Solving Transition-Independent Multi-agent MDPs with Sparse Interactions (Extended version)*

Joris Scharpff[1], Diederik M. Roijers[2], Frans A. Oliehoek[2,3], Matthijs T. J. Spaan[1], Mathijs M. de Weerdt[1]

[1] Delft University of Technology, The Netherlands
[2] University of Amsterdam, The Netherlands
[3] University of Liverpool, United Kingdom

Abstract

In cooperative multi-agent sequential decision making under uncertainty, agents must coordinate to find an optimal joint policy that maximises joint value. Typical algorithms exploit additive structure in the value function, but in the fully-observable multi-agent MDP (MMDP) setting such structure is not present. We propose a new optimal solver for transition-independent MMDPs, in which agents can only affect their own state but their reward depends on joint transitions. We represent these dependencies compactly in *conditional return graphs (CRGs)*. Using CRGs the value of a joint policy and the bounds on 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 finds solutions to previously unsolvable problems.

1 Introduction

When cooperative teams of agents are planning in uncertain domains, they must coordinate to maximise their (joint) team value. In several problem domains, such as traffic light control [Bakker et al., 2010], system monitoring [Guestrin et al., 2002a], multi-robot planning [Messias et al., 2013] or maintenance planning [Scharpff et al., 2013], the full state of the environment is assumed to be known to each agent. Such *centralised* planning problems can be formalised as multi-agent Markov decision processes (MMDPs) [Boutilier, 1996], in which the availability of complete and perfect information leads to highly-coordinated policies. However, these models suffer from exponential joint action spaces as well as a state that is typically exponential in the number of agents.

In problem domains with local observations, sub-classes of *decentralised* models exist that admit a value function that is exactly factored into additive components [Becker et al., 2003, Nair et al., 2005, Witwicki and Durfee, 2010] and more general classes admit upper bounds on the value function that are factored [Oliehoek et al., 2015]. In centralised models however, the possibility of a factored value function can be ruled out in general: by observing the full state, agents can predict the actions of others

*This article is an extended version of the paper that was published under the same title in the Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI16), held in Phoenix, Arizona USA on February 12-17, 2016. The most significant difference is that here a more strict definition of dependent actions, transition influence and, consequentially, the conditional return graphs is given. Furthermore, this version contains additional details and explanations that did not make it into the conference paper due to the page limit.

better 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.

A class of problems that exhibits particular structure is that of task-based planning problems, such as the *maintenance planning problem* (MPP) from [Scharpff et al., 2013]. In the MPP every agent needs to plan and complete its own set of road maintenance tasks at minimal (private) maintenance cost. Each task is performed only once and may delay with a known probability. As maintenance causes disruption to traffic, agents are collectively fined relative to the (super-additive) hindrance from their *joint actions*. Although agents plan autonomously, they depend on others via these fines and must therefore coordinate. Still, such *reward interactions* are typically sparse: they apply only to certain combinations of maintenance tasks, e.g. in the same area, and often involve only a few agents. Moreover, when an agent has performed its maintenance tasks that potentially interfere with others, it will no longer interact with any of the other agents.

Our main goal is to identify and exploit such structure in centralised models, for which we consider *transition independent* MMDPs (TI-MMDPs). In TI-MMDPS, agent rewards depend on joint states and actions, but transition probabilities are individual. Our key insight is that we can exploit the reward structure of TI-MMDPs by decomposing the *returns* of all execution histories (i.e., all possible state/action sequences from the initial time step to the planning horizon) into components that depend on local states and actions.

We build 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 states and actions of execution sequences. This allows a compact representation of rewards and efficiently computable bounds on the optimal policy value via a data structure we call the *conditional return graph* (CRG). 2) In TI-MMDPs agent interactions are often sparse and/or local, for instance in the domains mentioned initially, typically resulting in very compact CRGs. 3) In many (e.g. task-modelling) problems the state space is transient, i.e., states can only be visited once, leading to a directed, acyclic transition graph. With our first two key observations this often gives rise to *conditional reward independence*, i.e. the absence of further reward interactions, and enables agent decoupling during policy search.

Here we propose *conditional return policy search* (CoRe), a branch-and-bound policy search algorithm for TI-MMDPs employing CRGs, and show that it is effective when reward interactions between agents are sparse. We evaluate CoRe on instances of the aforementioned MPP with uncertain outcomes and very large state spaces. We demonstrate that CoRe evaluates only a fraction of the policy search space and thus finds optimal policies for previously unsolvable instances and commonly requires less runtime than its alternatives.

2 Related work

Scalability is a major issue in multi-agent planning under uncertainty. In response to this challenge, two important lines of work have been developed. One line of work proposed approximate solutions by imposing and exploiting an additive structure in the value function [Guestrin et al., 2002a]. This approach has been applied in a range of stochastic planning settings, fully and partially observable alike, both from a single-agent perspective [Koller and Parr, 1999, Parr, 1998] and multi-agent [Guestrin et al., 2002b, Meuleau et al., 1998, Kok and Vlassis, 2004, Oliehoek et al., 2013b]. The drawback of such methods is that typically no bounds on the efficiency loss can be given. We focus on optimal solutions, required to deal with strategic behaviour in a mechanism [Cavallo et al., 2006, Scharpff et al., 2013].

