Solving Multi-agent MDPs Optimally with Conditional Return Graphs

Joris Scharpff
Delft University of Technology
The Netherlands
j.c.d.scharpff@tudelft.nl

Diederik M. Roijers
University of Amsterdam
The Netherlands
d.m.roijers@uva.nl

Frans A. Oliehoek
University of Amsterdam
The Netherlands
f.a.oliehoek@uva.nl

Matthijs T. J. Spaan
Delft University of Technology
The Netherlands
m.t.j.spaan@tudelft.nl

Mathijs M. de Weerdt
Delft University of Technology
The Netherlands
m.m.deweerdt@tudelft.nl

ABSTRACT

In cooperative multi-agent sequential decision making under uncertainty, agents must coordinate in order find an optimal joint policy that maximises joint value. Typical solution algorithms exploit additive structure in the value function, but in the fully-observable multi-agent MDP setting (MMDP) such structure is not present. We propose a new optimal solver for so-called TI-MMDPs, where agents can only affect their local state, while their value may depend on the state of others. We decompose the *returns* into local returns per agent that we represent compactly in a *conditional return graph (CRG)*. Using CRGs the value of a joint policy as well as bounds on the value of partially specified joint policies can be efficiently computed. We propose CoRe, a novel branch-and-bound policy search algorithm building on CRGs. CoRe typically requires less runtime than the available alternatives and is able to find solutions to problems previously considered unsolvable.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms

Keywords

Multi-agent planning, fully observable, MMDP, transition independence

1. INTRODUCTION

When planning in uncertain domains with sequential planning decisions, cooperative teams of agents must coordinate to maximise their (joint) team value. For each possible state of the environment, the agents must select a joint action that leads to the highest expected sum of rewards [3]. In the context of *full-observability*, where each agent can observe the entire state space at every planning decision, the availability of complete and perfect information can be exploited to develop coordination policies with higher values. On the other hand, as such policies are conditioned on the global state – which is typically exponentially-sized in the number of agents – the benefit of higher values is often associated with a substantial increase in the computational burden to find such policies. Especially, when *optimal* plans are required, e.g., when dealing with strategic behavior as part of an allocation mechanism that optimises social welfare [4, 16], approaches are needed that combat this increase in complexity as a result of full observability.

In response to this complexity, two important lines of work developed. One line has proposed to find approximate solutions by imposing and exploiting an additive structure in the value function [6]. This approach has been applied in a range of stochastic planning settings, fully and partially observable alike [5, 10, 8, 14]. The drawback of such methods, however, is that typically no error guarantees can be given.

Another line of work has not sacrificed optimality, but instead focused on specific properties of sub-classes of problems that can be exploited [2, 1, 11, 19]. In particular, a number of methods that exploit the same type of additive structure in the value function have been shown to be exact, simply because the value functions of the sub-class of problems they address are guaranteed to have such shape [12, 13, 18]. However, all these approaches are for decentralised models in which the actions are conditioned only on *local* observations. Consequentially, optimal policies for decentralised models typically result in lower value than could have been achieved by optimal policies for their fully-observable counterparts (shown in our experiments).

Unfortunately, fully-observable problems do not belong to the class of problems that admit a value function that is exactly factored into additive components. The intuition is that, by observing the full state, it will be possible for an agent to better predict the actions of other agents than when only observing a local state. This in turn means that each agent's action should be conditioned on the full state, and that the value function therefore also depends on the full state, ruling out the possibility of a factored value function in general.

Nevertheless, we contribute to the second line of work by exploiting other properties in the fully observable context. In particular, we exploit the reward structure of multi-agent Markov decision processes (MMDPs) [3] in the context of

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

transition independent agents [2] (i.e., agents can influence only their own state) with additively factored rewards (i.e., rewards depend on small groups of agents [12]) operating in a fully-observable environment (i.e., there is no observation independence). We illustrate the importance of the TI-MMDP model with a real-world, numerical maintenance planning problem [16].

We propose to exploit the reward structure of such TI-MMDPs by decomposing the *returns* of all possible execution sequences—i.e., all possible sequences of states and actions that can occur from the initial time step to the planning horizon—into returns depending only locally on relevant states and actions of small subsets of agents.

That is, we suggest an approach based on three key observations: 1) Contrary to the optimal value function, returns *can* be decomposed without loss of optimality, as they depend only on local information about the execution sequence, while the optimal value depends on agents conditioning their local policies on the entire joint state. 2) These factored returns can be used in effective data structures, *conditional return graphs* (CRGs) to compute upper and lower bounds on the optimal (non-factored) *value function.* 3) We can detect the presence of *conditional reward independence*, i.e., the absence of further reward interactions, to decouple agents during policy search.

We propose *conditional return policy search* (CoRe), a branch-and-bound policy search algorithm for fully observable TI-MMDPs that uses CRGs, and show that it is an effective method when reward interactions between agents are sparse. We evaluate CoRe on instances of the aforementioned maintenance planning problem with uncertain outcomes and a very large state space. First we establish that decentralised policies are inadequate in achieving optimal coordination. Next, we demonstrate that CoRe evaluates only a fraction of the policy search space and, as a consequence, is able to find policies for previously unsolvable instances while commonly requiring less runtime than its alternatives.

