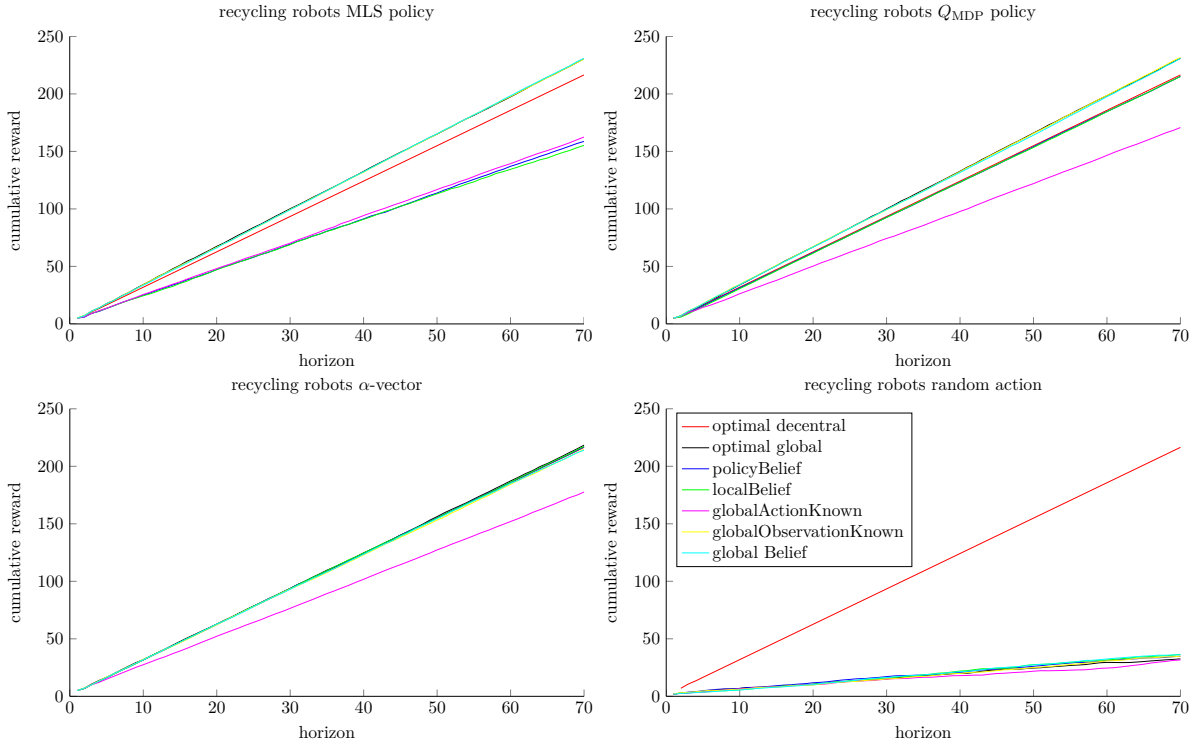


Figure C-5: Average cumulative reward obtained over time for the Recycling Robot problem