This is part of another line of work that has not sacrificed optimality, but instead targets sub-classes of problems with properties that can be exploited [Becker et al., 2003, Becker et al., 2004, Mostafa and Lesser, 2009, Witwicki and Durfee, 2010]. In particular, several methods that exploit the same type of additive structure in the value function have been shown exact, simply because value functions of the sub-class of problems they address are guaranteed to have such shape [Nair et al., 2005, Oliehoek

et al., 2008, Varakantham et al., 2007]. However, all these approaches are for decentralised models in which actions are conditioned only on *local* observations. Consequentially, optimal policies for decentralised models typically yield lower value than the optimal policies for their fully-observable counterparts (shown in our experiments).

Our focus is on transition-independent problems, suitable for multi-agent problems in which the effects of activities of agents are (assumed) independent. In domains where agents directly influence each other, e.g., by manipulating shared state variables, this assumption is violated. Still, transition independence allows agent coordination at a task level, as in the MPP, and is both practically relevant and not uncommon in literature [Becker et al., 2003, Spaan et al., 2006, Melo and Veloso, 2011, Dibangoye et al., 2013].

Another type of interaction between agents is through limited (global) resources required for certain actions. While this introduces a global coupling, some scalability is achievable [Meuleau et al., 1998]. Whether context-specific and conditional agent independence remains exploitable in the presence of such resources in TI-MMDPs is yet unclear. Additionally, there exist also methods that exploit reward sparsity and independence but through a reinforcement learning approach identifying 'interaction states' [De Hauwere et al., 2012, Melo and Veloso, 2009]. Although these target a similar structure, learning implies that there are no guarantees on the solution quality (until all states have been recognised as interaction states, which implies a brute-force solve of the MMDP).

3 Model

We consider a (fully-observable) *transition-independent*, *multi-agent Markov decision process*, or *TI-MMDP*, with a finite horizon of length h, and no discounting of rewards.

Definition 1. *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 agent-factored state space, which is the Cartesian product of* n *factored states spaces* S^i *(composed of 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*;*

$T(s, \vec{a}, \hat{s}) = \prod_{i \in N} T^i(s^i, a^i, \hat{s}^i)$ *defines a transition probability, which is the product of the local transition probabilities due to transition independence; and*

$\mathcal{R}$ *is the set of reward functions over transitions that we assume without loss of generality is structured as* $\{R^e | e \subseteq N\}$*. When* $e = \{i\}$*,* R^i *is the* local reward *function for agent* i*, and when* $|e| > 1$*,* R^e *is called an* interaction reward*. 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 all rewards:*

$$R(s, \vec{a}, \hat{s}) = \sum_{R^e \in \mathcal{R}} R^e(\{s^j\}_{j \in e}, \{\vec{a}^{\,j}\}_{j \in e}, \{\hat{s}^j\}_{j \in e}). \tag{1}$$

Two agents i and j are called *dependent* when there exists a reward function with both agents in its scope, e.g., a two-agent reward $R^{i,j}(\{s^i, s^j\}, \{a^i, a^j\}, \{\hat{s}^i, \hat{s}^j\})$ could describe the super-additive hindrance that results when agents in the MPP do concurrent maintenance on two nearby roads. We focus on problems with *sparse interaction rewards*, i.e., reward functions R^e with non-zero rewards for a small subset of the local joint actions (e.g., $\mathcal{A}^{ij} \subset A^i \times A^j$) or only a few agents in its scope. Of course, sparseness is not a binary property: the maximal number of actions with non-zero interaction rewards

and participating agents (respectively α and w in Theorem 1) determine the level of sparsity. Note that this is not a restriction but rather a classification of problems that benefit most from our approach.

The goal in a TI-MMDP is to find the optimal joint policy π^* of which the actions $\vec{a}$ maximise the expected sum of rewards, expressed by the Bellman equation:

$$V^*(s_t) = \max_{\vec{a}_t} \sum_{s_{t+1} \in S} T(s_t, \vec{a}_t, s_{t+1}) \Big(\sum_{R^e \in \mathcal{R}} R^e(s^e_t, \vec{a}^e_t, s^e_{t+1}) + V^*(s_{t+1}) \Big). \tag{2}$$

At the last timestep there are no future rewards, so $V^*(s_h) = 0$ for every $s_h \in S$. Although $V^*(s_t)$ can be computed through a series of maximisations over the planning period, e.g. via dynamic programming [Puterman, 2014], it cannot be written as a sum of independent local value functions without losing optimality [Koller and Parr, 1999], i.e. $V^*(s) \neq \sum_e V^{e,*}(s^e)$.