2. MODEL

We consider a (fully-observable) *transition-independent, multi-agent Markov decision process (TI-MMDP)*. We assume a finite-horizon of length h, and do not discount the rewards over time. A TI-MMDP is a tuple $\langle N, S, A, T, \mathcal{R}\rangle$:

- $N = \{1, ..., n\}$ is a set of n enumerated agents,
- $S = S^1 \times ... \times S^n$ is the factorised state space, which is the Cartesian product of n local factorised states S^i per agent (composed of state features $f \in F$, i.e., $s^i = \{f^i_x, f^i_y, \ldots\}$),
- $A = A^1 \times ... \times A^n$ is the joint action space, which is the Cartesian product of the n local action spaces A^i,
- $T(s, \vec{a}, \hat{s}) = \prod_{i\in N} T^i(s^i, a^i, \hat{s}^i)$ is the transition function, which is the product of the local transition functions due to transition independence, and
- $\mathcal{R}$ is the set of reward functions that we assume w.l.o.g. is structured as $\{R^1, ..., R^n\} \cup \mathcal{R}^e$. Here each R^i is the *local reward* function for agent i and $\mathcal{R}^e$ is a set of m *interaction rewards* such that every $R^e \in \mathcal{R}^e$ defines the reward over a set of agents $e \subseteq N$ with $|e| > 1$. The total team reward per time step, given a joint state s, joint action $\vec{a}$ and new joint state $\hat{s}$, is the sum of the local and interaction rewards:

  $$R(s, \vec{a}, \hat{s}) = \sum_{e=1}^{|\mathcal{R}|} R^e(s^e_t, \vec{a}^e_t, s^e_{t+1})$$
  $$= \sum_{i\in N} R^i(s^i, a^i, \hat{s}^i) + \sum_{e=1}^{m} R^e(\{s^j\}_{j\in e}, \{\vec{a}^j\}_{j\in e}, \{\hat{s}^j\}_{j\in e})$$

  where we abuse notation such that e denotes both the reward function index and the agent scope $e \subseteq N$ of that reward function.

Two agents, i and j are said to be *dependent* when there exists a reward function that has both agents in scope, e.g., a two-agent reward function $R^{i,j}(s^i, s^j, a^i, a^j, \hat{s}^i, \hat{s}^j)$. We focus on problems with sparse *interaction rewards*, i.e., those reward functions R^e that depend on more than one agent only have non-zero rewards for a small subset of the local joint actions (e.g., $A^i \times A^j$) in their scope. We call local joint actions with a non-zero interaction reward *dependent actions.* Note that this focus is not restrictive in any way—all local joint actions can be dependent—but just indicates a level of sparsity.

The goal of the agents in a TI-MMDP is to find the optimal joint policy π^* that maximises the expected sum of rewards obtained from following the policy, expressed recursively using the Bellman equation:

$$V^*(s_t) = E[\sum_{x=t}^{h} \sum_{e=1}^{|\mathcal{R}|} R^e(s^e_x, \vec{a}^e_x, s^e_{x+1})|s_t, \pi^*] = \qquad (1)$$
$$\max_{\vec{a}_t} \sum_{s_{t+1}\in S} T(s_t, \vec{a}_t, s_{t+1})\Big(\sum_{e=1}^{|\mathcal{R}|} R^e(s^e_t, \vec{a}^e_t, s^e_{t+1}) + V^*(s_{t+1})\Big)$$

At the last timestep ($t = h$) there are no future rewards, and $V^*(s_h) = 0$ for every $s_h \in S$. Although this value can be computed through a series of maximisations over the planning period, it cannot be written as a sum of independent local value functions $V^*(s) \neq \sum_{e=1}^{|\mathcal{R}|} V^{e,*}(s^e)$ without losing optimality [9].

Instead, we factor the *returns* of *execution sequences*, i.e., the sum of rewards obtained from following action sequences, which is optimality preserving. We denote an execution sequence up until time t as $\theta_t = [s_0, \vec{a}_0, ..., s_{t-1}, \vec{a}_{t-1}, s_t]$ and its return until time step t given θ_t is the sum of its rewards $\sum_{x=0}^{t-1} R(s_{\theta,x}, \vec{a}_{\theta,x}, s_{\theta,x+1})$, where $s_{\theta,x}$, $\vec{a}_{\theta,x}$ and $s_{\theta,x+1}$ respectively denote the state and joint action at time x, and the resulting state at time $x + 1$ in this sequence. A seemingly trivial, but important observation is that the return of an execution sequence can be written as the sum of local functions:

$$Z(\theta_t) = \sum_{e=1}^{|\mathcal{R}|} \sum_{x=0}^{t-1} R^e(s^e_{\theta,x}, \vec{a}^e_{\theta,x}, s^e_{\theta,x+1}), \qquad (2)$$

where $s^e_{\theta,x}$, $\vec{a}^e_{\theta,x}$ and $s^e_{\theta,x+1}$ denote the sets of local states and actions from θ_t that are relevant for R^e. Contrary to the optimal value function, (2) is an additive function of the reward components and, as such, can be computed locally.

The return of (2) does not directly give us the optimal value, i.e., the expected reward achieved from following the optimal policy. To compute the expected value of a policy, we sum the expected returns of all possible execution paths reachable under policy π, denoted by $\theta_h|\pi$, such that:

$$\sum_{\theta_h|\pi} Pr(\theta_h)Z(\theta_h) = \sum_{\theta_h|\pi} \prod_{t=0}^{h-1} T(s_{\theta,t}, \pi(s_{\theta,t}), s_{\theta,t+1})Z(\theta_h), \tag{3}$$

Now, (3) is nicely structured such that it expresses the value in terms of additively factored terms (the $Z(\theta_h)$). However, comparing (1) and (3), we see that the price for this is that we no longer are expressing the *optimal* value function, but instead the value for some joint policy. In fact (3) corresponds to an equation for *policy evaluation*. This means that it is not a obvious basis for dynamic programming, but it is suited for use in policy search methods. Even though policy search methods have their own problems in scaling to large problems, we show in this paper that the structure of (3) can be leveraged.