Broadcast Channel Problem

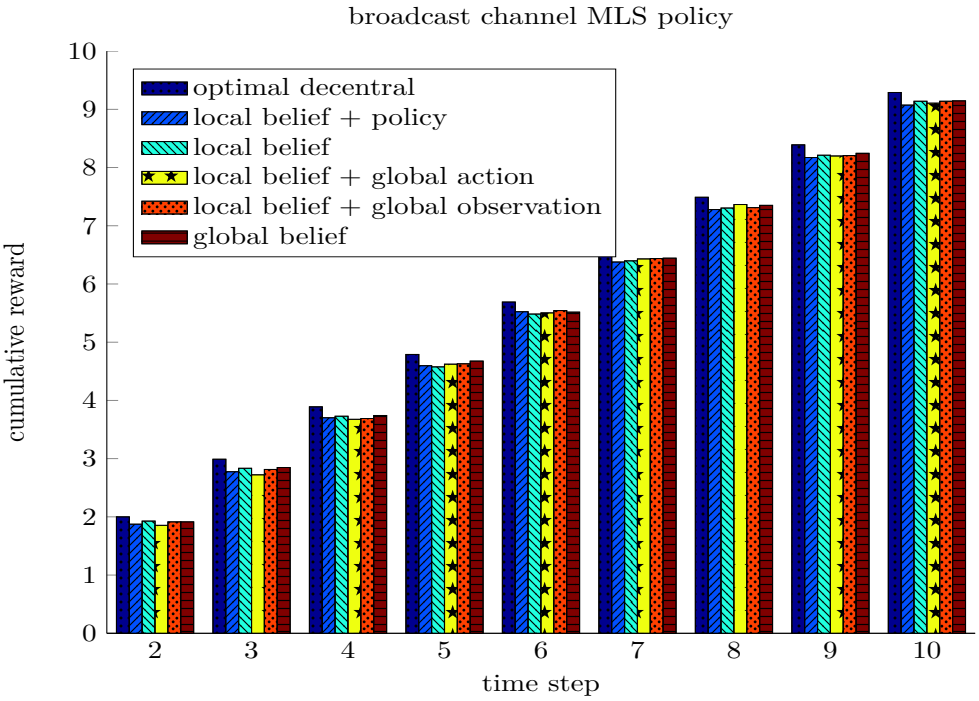


Figure C-6: Average cumulative reward obtained at each time step for the Broadcast Channel problem using the MLS policy heuristic in combination with multiple belief update rules

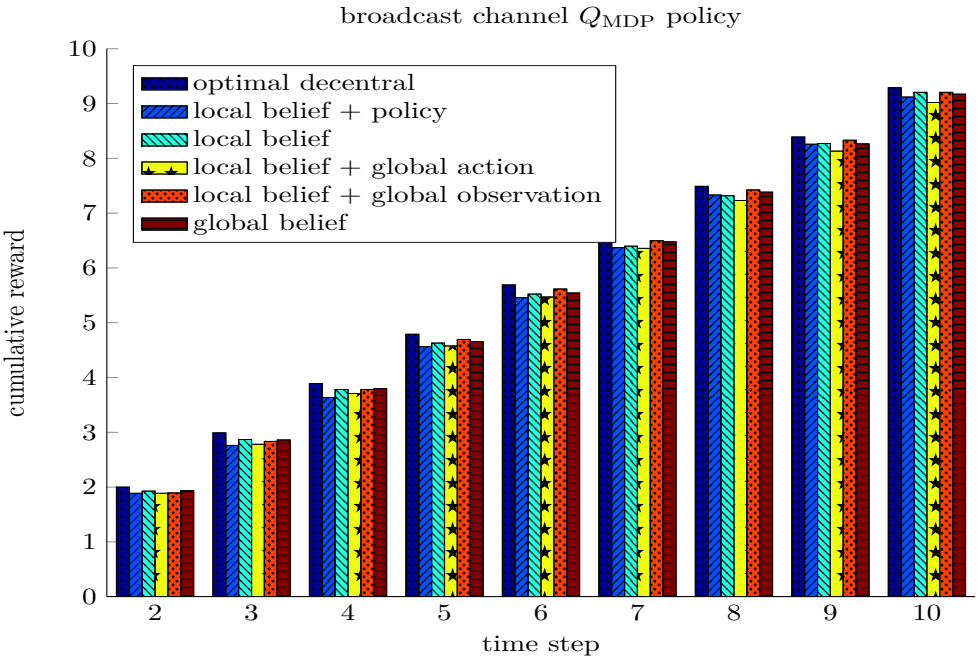


Figure C-7: Average cumulative reward obtained at each time step for the Broadcast Channel problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules

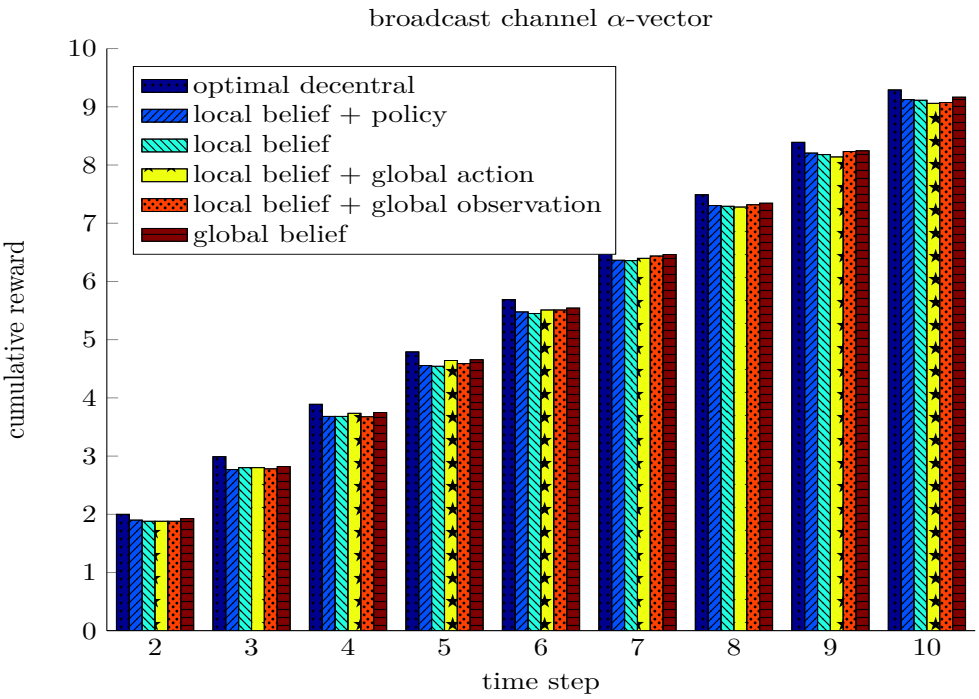


Figure C-8: Average cumulative reward obtained at each time step for the Broadcast Channel problem using the α-vector policy heuristic in combination with multiple belief update rules

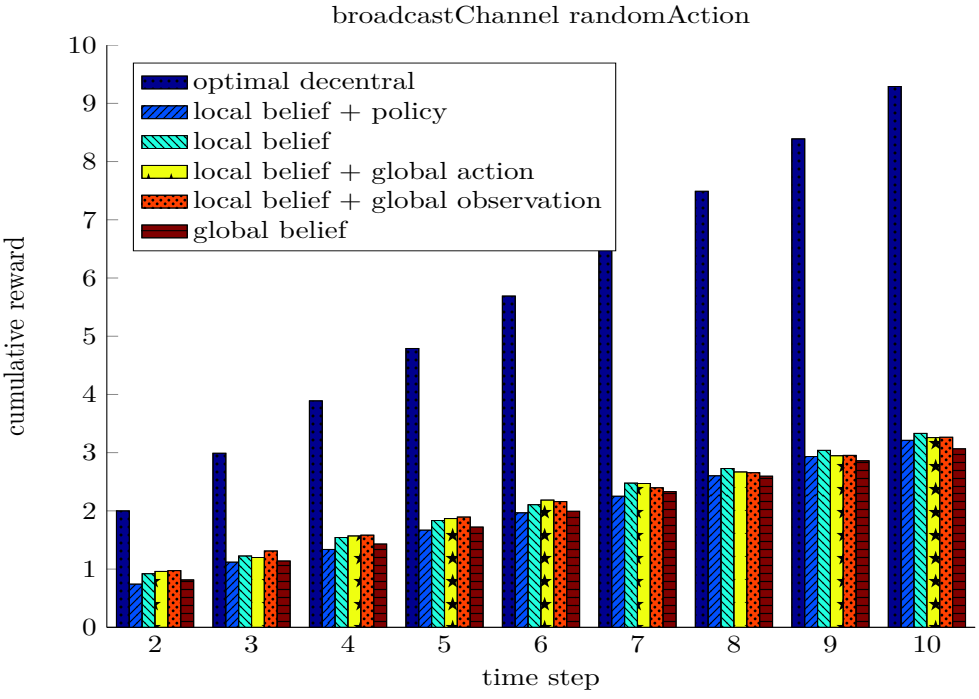


Figure C-9: Average cumulative reward obtained at each time step for the Broadcast Channel problem using random uninformed actions