Instead, we factor the *returns* of *execution sequences*, the sum of rewards obtained from following state/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 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:

$$Z(\theta_t) = \sum_{R^e \in \mathcal{R}} \sum_{x=0}^{t-1} R^e(s^e_{\theta,x}, \vec{a}^e_{\theta,x}, s^e_{\theta,x+1}), \tag{3}$$

where $s^e_{\theta,x}$, $\vec{a}^e_{\theta,x}$ and $s^e_{\theta,x+1}$ denote local states and actions from θ_t that are relevant for R^e. Contrary to the optimal value function, (3) is additive in the reward components and can thus be computed locally.

Nonetheless, the return of (3) does not directly give us the value of an optimal policy. To compute the expected policy value using (3), we sum the expected return of all future execution sequences θ_h reachable under policy π starting at s_0 (denoted $\theta_h | \pi, s_0$):

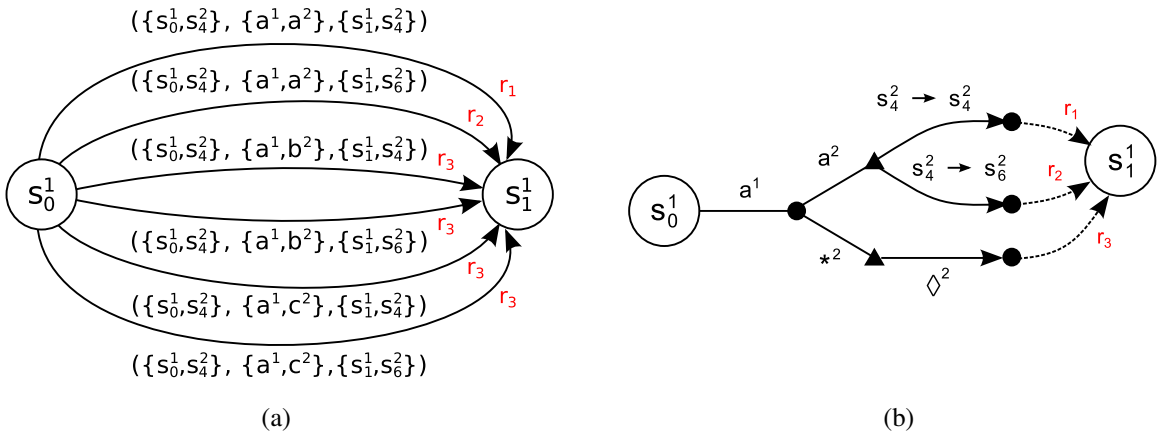
$$V^\pi(s_0) = \sum_{\theta_h | \pi, s_0} Pr(\theta_h) Z(\theta_h) = \sum_{\theta_h | \pi, s_0} Z(\theta_h) \prod_{t=0}^{h-1} T(s_{\theta,t}, \pi(s_{\theta,t}), s_{\theta,t+1}). \tag{4}$$

Now, (4) is structured such that it expresses the value in terms of additively factored terms ($Z(\theta_h)$). However, comparing (2) and (4), we see that the price for this is that we no longer are expressing the *optimal* value function, but that of a given policy π. In fact, (4) corresponds to an equation for *policy evaluation*. It is thus not a basis for dynamic programming, but it is usable for policy search. Although policy search methods have their own problems in scaling to large problems, we show that the structure of (4) can be leveraged.

In particular, because the return of an execution sequence can be decomposed into additive components (Eq. 3), we decouple and store these returns locally in *conditional return graphs* (CRGs). The CRG is used in policy search to efficiently compute (4) 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, when constructing CRGs it is possible to detect the absence future reward interactions between agents and thus optimally decouple the policy search.

4 Conditional Return Graphs

We now partition the reward function into additive components $\mathcal{R}_i$ and assign them to agents. The *local* reward for an agent $i \in N$ is given by $\mathcal{R}_i = \{R^i\} \cup \mathcal{R}^e_i$, where $\mathcal{R}^e_i$ are the interaction rewards assigned



(a) (b)

Figure 1: Example of a transition for one agent of a two-agent problem where (a) shows the complete state/transition graph with unique rewards r_x and (b) the equivalent but more compact CRG when R_1 only depends on a_1^2.

to i (restricted to R^e where $i \in e$). The sets $\mathcal{R}_i$ are disjoint sub-sets of the reward functions $\mathcal{R}$ such that together they again form the complete set of joint reward functions. Note that such a partitioning can be done in many ways. In our preliminary experiments we observed that for the maintenance planning domain balancing the number of functions per agent works well. Nevertheless, further study is required to establish potentially better or more generic assignment heuristics.