In particular, because the return of an execution sequence can be decomposed into additive components (Eq. 2), we decouple and store these returns locally for each agent in a structure we call the *conditional return graph* (CRG). This CRG can subsequently be used in policy search to efficiently compute the expected value of (3) when, during evaluation, the transition probability $Pr(\theta)$ of an execution sequence θ becomes known. Moreover, the returns stored in the CRG can be used to bound the expected value of sequences, allowing branch-and-bound pruning. Finally, using a CRG it is possible to dynamically detect conditional reward independence – reward independencies as a result from previous planning choices – between sets of agents and decouple the remaining planning problems without losing optimality.

3. CONDITIONAL RETURN GRAPHS

Following (3), the sum of returns can be partitioned into additive components $\mathcal{R}_i$ such that for each agent $i \in N$ its *local* reward is given by $\mathcal{R}_i = R^i \cup \mathcal{R}^e_i$, where R_i is the individual reward function of that agent and $\mathcal{R}^e_i \subseteq \mathcal{R}^e$ is an (assigned) sub-set of interaction reward functions in which the agent is involved. Then, a *conditional return graph* (CRG) for agent i is a directed acyclic graph (DAG) that represents all possible histories of *local states* s^i (indicated by the circles in Fig. 1a). The goal is to represent all possible local rewards $\mathcal{R}_i$ that can be received for a given individual state transition $(s^i, a^i, \hat{s}^i)$, which we denote with τ^i. These different rewards are encoded using differently labeled edges $(s^i, \hat{s}^i)$ between the individual states. However, clearly the reward received can depend on states and actions of other agents that participate in $\mathcal{R}_i$. As such, each pair of individual states $(s^i, \hat{s}^i)$ can be connected by many edges, each one corresponding to a different local transition $\tau^e = (s^e, a^e, \hat{s}^e)$ (used as label) and thus potentially different local reward $\mathcal{R}_i(s^e, \vec{a}^e, \hat{s}^e)$.

While this does capture all the rewards possible, with the DAG of Fig. 1a as a result, we would like to better represent the structure present in the local rewards: we want to group together all local transitions $\tau^e = (s^e, \vec{a}^e, \hat{s}^e)$ that lead to the same local reward. To this end, we introduce a slightly more complex graph (shown in Fig. 1b) which can essentially be interpreted as an abstraction of local transitions $\tau^e = (s^e, \vec{a}^e, \hat{s}^e)$ that lead to the same rewards. This is exactly the CRG:

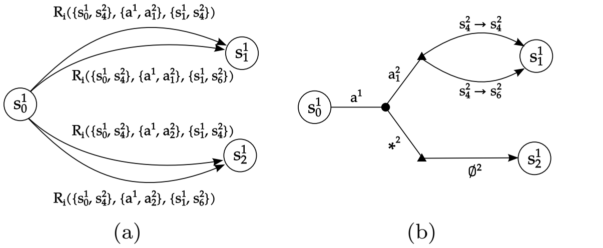

$R_i(\{s^1_0, s^2_4\}, \{a^1, a^2_1\}, \{s^1_1, s^2_4\})$
$R_i(\{s^1_0, s^2_4\}, \{a^1, a^2_1\}, \{s^1_1, s^2_6\})$
$R_i(\{s^1_0, s^2_4\}, \{a^1, a^2_2\}, \{s^1_1, s^2_4\})$
$R_i(\{s^1_0, s^2_4\}, \{a^1, a^2_2\}, \{s^1_1, s^2_6\})$
s^1_0
s^1_1
s^1_2
(a)
s^1_0
a^1
a^2_1
$*^2$
$s^2_4 \rightarrow s^2_4$
$s^2_4 \rightarrow s^2_6$
$\emptyset^2$
s^1_1
s^1_2
(b)

Figure 1: Example of a transition graph for one agent of a two-agent problem where (a) shows the complete state/transition graph and (b) the equivalent but more compact CRG.

Definition 1 (Conditional Return Graph ϕ). *Given a valid partitioning $\bigcup_{i \in N} \mathcal{R}^e_i$ of the interaction rewards $\mathcal{R}^e$ such that for every agent $i \in N$: $\mathcal{R}_i = R^i \cup \mathcal{R}^e_i$, the* Conditional Return Graph (CRG) ϕ_i *is a* directed acyclic graph *that for every state transition $\tau^e = (s^e, \vec{a}^e, \hat{s}^e)$ that may result in a unique reward $\mathcal{R}_i(\tau^e)$ contains a path consisting of:*

- ***local state nodes** $s^i \in s^e$ and $\hat{s}^i \in \hat{s}^e$,*
- ***after-state nodes** for every action a^i of agent i in joint action $\vec{a}^e$ connected by **action arcs** such that every after-state captures an extension of the execution sequence by joint action $\vec{a}$ through individual action arcs for every $a^i \in \vec{a}^e$. Action arcs are labelled either by a single action a^i or a 'wildcard' $*^i$ to denote all of the non-dependent actions of agent i for which no branch exists,*
- *a **fully-specified after-state node** that is connected to the local state s^i through all after-states and action arcs for the joint action $\vec{a}^e$. From the fully-specified after-state an **influence arc** to the new local state node $\hat{s}^i$ is added for the (external) state transition from $s^{e\setminus i}$ to $\hat{s}^{e\setminus i}$ and this arc is labelled by reward $\mathcal{R}_i(s^e, \vec{a}^e, \hat{s}^e)$. Influence arcs can also be labelled by a 'wildcard' $\varnothing$.*