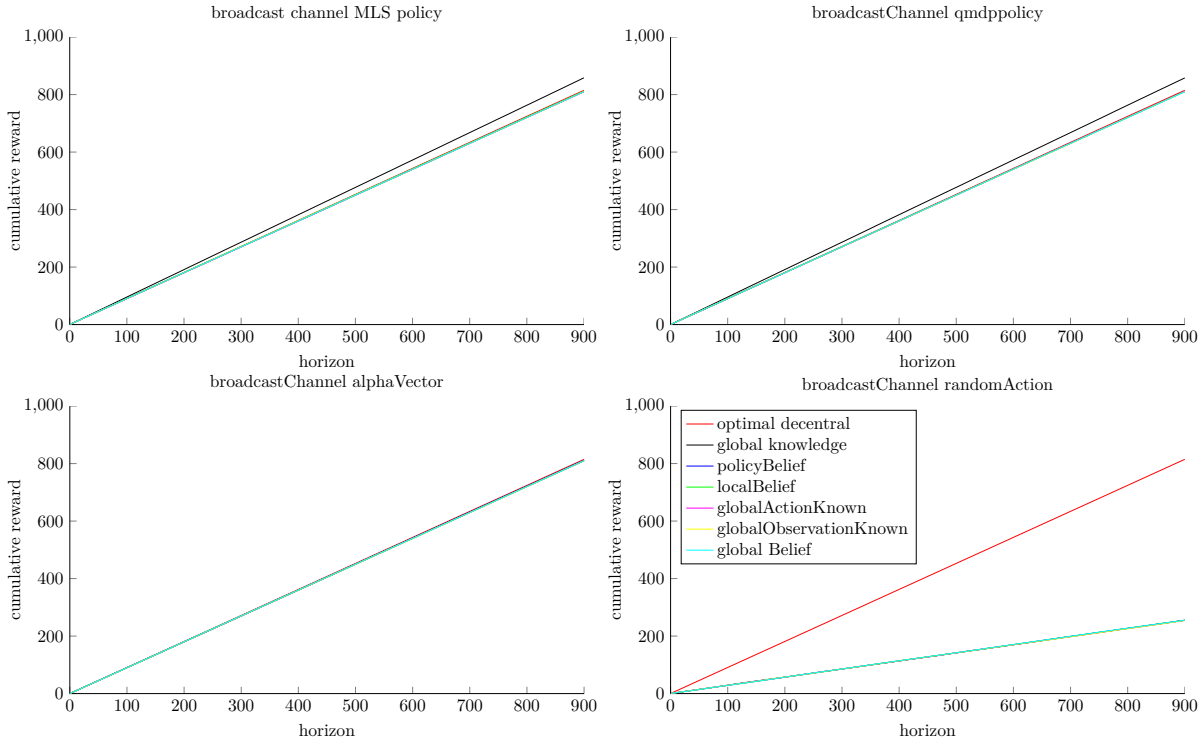


Figure C-10: Average cumulative reward obtained over time for the Broadcast Channel problem