Given such a disjoint partition of rewards, a conditional return graph for agent i is a data structure that represents all possible *local* returns, for all possible *local* execution histories. Particularly, it is a directed acyclic graph (DAG) with a layer for every stage $t = 0, \ldots, h-1$ of the decision process. Each layer contains nodes corresponding to the reachable *local states* $s^i \in S^i$ of agent i at that stage. As the goal is to include interaction rewards, the CRG includes for every local state s^i, local action a^i, and successor state $\hat{s}^i$ a representation of all transitions $(s^e, \vec{a}^{\,e}, \hat{s}^e)$ for which $s^i \in s^e$, $a_i \in \vec{a}^{\,e}$, and $\hat{s}^i \in \hat{s}^e$.

While a direct representation of these transitions captures all the rewards possible, we can achieve a much more compact representation by exploiting sparse interaction rewards, enabling us to group many joint actions $\vec{a}^{\,e}$ leading to the same rewards. Consider a two-agent example $N = \{1, 2\}$ with actions $A^1 = \{a^1\}$ and $A^2 = \{a^2, b^2, c^2\}$ respectively. Both agents have a local reward function, resp. R^1 and R^2, and there is one interaction reward $R^{1,2}$ that is 0 for all transitions but the ones involving joint action $\{a^1, a^2\}$. This could for example be a network cost function of MPP that is non-zero when the maintenance activities corresponding to actions a^1 and a^2 are performed, e.g. because they take place within close proximity of each other. The interaction reward $R^{1,2}$ is assigned to agent 1, thus $\mathcal{R}_i^1 = \{R^i, R^{1,2}\}$ and of course $\mathcal{R}_i^2 = \{R^2\}$. A naive representation of all rewards would result in the graph of Figure 1a, illustrating a transition from s_0^1 to s_1^1 where agent 2 may remain in state s_4^2 or transition to state s_6^2.

Observe that now there are only three unique rewards (shown in red) whereas there are six possible transitions. Intuitively, all transitions resulting in the same reward should be grouped such that only when the actions/states of other agents influence the reward they should be included in the CRG, resulting in the graph of Figure 1b. To make this explicit, we denote a single transition for agents $e \subseteq N$ by $\tau^e = (\{s^j\}_{j\in e}, \{\vec{a}^{\,j}\}_{j\in e}, \{\hat{s}^j\}_{j\in e}) = (s^e, \vec{a}^{\,e}, \hat{s}^e)$. A local transition τ^i is said to be contained in τ^e, denoted $\tau^i \in \tau^e$, if $i \in e$, $s^i \in s^e$, $a^i \in \vec{a}^{\,e}$ and $\hat{s}^i \in \hat{s}^e$. Moreover the set set of all available (joint) transitions can be written as

$$\mathcal{T}^e = \{(s^e, \vec{a}^{\,e}, \hat{s}^e) \,|\, s^e, \hat{s}^e \in \{s^j\}_{j\in e}, \vec{a}^{\,e} \in \{\vec{a}^{\,j}\}_{j\in e}, T(s^e, \vec{a}^{\,e}, s^e) > 0\} \tag{5}$$

Now we can formalise the set of actions (state transitions follow briefly afterwards) of other agents

that may interact with the rewards $\mathcal{R}_i$, assigned to each of the CRGs, given a current local transition $\tau^i = (s^i, a^i, \hat{s}^i)$. An action a^j of agent $j \neq i$ is said to be *dependent* with respect to local transition τ^i if it occurs in one of the available joint transitions that contains τ^i, its presence influences the interaction reward and there is at least one other action of agent j that does *not* cause the same interaction reward. The last condition is included to prevent marking all actions as dependent when actually the interaction reward depends on the state transition of agent j. This leads to the following definition of *dependent actions*:

Definition 2 (Dependent Actions)**.** *The set of* dependent actions *of an agent $j \in N$ that may reward-interact with agent $i \neq j$ when agent i's local transition is $\tau^i = (s^i, a^i, \hat{s}^i)$ is given by:*

$$\begin{aligned}\mathcal{A}(\tau^i, j) = \{a^j \in A^j \mid \exists\, R^e \in \mathcal{R}_i,\, \exists\, \tau^e \in \mathcal{T}^e, \exists\, b^j \neq a^j \in A^j :\\ \tau^i \in \tau^e \wedge a^j \in \vec{a}^{\,e}\\ \wedge\, R^e(\tau^e) \neq R^e(\tau^e \setminus \{\tau^j\}), \qquad \textit{s.t. } \tau^j \in \tau^e\\ \wedge\, R^e(\tau^e) \neq R^e(s^e, \vec{a}^{\,e} \setminus \{a^j\} \cup \{b^j\}, \hat{s}^e)\}\end{aligned}$$

Actions by other agents that are not dependent with respect to a transition τ^i, i.e. the actions $A^j \setminus \mathcal{A}(\tau^i, j)$, are (made) anonymous in the CRG for agent i via 'wildcards' (e.g. $*^2$ of Figure 1b), since they do not influence the reward from the functions in R^i.