In Fig. 1b, the local state nodes are displayed as circles, the after-states as black dots and fully-specified after-states as black triangles. The action arcs are labelled by their actions (e.g., a^1 and 'wildcard' $*^2$) whereas influence arcs are labelled by external state transitions ($s^2_4 \rightarrow s^2_6$). Furthermore, although Definition 1 captures the general case for fully specified transitions, often it suffices to consider only transitions $(s^i \cup F^{e\setminus i}, \vec{a}^e, \hat{s}^i \cup \hat{F}^{e\setminus i})$, where $F^{e\setminus i}$ is the subset of state features on which the reward functions assigned to agent i depend. This again is an abstraction of transitions: only influence arcs regarding the feature need to be included, therefore requiring typically much less arcs. An example of this is given later (Figure 2).

3.1 Bounding the optimal value

In addition to storing rewards compactly, we can use the CRGs to bound the optimal value. Specifically, the maximal (resp. minimal) attainable return from a joint state s_t onwards, is an upper (resp. lower) bound on the value. Moreover, the sum of bounds on local returns provides a bound on global return and thus on the optimal value. We define these bounds recursively:

$$U(s^i) = \max_{\tau^e \in \phi_i(s^i)} \Big[\mathcal{R}_i(\tau^e) + U(\hat{s}^i_\tau)\Big] \tag{4}$$

such that $\phi_i(s)$ is the set of available local transitions $\tau^e = (s^e_\tau, \vec{a}^e_\tau, \hat{s}^e_\tau)$ from state s^i in the CRG ϕ_i. The lower bounds $L(s^i)$ are defined accordingly over minimal returns. Finally, the bound on the global optimal value at a joint state s is

$$B(s) = \sum_{i \in N} B(s^i), \tag{5}$$

and for a joint transition $(s, \vec{a}, \hat{s})$,

$$B(s, \vec{a}, \hat{s}) = \sum_{i \in N} \mathcal{R}_i(s, \vec{a}, \hat{s}) + B(\hat{s}^i), \tag{6}$$

where B is either an upper or a lower bound.

3.2 Conditional Reward Independence

Furthermore, CRGs can be used to exploit independence in the local reward function as a result of past decisions. In many planning problems, actions can be performed only a limited amount of times and thereafter reward interactions involving that action no longer occur. When an agent can no longer perform its actions that may have a reward-interaction with actions of other agents, the expected value of the remaining decisions can be found through local optimisation. Moreover, when dependencies between two groups of agents no longer occur, the policy search space can be decoupled into independent components for which a policy may be found separately while their combination is still globally optimal. Let θ_t be the current execution sequence up to time t, then we denote the set of joint actions that are still available after completing this sequence by $A|\theta_t$ and we define:

Definition 2 (Conditional Reward Independence). *Given an execution sequence θ_t, two agents $i, j \in N$ are conditionally reward independent, denoted $CRI(i, j, \theta_t)$, if for all states $s, \hat{s} \in S$, and every joint action $\vec{a} \in A|\theta_t$ still available given execution history θ_t:*

$$\forall e \subseteq N \text{ s.t. } \{i, j\} \subseteq e \colon R^e(s, \vec{a}, \hat{s}) = 0.$$

Although generally reward independence is concluded from joint execution sequence θ_t, in some cases independence can be determined based on the local execution sequence θ^i_t only, for example when an agent i has completed its own set of dependent actions. Such *local conditional reward independence* occurs when $\forall j \in N : CRI(i, j, \theta^i_t)$, where $A|\theta^i_t$ contains all joint actions except those including actions no longer possible for agent i. Local reward independence is established during CRG generation and, from a state s^i that is flagged independent, we compute the optimal policy $\pi^*_i(s^i)$ and include only the optimal transitions in the remainder of the CRG.

3.3 Conditional Return Policy Search

All of the previous comes together in our *Conditional Return Policy Search (CoRe)*. CoRe performs a branch-and-bound search over the joint policy space, represented as a DAG with nodes s_t and edges $\langle \vec{a}_t, s_{t+1} \rangle$, such that a joint policy corresponds to selecting a subset of edges (corresponding to the same $\vec{a}_t$). This search is made effective by using the returns and bounds compactly stored in the CRGs for fast policy evaluation and pruning, and by exploiting the conditional reward independence between subsets of agents.