Meeting on a Grid

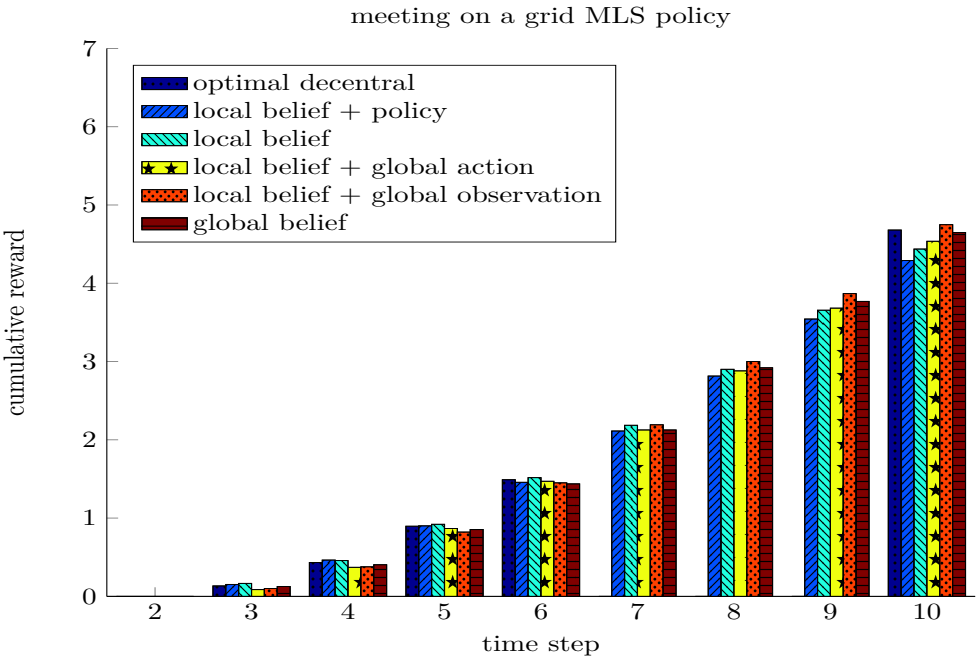


Figure C-11: Average cumulative reward obtained at each time step for the Meeting on a Grid problem using the MLS policy heuristic in combination with multiple belief update rules

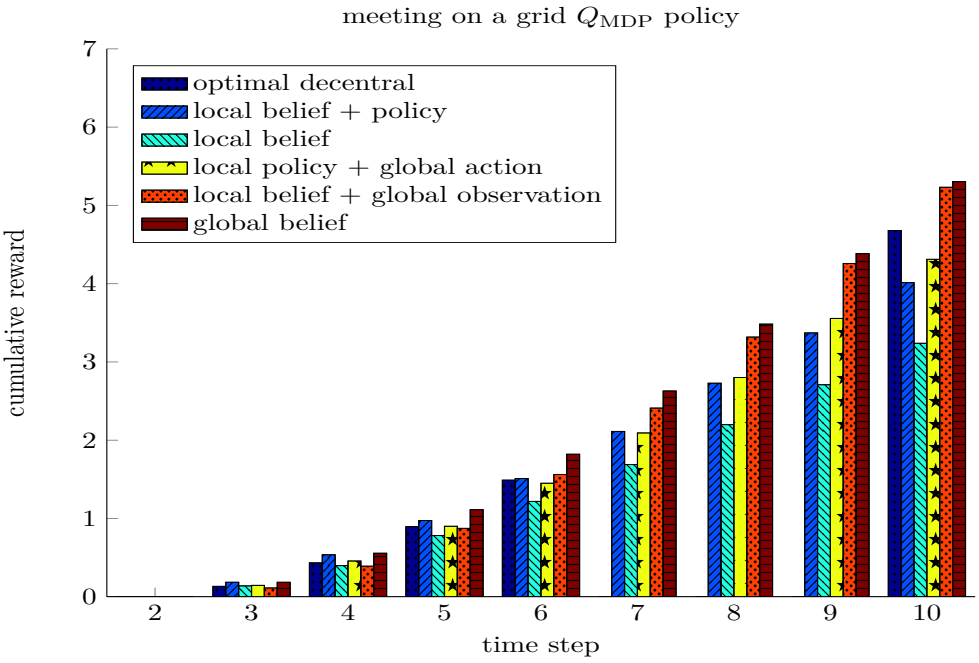


Figure C-12: Average cumulative reward obtained at each time step for the Meeting on a Grid problem using the Q_{MDP} policy heuristic in combination with multiple belief update rules