Besides actions, the interaction reward may also be affected by the state transitions of the other agents. This is captured by the *transition influence*.[1] Its definition is rather similar to that of dependent actions. For a state transition $s^j \to \hat{s}^j$ of an agent $j \neq i$ to be considered an influence with respect to local transition τ^i, both states must be part of a transition τ^j such that both τ^i and τ^j are contained in a joint transition τ^e, such a transition τ^j must have an impact on at least one interaction reward and there must exist at least one other transition of agent j that does not have the same interaction reward impact. The latter condition is, as before, to prevent all state transitions being marked an influence whereas the interaction reward depends solely on the action. This is formalised by the following definition:

Definition 3. *The set of state pairs of an agent j that may lead to a reward interaction when agent $i \neq j$ perform transition $\tau^i = (s^i, a^i, \hat{s}^i)$ and agent j performs action $a_j \in A_j$ concurrently, is known as the* (transition) influence*, defined as*

$$\begin{aligned}I^i(\tau^i, a^j) = \{(s^j,\hat{s}^j) \in S^j \times S^j \mid \exists\, R^e \in \mathcal{R}_i, \exists\, \tau^e \in \mathcal{T}^e, \exists\, (s_2^j, \hat{s}_2^j) \neq (s^j, \hat{s}^j) \in S^j \times S^j :\\ \tau^i \in \tau^e \wedge \tau^j = (s^j, a^j, \hat{s}^j) \in \tau^e \qquad (6)\\ \wedge\, R^e(\tau^e) \neq R^e(\tau^e \setminus \{\tau^j\}), \qquad \textit{s.t. } \tau^j \in \tau^e \qquad (7)\\ \wedge\, R^e(\tau^e) \neq R^e(s^e \setminus \{s^j\} \cup \{s_2^j\}, \vec{a}^{\,e}, \hat{s}^e \setminus \{\hat{s}^j\} \cup \{\hat{s}_2^j\})\} \qquad (8)\end{aligned}$$

Finally, for any set of actions A^j of an agent j we define the transition influence of that set with respect to local transition τ^i as the union of all influences, or $I^i(\tau^i, A^j) = \bigcup_{a^j \in A^j} I^i(\tau^i, a^j)$. This last definition is useful to capture the influence of a wildcard set $*^j$, which occurs when multiple actions

[1] In [Scharpff et al., 2013] only the set of dependent actions was defined. Although the definition in the paper is correct, for some problems it may lead to an unnecessary blow-up of the CRG size. For instance, an interaction reward assigned to agent i may actually depend on the state transition of another agent j, regardless of the action it performs. By the previous definition all actions of agent i would have been marked dependent. Here we strengthen the previous definition by decoupling the influence of actions and state transitions on the interaction reward, such that the CRGs constructed based on these definitions are indeed of minimum size.

lead to the same state-transition interaction reward. Again, non-influencing state transitions can be grouped. We use the symbol $\diamond^j$ to denote the set of all non-influencing transitions of agent j given a local transition τ^i and action a^j (or wildcard $*^j$).

Given Defs. 2 and 5 above, a *conditional return graph* (CRG) ϕ_i for agent i can be defined as follows.

Definition 4 (Conditional Return Graph)**.** *Given a disjoint, complete partitioning $\mathcal{R} = \bigcup_{i \in N} \mathcal{R}_i$ of rewards over agents $i \in N$, the* Conditional Return Graph (CRG) *ϕ^i is a* directed acyclic graph *with for every stage t of the decision process a node for every reachable local state $s^i \in S$ and for every available local transition $\tau^i = (s^i, a^i, \hat{s}^i)$ a tree compactly representing all joint transitions $\tau^e = (s^e, \vec{a}^{\,e}, \hat{s}^e)$ of the agents $e \subseteq N$ in the scope of $\mathcal{R}_i$, or $e = \{i \in N \mid \exists R^e \in \mathcal{R}_i \colon i \in e\}$.*

The tree consists of two parts: an action tree *that specifies all dependent actions and an* influence tree *that contains the relevant local state transitions. For every action $a^i \in A^i$ of agent i, the state s^i is connected to the root node v_{a^i} of an action tree by an arc labeled with action a^i. For every root node v_{a^i}, let $v = v_{a^i}$ be the root of an action tree such that it is defined recursively over the remaining $N' = e \setminus \{i\}$ agents as follows:*

A1 If $N' \neq emptyset$ take some $j \in N'$, otherwise stop.

A2 For every $a^j \in \mathcal{A}(\tau^i, j)$, create an internal node v_{a^j} connected from v by an arc labeled with the action a^j.

A3 Create one internal node v_{*^j} to represent all actions of agent j not in $\mathcal{A}(\tau^i, j)$ (if any), connected by an arc labeled by the 'other action' wildcard $*^j$ from the root node v.

A4 For each leaf node v_{a^j} (or v_{*^j}) that results from the previous steps, create a sub-tree with $N' = N' \setminus \{j\}$ and $v = v_{a^j}$ as its root using the same procedure.

When all action arcs have been created, each leaf node v_{a^x} of the action tree is the root node u of an influence tree, where a^x is either the last dependent action or wildcard $*^x$ of the agent x that is visited in the last recursion. Starting again from $N' = e \setminus \{i\}$:

B1 If $N' \neq emptyset$ take some $j \in N'$, otherwise stop.