Algorithm 1: $\texttt{CoRe}(\Phi, \theta^N_t, h, N)$

```
   Input: The set of CRGs $\Phi$, execution sequence $\theta^N_t$, the
          horizon $h$, and a (sub)set of agents $N$
1  if $t = h$ then return 0;
2  $V^* \leftarrow 0$
3  foreach $N' \in \texttt{SCC}(\texttt{N}, \theta^{\texttt{N}}_{\texttt{t}})$ do
       // Compute weighted sums of bounds $B(s_{\theta,t}, \vec{a}_t, s_{t+1})$:
4      $\forall \vec{a}_t : U(s_{\theta,t}, \vec{a}_t) \leftarrow \sum_{s_{t+1}} T(s_{\theta,t}, \vec{a}_t, s_{t+1}) U(s_{\theta,t}, \vec{a}_t, s_{t+1})$
5      $L_{max} \leftarrow \max_{\vec{a}_t} \sum_{s_{t+1}} T(s_{\theta,t}, \vec{a}_t, s_{t+1}) L(s_{\theta,t}, \vec{a}_t, s_{t+1})$
       // Find joint action maximising expected reward
6      foreach $\vec{a}_t$ for which $U(s_{\theta,t}, \vec{a}_t) \geq L_{max}$ do
7          $V_{\vec{a}_t} \leftarrow 0$
8          foreach $s_{t+1}$ reachable from $s_{\theta,t}$ and $\vec{a}_t$ do
9              $V_{\vec{a}_t} \mathrel{+}= T(s_{\theta,t}, \vec{a}_t, s_{t+1}) \Big( R(s_{\theta,t}, \vec{a}_t, s_{t+1}) +$
                       $\texttt{CoRe}(\Phi, \theta^N_t \oplus [\vec{a}_t, s_{t+1}], h, N') \Big)$
10         $L_{max} \leftarrow \max(V_{\vec{a}_t}, L_{max})$ ;      // update lower bound
11     $V^* \mathrel{+}= \max_{\vec{a}_t} V_{\vec{a}_t}$
12 return $V^*$
```

Before running CoRe, however, first the CRGs ϕ_i are constructed for the local rewards $\mathcal{R}_i$ of each agent $i \in N$. The interaction rewards $R^e_i \in \mathcal{R}^i$ are partitioned heuristically over the agents. In our experiments we used a heuristic that visits the agents with the most interaction rewards in descending order. When an agent is visited we collect all its attached interaction reward functions. An interaction reward function is assigned to this agent when there are no other agents left in its scope that are unvisited. In preliminary experiments we established this to be a reasonable heuristic; finding a better (possibly problem or domain specific) heuristic is left for future work.

The generation of the CRGs is done according to Definition 1 using a recursive procedure that for each local state and local transition generates the corresponding after-states and fully-specified after-states, connected by action arcs and appended by influence arcs. When actions cause no reward interaction, they are combined using the wildcard label $*$. Finally, from every fully-specified after-state that may possibly cause a reward interaction, the influence arcs are generated, again grouping non-interactive influences using a wildcard-labelled arc $\varnothing$. Additionally, during generation we compute and store bounds (Eqs. 5 and 6) and flag local states that are locally conditionally reward independent. The procedure to construct CRGs assesses all trajectories of the *local* planning problem, including external influences, and its computational complexity is therefore exponential in the (augmented) local problem. In practice, however, its runtime is negligible compared to that required by policy search and will scale much better when the problem size increases.

After the construction of the CRGs, the CoRe algorithm performs depth-first policy search (Algorithm 1) over the (sub)set of agents N. CoRe finds an optimal joint action for every *disjoint* subset of agents N' in which reward dependencies may still occur (line 3). These subsets are found using a connected component algorithm $\texttt{SCC}$ on a graph with nodes N and an edge (i, j) for every pair of dependent agents $i, j \in N$ for which $\neg CRI(i, j, \theta^N_t)$. In the evaluation of each subset N' (lines 3 to 11), we only consider state

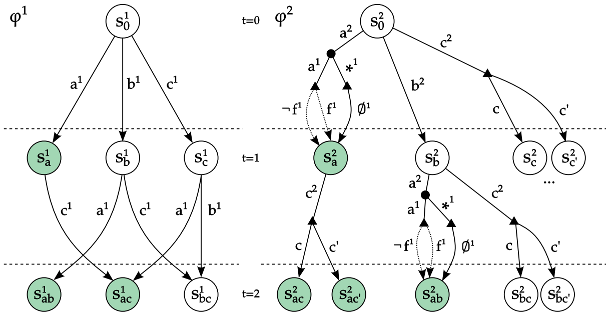


Figure 2: The CRGs of the two agents. We have omitted the branches for a^2 and b^2 from states s_c^2 and $s_{c'}^2$. The highlighted state nodes illustrate locally reward independent states.

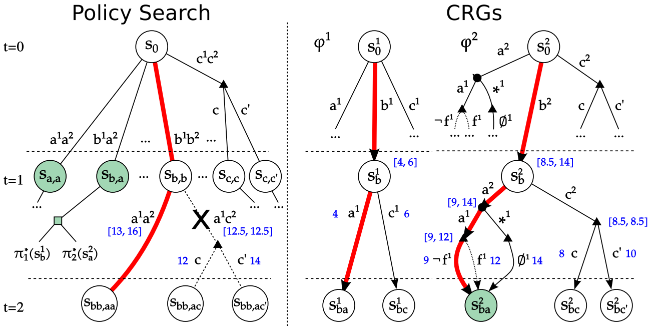


Figure 3: Example of policy evaluation. The left graph shows (a part of) the policy search tree with joint states and joint actions, and the right graph the CRGs per agent. One possible execution sequence θ_h is highlighted in thick red.

space $S^{N'} \subseteq S$ and joint actions $\vec{a} \in A^{N'}$, but to preserve readability these superscripts have been omitted in the algorithm. Lines 4 and 5 determine the upper and lower bounds for this subset of agents, retrieved from the CRGs, which are used to prune suboptimal joint actions in line 6. For the remaining joint actions, CoRe recursively computes the expected value by extending the current execution sequence with the joint action $\vec{a}_t$ and all possible result states s_{t+1} and summing over their returned expected values times the probability of each transition (line 9) and it returns the one that maximises expected value (line 11). Note that after evaluation the lower bound can be tightened (line 10). Finally, the optimal expected values over all agent subsets are combined and returned in line 12.