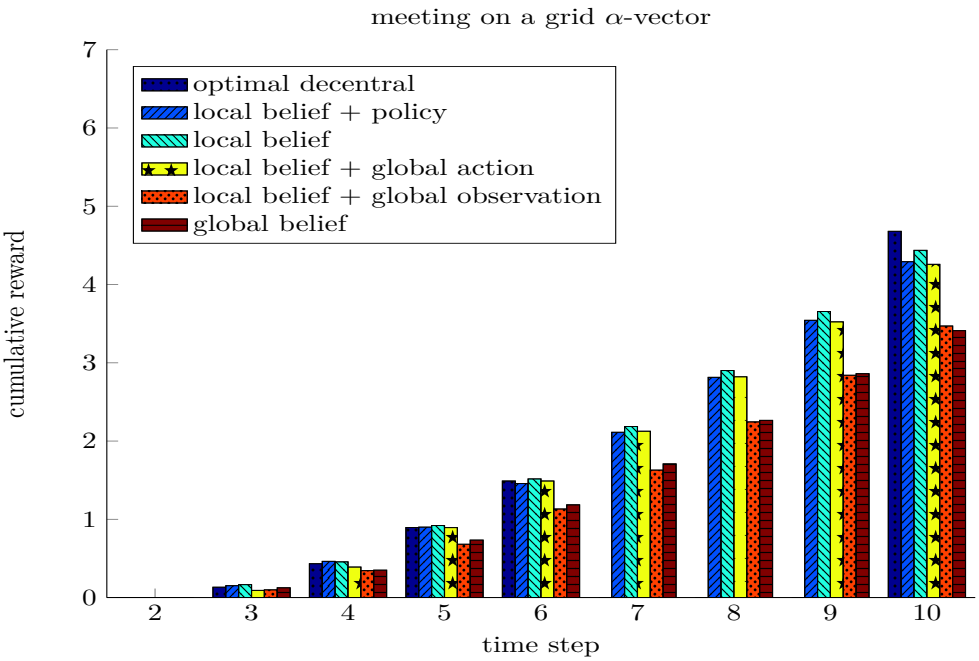


Figure C-13: Average cumulative reward obtained at each time step for the Meeting on a Grid problem using the α-vector policy heuristic in combination with multiple belief update rules

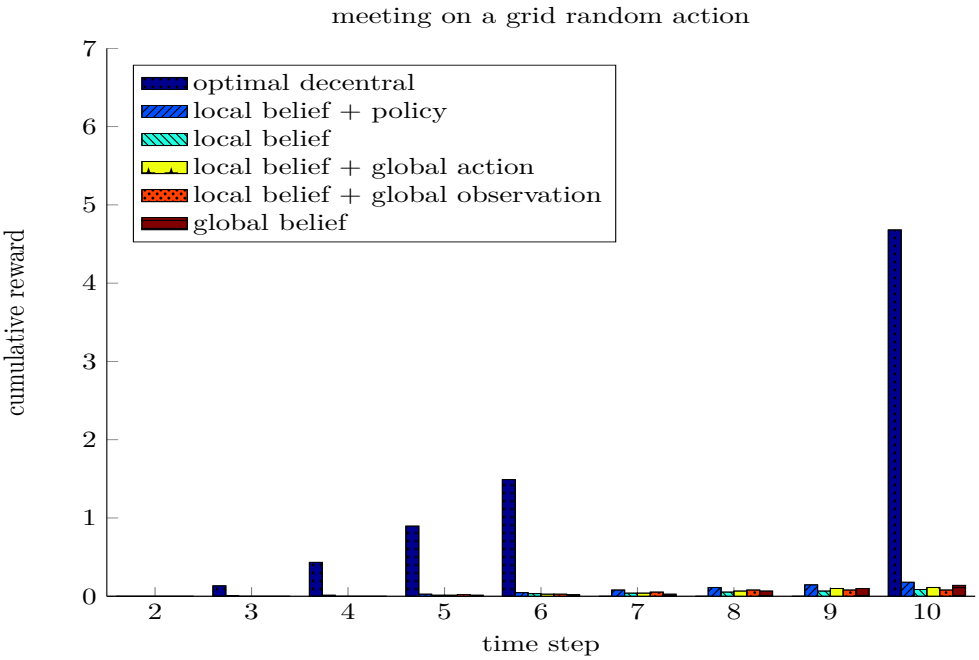


Figure C-14: Average cumulative reward obtained at each time step for the Meeting on a Grid problem using random uninformed actions