B2 If the path from s^i to node u contains an arc labelled with action $a^j \in \mathcal{A}(\tau^i, j)$, create child nodes $u_{s^j \to \hat{s}^j}$ to represent all local pairs of state transitions $(s^j, \hat{s}^j)$ of agent j in the action influence $I^i(\tau^i, a^j)$, connected to node u by arcs labeled $s^j \to \hat{s}^j$.

else

The path from s^i to node u contains the wildcard $*^j$. Create child nodes $u_{s^j \to \hat{s}^j}$ for all pairs of local state transitions $(s^j, \hat{s}^j) \in I^i(\tau^i, *^j)$, i.e. the influence of the set of actions represented by $*^j$ (all $a^j \notin \mathcal{A}(\tau^i, j)$), and connect them to u with arcs labelled $s^j \to \hat{s}^j$.

B3 If there remains any pair of local states $(s^j, \hat{s}^j) \in S^j \times S^j$ with $T(s^j, a^j, \hat{s}^j) > 0$ that is not in $I^i(\tau^i, a^j)$ or a pair with $\sum_{a^j \in *^j} T(s^j, a^j, \hat{s}^j) > 0$ that is not in $I^i(\tau^i, *^j)$, depending on the action of agent j on the path to node u, create another child node $u_{\diamond^j}$ connected by an arc labeled by the 'other state pairs' wildcard $\diamond^j$.

B4 For each leaf node $u_{s^j \to \hat{s}^j}$ (or $u_{\diamond^j}$) that results from the previous step, create a sub-tree with $N' = N' \setminus \{j\}$ and root $u = u_{s^j \to \hat{s}^j}$ (resp. $u = u_{\diamond^j}$) repeating the same procedure.

Finally, each leaf node $u_{s^x \to \hat{s}^x}$ (x again being the last agent) of every influence tree is connected to the new local state node $\hat{s}^i$ by an arc labeled with the transition reward $\mathcal{R}_i(\tau^e)$ that corresponds to the actions and state pairs on the path from s^i to $\hat{s}^i$.

The labels on the path to a leaf node of an influence tree, via a leaf node of the action tree, sufficiently specify the joint transitions of the agents in scope of the functions $R^e \in \mathcal{R}^i$, such that we can compute the reward $\sum_{R^e \in \mathcal{R}_i} R^e(s^e, \vec{a}^{\,e}, \hat{s}^e)$. Note that for each R^e for which an action is chosen that is not in $\mathcal{A}(a^i, j)$ (a wildcard $*^j$ in the action tree), the interaction reward must be 0 by definition (and similarly for state transitions in $\diamond^j$).

In Figure 1b an example CRG is illustrated. The local state nodes are displayed as circles; the internal nodes as black dots and action tree leaves as black triangles. The action arcs are labelled a^1, a^2 and 'wildcard' $*^2$, whereas transition influence arcs are labelled $(s_4^2 \rightarrow s_4^2)$, $(s_4^2 \rightarrow s_6^2)$ and $\diamond^j$. Note that Definition 4 captures the general case, but often it suffices to consider 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 set of state features on which the reward functions $\mathcal{R}_i$ depend. This is a further abstraction: only feature influence arcs are needed, typically requiring much less arcs (demonstrated later in Figure 2).

Now we investigate the maximal size of the CRGs to derive a theoretical upper bound. Let $|S^{max}| = \max_{i \in N} |S^i|$ and $|A^{max}| = \max_{i \in N} |A^i|$ denote respectively the maximal individual state and action space sizes, $w = \max_{R^e \in \mathcal{R}^e} |e| - 1$ denote the maximal interaction function scope size, $\alpha = \max_{i,j \in N} \max_{a^i \in A^i} |\mathcal{A}(a^i, j)|$ the set size of the largest dependent action set, and let finally $\beta = \max_{i,j \in N} \max_{\tau^i \in \mathcal{T}^i} \max_{a^j \in A^j} |I(\tau^i, a^j)|$ denote the size of the largest transition influence set. First note that the full joint policy search space is $\Theta(h|S^{max}|^{2n}|A^{max}|^n)$, however we show that the use of CRGs can greatly reduce this:

Theorem 1. *The maximal size of a CRG is*

$$O(\, h \cdot |A^{max}||S^{max}|^2 \cdot (\alpha\beta)^w \,). \tag{9}$$

Proof. A CRG has as many layers as the planning horizon h. In the worst case, in every stage there are $|S^{max}|$ local state nodes, each connected to at most $|S^{max}|$ next-stage local state nodes via multiple arcs. The number of action arcs between two local state nodes s^i and $\hat{s}^i$ is at most $|A^i|$ times the maximal number of dependent actions, α^w. Finally, the number of influence arcs is bounded by β^w. □