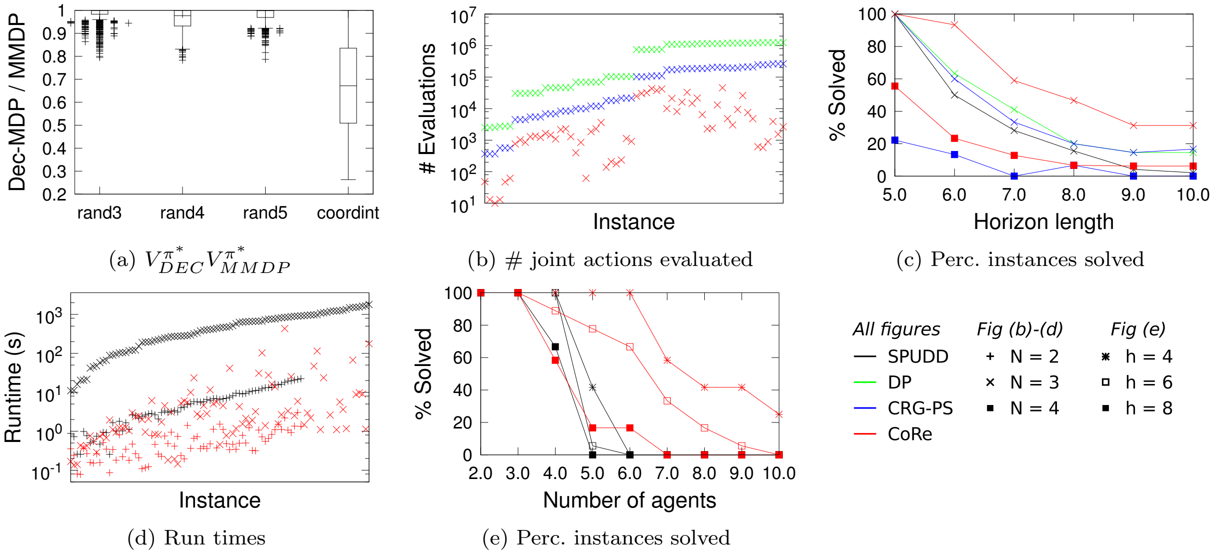
4. CORE EXAMPLE

To illustrate the CoRe algorithm, we use an example two-agent problem in which both agents can perform actions a, b and c, but every action can be performed only once. Action c^2 of agent 2 is (for ease of exposition) the only stochastic action and has possible outcomes c and c' with probabilities 0.75 and 0.25, respectively. In this simple problem there is only one interaction between actions a^1 and a^2, and the reward depends on feature f^1 of agent 1 being set from 'unset' (f^1?) to true (f^1) or false ($\neg f^1$). Therefore we have one interaction reward function with two unique rewards $R^{1,2}(f^1?, \{a^1, a^2\}, f^1)$ and $R^{1,2}(f^1?, \{a^1, a^2\}, \neg f^1)$. Additionally, we have local reward functions R^1 and R^2, and a planning horizon of two time steps. Without specifying any actual rewards, we demonstrate the CRG construction and algorithm using this example.

Figure 2 illustrates the two CRGs. On the left is the CRG ϕ^1 of agent 1 that only includes its local reward R^1. The CRG of agent 2 includes the reward interaction function $R^{1,2}$ in addition to its local reward R^2. Notice that only when sequences start with action a^2, additional arcs have to be included in CRG ϕ^2 to account for a possible reward interaction. The sequence that starts with a^2 is followed by an after-state node with two arcs: one for the case where agent 1 is performing a^1 and one for the case where it is not, i.e., it performs either b^1 or c^1, denoted by the wildcard $*^1$. Finally, the interaction reward depends also on whether feature f^1 is (stochastically) set to true or not from its initial value f^i?, concisely labeled f^1 and $\neg f^1$ in this illustration. As the interaction reward only occurs when $\{a^1, a^2\}$ is executed, the fully-specified after-state node (the triangle below the arcs a^2 and $*^1$) has only the influence arc $\varnothing^1$. All other transitions are reward independent and thus captured by a local transition arc $(s^i, a^i, \hat{s}^i)$. Locally reward independent states are highlighted green and from each of these states only the optimal action transition is kept in the CRG. In this example, only action arc c^1 is included from s_a^1, which was determined optimal from the local state, and no arc for b^1 is included.

An example of the CoRe algorithm in progress is shown in Figure 3, with the policy search space on the left, and the CRGs on the right, now annotated with return bounds. Note that only a couple of the branches of the full DAG, as well as the CRGs, are shown to preserve clarity. While there are 9 joint actions possible with 12 result states for $t = 0$, the CRGs need only 3 + 4 states and 3 + 6 transitions to represent all rewards. Because for ϕ^2, neither action b^1 or c^1 of agent 1 can result in a reward in R^2 or $R^{1,2}$, we can represent the reward by the same transition $\varnothing^1$. The current execution path θ_h that is evaluated by CoRe is highlighted in thick red. This sequence starts with non-dependent actions $\{b^1, b^2\}$, resulting in joint state $s_{b,b}$ (ignore the bounds in blue for now). The execution sequence at $t = 1$ is thus $\theta_1 = [s_0, \{b^1, b^2\}, s_{b,b}]$. In the CRGs we have highlighted the corresponding transitions to states s_b^1 and s_b^2. For $t = 2$ CoRe is evaluating joint action $\{a^1, a^2\}$ that is reward-interacting and therefore the value of state feature f^1 is required to determine the transition in ϕ^2, here chosen arbitrarily as $\neg f_x^1$. The corresponding execution sequence (in agent 2) is therefore $\theta_2^2 = [s_0^2, \{b^1, b^2\}, s_b^2 \cup \{f^1?\}, \{a^1, a^2\}, s_{ba}^2 \cup \{\neg f^1\}]$ (remember that we are 'setting' f^1). Had agent 1 performed another action, we would have traversed the branch $\varnothing^1$ that directly leads to new state s_{ba}^2 as no reward interaction occurs for its other actions.