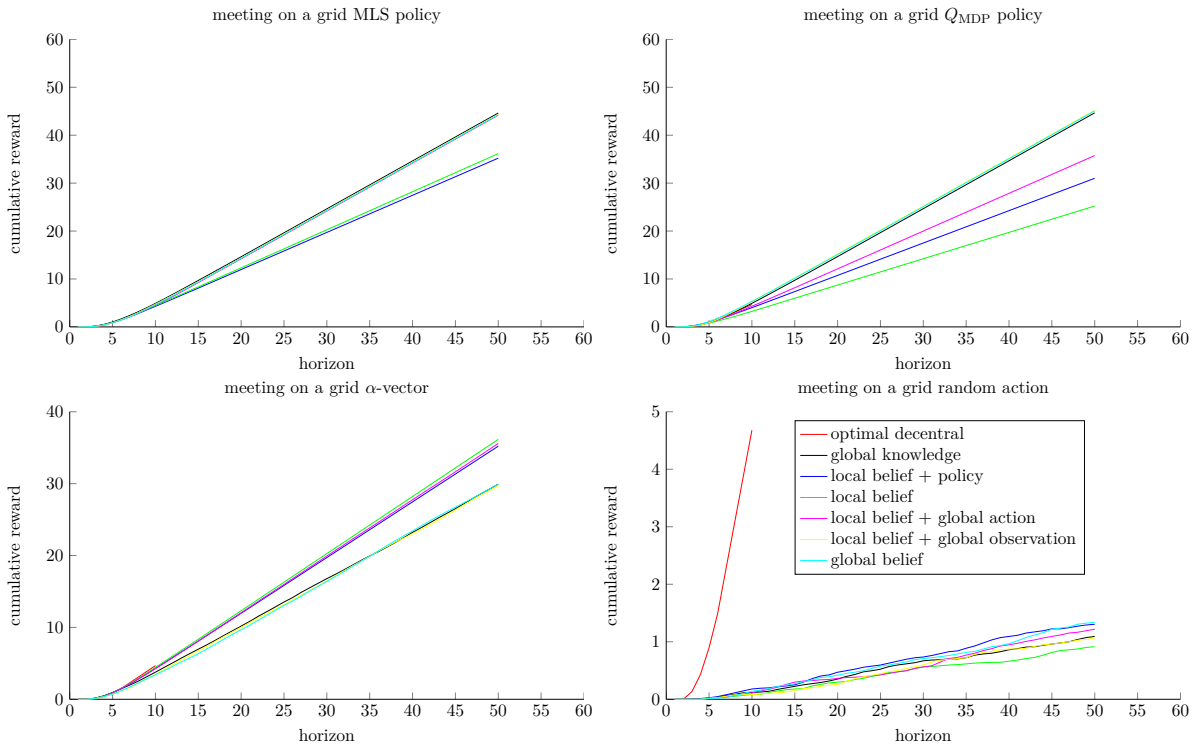


Figure C-15: Average cumulative reward obtained over time for the Meeting on a Grid problem

Bibliography

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[2] M. Wiering and M. Van Otterlo, *Reinforcement Learning: State-Of-The-Art*, vol. 12. Springer-Verlag New York Incorporated, 2012.

[3] R. Becker, "Solving transition independent decentralized markov decision processes," *Computer Science Department Faculty Publication Series*, p. 208, 2004.

[4] R. Becker, S. Zilberstein, and V. Lesser, "Decentralized markov decision processes with event-driven interactions," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 302–309, IEEE Computer Society, 2004.

[5] P. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multiagent settings," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 49–79, 2005.

[6] F. Wu and X. Chen, "Solving large-scale and sparse-reward Dec-POMDPs with correlation-mdps," *RoboCup 2007: Robot Soccer World Cup XI*, pp. 208–219, 2008.

[7] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "Robocup: A challenge problem for ai," *AI magazine*, vol. 18, no. 1, p. 73, 1997.

[8] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3323–3328, IEEE, 2012.

[9] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Distributed value functions for multi-robot exploration," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1544–1550, IEEE, 2012.

[10] J. Shen, V. Lesser, and N. Carver, "Minimizing communication cost in a distributed bayesian network using a decentralized MDP," in *Proceedings of the second international*

joint conference on Autonomous agents and multiagent systems, pp. 678–685, ACM, 2003.

[11] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, p. 133, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[12] A. Munir and A. Gordon-Ross, "An mdp-based dynamic optimization methodology for wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 616–625, 2012.

[13] S. Tadokoro, H. Kitano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjoh, T. Koto, I. Takeuchi, H. Takahashi, F. Matsuno, *et al.*, "The robocup-rescue project: A robotic approach to the disaster mitigation problem," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4, pp. 4089–4094, IEEE, 2000.