Note that in general all actions and transitions can cause interaction rewards, in which case the size of all n CRGs combined is $O(nh|S^{max}|^{2+2w}|A^{max}|^{1+w})$; typically still much more compact than the full joint policy search space unless $w \approx |N|$. For many problems however, the interaction rewards are more sparse and $\alpha^w \ll |A^{max}|^w$. Moreover, (9) gives an upper bound on the CRG size in general, for a specific CRG ϕ_i this bound is often expressed more tightly by $O(h \cdot |A^i||S^i|^2 \cdot \prod_{j \in N} (\alpha^i \beta^i)^w)$, where α^i and β^i denote respectively the maximal dependent action and transition influence set sizes for agent i. Finally, each $|S^i|$ can be reduced to $|F^i|$ when conditioning on state features is sufficient.

Bounding the optimal value In addition to storing rewards compactly, we use CRGs to bound the optimal policy value. Specifically, the maximal (resp. minimal) return from a joint state s_t onwards, is an upper (resp. lower) bound on the attainable reward, later to combined with its probability to obtain the expected value. Moreover, the sum of bounds on local returns bounds the global return and thus on the globally optimal joint policy value. We define the bounds recursively:

$$U(s^i) = \max_{(s^e, \vec{a}^{\,e}_t, \hat{s}^e) \in \phi_i(s^i)} \left(\mathcal{R}_i(s^e, \vec{a}^{\,e}_t, \hat{s}^e) + U(\hat{s}^i) \right), \tag{10}$$

such that $\phi_i(s^i)$ denotes the set of local transitions available from state $s^i \in s^e$ (ending in $\hat{s}^i \in \hat{s}^e$) represented in CRG ϕ_i. The bound on the optimal value for a joint transition $(s, \vec{a}, \hat{s})$ of all agents is

$$U(s, \vec{a}_t, \hat{s}) = \sum_{i \in N} \left(\mathcal{R}_i(s^e, \vec{a}^{\,e}_t, \hat{s}^e) + U(\hat{s}^i) \right), \tag{11}$$

and lower bound L is defined similarly over minimal returns.

Conditional Reward Independence Furthermore, CRGs can exploit independence in local reward functions as a result of past decisions. In many task-modelling MMDPs, e.g. those mentioned in the introduction, actions can be performed a limited amount of times, after which reward interactions involving that action no longer occur. When an agent can no longer perform dependent actions, the expected value of the remaining decisions is found through local optimisation. More generally, when dependencies between 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.

Definition 5 (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 future states $s_t, s_{t+1} \in S$ and every future joint action $\vec{a}_t \in A$:*

$$\forall R^e \in \mathcal{R}\ s.t.\ \{i,j\} \subseteq e : \sum_{x=t}^{h-1} R^e(s_x, \vec{a}_x, s_{x+1}) = 0. \tag{12}$$

Although reward independence is concluded from joint execution sequence θ_t, some independence can be detected from the local execution sequence θ_t^i, for example when agent i completes its dependent actions. This *local conditional reward independence* occurs when $\forall j \in N : CRI(i, j, \theta_t^i)$ and is easily detected from the state during CRG generation. For each such state s^i, we find optimal policy $\pi_i^*(s^i)$ and add only the optimal transitions dictated by that policy to our CRG, further reducing the CRG size.

Conditional Return Policy Search All the previous combined leads to the *Conditional Return Policy Search (CoRe)* (Algorithm 1). CoRe performs a branch-and-bound search over the joint policy space, represented as a DAG with nodes s_t and edges $\langle \vec{a}_t, \hat{s}_{t+1} \rangle$, such that finding a joint policy corresponds to selecting a subset of action arcs from the CRGs (corresponding to $\vec{a}_t$ and $\hat{s}_{t+1}$). First, however, the CRGs ϕ_i are constructed for the local rewards $\mathcal{R}_i$ of each agent $i \in N$, assigned heuristically to obtain the CRGs. Preliminary experiments provided evidence that a balanced distribution of rewards over the CRGs leads to the best results in the MPP domain. Further research is required to find effective heuristics for other domains. The generation of the CRGs follows Definition 4 using a recursive procedure, during which we store bounds (Equation 10) and flag local states that are locally conditionally reward independent according to Definition 5. During the subsequent policy search CoRe detects when subsets of agents, $N' \subset N$, become conditionally reward independent, and recurses on these subsets separately.

After construction of the CRGs, CoRe performs depth-first policy search (Algorithm 1) over the (disjoint sub-)set of agents N with potential reward interactions (line 2). These subsets are found with a connected component algorithm on a graph with nodes N and an edge (i, j) for every pair of agents $i, j \in N'$ that are still dependent given the current execution sequence θ_t^N, or $\neg CRI(i, j, \theta_t^N)$. In lines 2 to 10, we thus only consider local state space $S^{N'} \subseteq S$ and joint actions $\vec{a} \in A^{N'}$. Lines 3 and 4 determine the upper and lower bounds for this subset of agents, retrieved from the CRGs, used to prune in line 5. 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} (line 8), of which the highest is returned (line 10). As an extra, the lower bound is tightened (if possible) after every evaluation (line 9).