Branch-and-bound is shown for the evaluation at global state $s_{b,b}$, where the rewards (over the transitions) and their bounds (at the nodes) are shown as blue figures. The bounds stored for joint actions $\{a^1, a^2\}$ and $\{a^1, c^2\}$ are $[13, 16]$ and $[12.5, 12.5]$ respectively, found by summing over the local bounds stored in the CRGs. As a result, action $\{a^1, c^2\}$ can be pruned. Note that we can compute the expected value of $\{a^1, c^2\}$ exactly in the CRG, but not that of $\{a^1, a^2\}$. This is because agent 2 knows the transition probability of action c^2 but it does not know what value f^1 has during CRG generation or with what probability a^1 will be performed. Still we can bound the return of action a^2 over all possible



(a) $V_{DEC}^{\pi^*} V_{MMDP}^{\pi^*}$

(b) # joint actions evaluated

(c) Perc. instances solved

(d) Run times

(e) Perc. instances solved

Figure 4: Experimental results

feature values, stored in ϕ^2, and they can be updated as the probabilities become known during policy search.

Finally, conditional reward independence between agents occurs in the policy search tree states highlighted in green. After execution of joint action $\{b^1, a^2\}$ no more reward interactions can take place between the agents and therefore we can decouple the problem into two independent sub problems. From state $s_{b,a}$ CoRe finds optimal policies $\pi_1^*(s_b^1)$ and $\pi_2^*(s_a^2)$ and combines them into the optimal joint policy $\pi^*(s_{b,a}) = \langle \pi_1^*(s_b^1), \pi_2^*(s_a^2) \rangle$ for that state.

5. EVALUATION

In our experiments we find optimal policies for instances of the *maintenance planning problem (MPP)*, in which agents are responsible for completing a set of one-time maintenance tasks with uncertain duration within the planning horizon [16]. In this problem, the agents represent contractors that participate in an allocation mechanism that optimises the social welfare of all agents – the expected value of the joint policy – based on their *reported* costs. Important is that only when the joint plan is optimal the agents have an incentive to report truthfully and thus optimal planning is required.

The goal in MPP is to find a contingent maintenance schedule (i.e., policy) that minimises both individual maintenance costs as well as economic loss due to traffic hindrance caused by joint maintenance. Agents maintain a state with start and end times of their maintenance tasks. The individual maintenance costs are task and time dependent, while traffic hindrance is modelled through interaction rewards. Using this domain we conduct three experiments with CoRe to study 1) the expected value when solving centrally versus decentralised methods, 2) the impact on the number of joint actions evaluated and 3) the scalability in terms of agents.

For the first experiment we generated 3 sets of 1000 random two-agent, two-activity instances with planning horizons $3, 4$ and 5 (**rand[h]**) and a set of 1000 coordination-intensive `coordint` instances where decentralised policies are likely to result in poor values. For these (all relatively easy to solve) instances we compared the value of the optimal decentralised Dec-MDP policy [17] against the value of optimal CoRe policies to confirm that decentralised policies are suboptimal. Figure 4a shows the ratio $V_{DEC}^{\pi^*} / V_{MMDP}^{\pi^*}$. In the **rand[h]** instances, the expected value of the optimal decentralised policy is equal to the optimal MMDP value for approx. half of the instances. However, for the coordination-intensive instances `coordint` the value of a decentralised policy is much lower: an agent following a decentralised policy cannot react directly to the outcome of another agent's action. The activity causes additional hindrance unless the policy reacts to this delay. Here the decentralised policy value never matches the optimal MMDP policy value: on average it is about 33% less but this loss can be as bad as 75%.

For our remaining experiments we thus focus only on fully-observable approaches and for this we generated a representative random test set **mpp**, consisting of 2, 3 and 4 agent problems (400 each), with 3 maintenance tasks, varying in planning horizon, task length and delay probabilities and reward interactions of the form $R^{i,i+1}$ for all $i = 1, 2, \ldots, n-1$. On these instances we compare CoRe against the current state-of-the-art method for optimal MPP planning from [16], solving a compact encoding of the problem using SPUDD [7], and a dynamic programming algorithm with domain knowledge. Finally, we included a CRG policy search algorithm without value bounds (CRG-PS) to investigate the effectiveness of branch-and-bound.

Figure 4b shows the search space size reduction that can be achieved using CRGs in this domain. Our CRG-enabled algorithm (CRG-PS, blue) is able to reduce the number of evaluated joint actions by about one order of magnitude compared to the domain-tailored dynamic programming method (DP, green). Furthermore, we can observe that when value bounds are used (CoRe, red), this number can be reduced even more, although its effect varies per instance.