[14] P. Velagapudi, P. Varakantham, K. Sycara, and P. Scerri, "Distributed model shaping for scaling to decentralized pomdps with hundreds of agents," in *Proceedings of AAMAS*, vol. 11, 2011.

[15] R. Bellman, "Dynamic programming," *Princeton University Press*, 1957.

[16] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[17] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[18] E. Sondik, *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University, 1971.

[19] K. P. Murphy, "A survey of pomdp solution techniques," *environment*, vol. 2, p. X3, 2000.

[20] M. L. Littman, A. R. Cassandra, and L. Pack Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning-International Workshop*, pp. 362–370, Citeseer, 1995.

[21] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[22] D. Braziunas, "Pomdp solution methods," *University of Toronto*, 2003.

[23] D. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of markov decision processes," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 32–37, Morgan Kaufmann Publishers Inc., 2002.

[24] C. Boutilier, "Sequential optimality and coordination in multiagent systems," in *International Joint Conference on Artificial Intelligence*, vol. 16, pp. 478–485, 1999.

[25] E. Hansen, D. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 709–715, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[26] P. Xuan, V. Lesser, and S. Zilberstein, "Communication decisions in multi-agent cooperation: Model and experiments," in *Proceedings of the fifth international conference on Autonomous agents*, pp. 616–623, ACM, 2001.

[27] M. Roth, R. Simmons, and M. Veloso, "Decentralized communication strategies for coordinated multi-agent policies," *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pp. 93–105, 2005.

[28] M. Roth, R. Simmons, and M. Veloso, "What to communicate? execution-time decision in multi-agent POMDPs," in *Distributed Autonomous Robotic Systems 7*, pp. 177–186, Springer, 2006.

[29] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 289–353, 2008.

[30] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *International joint conference on artificial intelligence*, vol. 18, pp. 1025–1032, LAWRENCE ERLBAUM ASSOCIATES LTD, 2003.

[31] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing memory-bounded controllers for decentralized POMDPs," in *in Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI*, 2007.

[32] J. M. Ooi and G. W. Wornell, "Decentralized control of a multiple access broadcast channel: Performance bounds," in *Decision and Control, 1996., Proceedings of the 35th IEEE*, vol. 1, pp. 293–298, IEEE, 1996.

[33] D. Bernstein, E. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 52–57, 2005.

[34] C. Amato, J. S. Dibangoye, and S. Zilberstein, "Incremental policy generation for finite-horizon DEC-POMDPs," in *Proc. of the 19th IntŠl Conf. on Automated Planning and Scheduling*, pp. 2–9, 2009.

[35] M. T. Spaan, F. A. Oliehoek, and C. Amato, "Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pp. 2027–2032, AAAI Press, 2011.

Glossary

List of Acronyms

AH Action History

AOH Action Observation History

Dec-MDP Decentralised Markov Decision Process

Dec-POMDP Decentralized-Partially Observable Markov Decision Process

DP Dynamic Programming

MADP Multi-Agent Decision Process

MDP Markov Decision Process

MMDP Multi-Agent Markov Decision Process

NE Nash equilibrium

OH Observation History

PBVI Point Based Value Iteration

POMDP Partially Observable Markov Decision Process

POSG Partially Observable Stochastic Game

RL Reinforcement Learning

List of Symbols

a	action
A	action space
a_i	local action for agent i
A_i	set of local actions for agent i
$\mathbf{a}$	global action (or joint action)
$\mathbf{A}$	set of global (or joint) actions
$\vec{a}$	action history
b	belief-state
$\mathcal{D}$	set of agents
γ	discount factor
h	horizon
k	time step
o	observation
Ω	set of observations
o_i	local observation for agent i
Ω_i	set of local observations for agent i
$\mathbf{o}$	global (or joint) observation
$\mathbf{\Omega}$	set of global (or joint) observations
$\vec{o}$	observation history
O	observation function
π	policy
π_{MLS}	most likely state policy heuristic
π_{QMDP}	Q_{MDP} policy heuristic
$Pr\{x\}$	probability of x taking place
Q	state-action value function
r	reward
R	reward function
s	state
s'	state in the next time step k
S	set of states
T	state transition probability function
$\vec{\theta}$	action-observation history
V	(state) value function