Having seen the search space reduction CoRe can achieve, we compare to the best-known solution for MPP from [16]. We analyse the percentage of problems of varying horizon length from the **mpp** test set that both approaches are able to solve within a fixed time limit of 30 minutes in Figure 4c (all two-agent instances were solved and thus omitted). We observe that the CoRe algorithm is able to solve more instances than SPUDD (black) of the 3 agent problems (labelled $N = 3$). For 4 agent problems, only CRG-PS and CoRe are able to solve any instances. The main reason for this is that the CRGs successfully exploit the conditional action independence that decouples the agents *for most of the planning decisions*. Only in cases where reward interactions occur, actions are coordinated between the agents, whereas SPUDD always coordinates every joint decision.

We also compare the performance of both algorithms in terms of runtime. As CoRe achieves a greater coverage than SPUDD, we compare run times for all instances that have been successfully solved by the latter (Figure 4d). We have ordered the instances based on their SPUDD runtime and plot both the SPUDD and CoRe runtime (including CRG generation). Almost all of the instances solved by SPUDD are solved faster by CoRe in both the 2 as well as the 3 agent setting. CoRe failed to solve 3.4% of the instances solved by SPUDD whereas SPUDD failed 63.9% of the instances that CoRe was able to solve.

Finally, to study the agent-scalability of CoRe, we generated a test set `pyra` with a pyramid-like reward interaction structure: every first action of the k-th agent depends on the first action of agent $2k$ and agent $2k + 1$. Figure 4e shows the percentage of instances from the `pyra` test set that are solved by SPUDD and CoRe for various problem horizons. Whereas previous state-of-the-art could solve instances up to only 5 agents, CoRe successfully solved about a quarter of the 10 agent problems ($h = 4$) and overall solves many of the previously unsolvable instances.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we focus on optimally solving fully-observable, stochastic planning problems where agents are dependent only through interaction rewards. When such interactions are limited, we can partition individual and interaction rewards over agents and store them compactly in *conditional return graphs*. We propose a conditional return policy search algorithm (CoRe) that uses CRGs to greatly reduce the policy search space that needs to be evaluated to produce an optimal policy, based on three key insights: 1) when interactions are sparse, the number of unique returns per agent is relatively small and can be stored efficiently, 2) CRGs can maintain bounds on the return, and thus the expected value, to guide our search, and 3) in the presence of conditional reward independence, i.e., the absence of further reward interactions, we can decouple agents during policy search.

Our experiments show that this reduction can be by orders of magnitude in the maintenance planning domain. This enabled CoRe to solve instances that were previously deemed unsolvable. In addition, CoRe is almost always able to produce solutions faster than the previously known best approach. Moreover, CoRe is able to solve 10-agent instances when the reward structure exhibits a high level of conditional reward independence, whereas previous methods did not scale beyond 5 agents. Finally, our experiments show that decentralised MDP methods, for which many scalable approaches in terms of agents and reward structures have been researched, lead to suboptimal policies in the fully-observable setting.

Although we focused on transition-independent MMDPs, CRGs can be employed to solve general MMDPs when transition dependencies are sparse. This would require including dependent-state transitions in the CRGs that are similar to reward-interaction paths. The extension to general MMDPs is considered future work. Other interesting avenues include exploiting conditional reward independence between agents during joint action generation and bounded approximations.

Furthermore, our work could be extended to apply to multi-objective MMDPs, i.e., settings in which the agents care about multiple goals. For example, in the MPP setting the objectives would be to minimise traffic hindrance while maximising revenue [15].

Acknowledgements

Joris Scharpff is supported by Next Generation Infrastructures and Almende BV, Diederik M. Roijers is supported by the NWO DTC-NCAP (#612.001.109) project, and Frans Oliehoek is supported by NWO Innovational Research Incentives Scheme Veni (#639.021.336).

REFERENCES

[1] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 302–309, 2004.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 41–48, 2003.

[3] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proc. of the International Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

[4] R. Cavallo, D. C. Parkes, and S. Singh. Optimal coordinated planning amongst self-interested agents with private state. In *Proc. of Uncertainty in Artificial Intelligence*, 2006.

[5] C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored MDPs. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 253–259, 2002.

[6] C. E. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems*, pages 1523–1530, 2001.

[7] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. *Proc. of Uncertainty in Artificial Intelligence*, 1999.

[8] J. R. Kok and N. Vlassis. Sparse cooperative Q-learning. In *Proc. of the International Conference on Machine Learning*, pages 481–488, 2004.

[9] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

[10] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. L. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National Conference on*

Artificial Intelligence, pages 165–172, 1998.

[11] H. Mostafa and V. Lesser. Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *Proc. of the International Joint Conference on Web Intelligence and Intelligent Agent Technologies*, volume 2, pages 193–200, 2009.

[12] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

[13] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 517–524, May 2008.

[14] F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Approximate solutions for factored Dec-POMDPs with many agents. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, pages 563–570, 2013.

[15] D. M. Roijers, J. Scharpff, M. T. J. Spaan, F. A. Oliehoek, M. De Weerdt, and S. Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In *Proc. of the International Conference on Automated Planning and Scheduling*, pages 262–270, 2014.

[16] J. Scharpff, M. T. J. Spaan, M. de Weerdt, and L. Volker. Planning under uncertainty for coordinating infrastructural maintenance. In *Proc. of the International Conference on Automated Planning and Scheduling*, pages 425–433, 2013.

[17] M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 2027–2032, 2011.

[18] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, May 2007.

[19] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *Proc. of the International Conference on Automated Planning and Scheduling*, pages 185–192, 2010.