Theorem 2 (CoRe Correctness)**.** *Given TI-MMDP $M = \langle N, S, A, T, \mathcal{R} \rangle$ with (implicit) initial state s_0, CoRe always returns the optimal MMDP policy value $V^*(s_0)$ (Eq. 2).*

Proof. (Proof Sketch) Conditional reward independence enables optimal decoupling of policy search, the bounds are admissible with respect to the optimal policy value and our pruning does not exclude optimal execution sequences. The full proof can be found in the Appendix. □

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

Input: CRGs Φ, current execution sequence θ_t^N, planning horizon h, agent (sub) set N

if $t = h$ **then return** 0 $V^* \leftarrow 0$

foreach *conditionally independent subset* $N' \subseteq N$ *given* θ_t^N **do**

  // Compute weighted sums of bounds:

  $\forall \vec{a}_t^{N'} : U(s_{\theta,t}^{N'}, \vec{a}_t^{N'}) \leftarrow \sum_{s_{t+1}^{N'}} T(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'}) U(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'})$

  $L_{max} \leftarrow \max_{\vec{a}_t^{N'}} \sum_{s_{t+1}^{N'}} T(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'}) L(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'})$

  // Find joint action maximising expected reward

  foreach $\vec{a}_t^{N'}$ *for which* $U(s_{\theta,t}^{N'}, \vec{a}_t^{N'}) \geq L_{max}$ **do**

    $V_{\vec{a}_t^{N'}} \leftarrow 0$

    foreach $s_{t+1}^{N'}$ *reachable from* $s_{\theta,t}^{N'}$ *and* $\vec{a}_t^{N'}$ **do**

      $V_{\vec{a}_t^{N'}} += T(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'}) \Big(R(s_{\theta,t}^{N'}, \vec{a}_t^{N'}, s_{t+1}^{N'}) + \texttt{CoRe}(\Phi, \theta_t^{N'} \oplus [\vec{a}_t^{N'}, s_{t+1}^{N'}], h, N') \Big)$

    $L_{max} \leftarrow \max(V_{\vec{a}_t^{N'}}, L_{max})$ // update lb

  $V^* += \max_{\vec{a}_t^{N'}} V_{\vec{a}_t^{N'}}$

return V^*

5 CoRe Example

We present a two-agent example problem in which both agents have actions a, b and c, but every action can be performed only once within a 2 step horizon. Action c^2 of agent 2 is (for ease of exposition) the only stochastic action with outcomes c and c', and corresponding probabilities 0.75 and 0.25. 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 $f^1?$ to f^1 or $\neg f^1$. Thus we have one interaction reward function with rewards $R^{1,2}(f^1?, \{a^1, a^2\}, f^1)$ and $R^{1,2}(f^1?, \{a^1, a^2\}, \neg f^1)$, and local rewards R^1 and R^2. Without specifying actual rewards, we demonstrate the CRGs and CoRe.

Figure 2 illustrates the two CRGs. On the left is the CRG ϕ^1 of agent 1 with only its local reward R^1, while the CRG of agent 2 includes both the reward interaction function $R^{1,2}$ and its local reward R^2. Notice that only when sequences start with action a^2 additional arcs are included in CRG ϕ^2 to account for reward interactions. The sequence starting with a^2 is followed by an after-state node with two arcs: one for agent 1 performing a^1 and one for its other actions, $*^1 = \{b^1, c^1\}$. The interaction reward depends on what feature f^1 is (stochastically) set to, thus the influence arcs f^1 and $\neg f^1$ are now required. As the interaction reward only occurs when $\{a^1, a^2\}$ is executed, the fully-specified after-state node after a^2 and $*^1$ (the triangle below it) has a no-influence arc $\diamond^1$. All other transitions are reward independent and captured by local transitions (s_0^1, b^1, s_b^1) and (s_0^1, c^1, s_c^1). Locally reward independent states are highlighted green and from each of these states only the optimal action transition is kept in the CRG, e.g. only action arc c^1 is included from s_a^1. This action was determined optimal from the local state by single-agent optimal policy search, and thus no arcs for other actions (here b^1) are necessary.

Consider for example the state s_a^1 of ϕ^1, in which action a^1 has been taken in the first time step. From this state onward, the reward interaction between action a^1 and a^2 will no longer take place and therefore agent 1 is locally independent. Consequentially, we can remove either the branch for action b^1 or c^1 based on which action maximises the expected value. In addition, this action will cause both agents to become independent from each other because of the single dependency function between them.

Policy Search An example of CoRe policy search is shown in Figure 3, with the policy search space on the left and the CRGs on the right, now annotated with return bounds. Only several of the branches of the full DAG and CRGs are shown to preserve clarity. At $t = 0$, there are 9 joint actions with 12 result states, while the CRGs need only 3 + 4 states and 3 + 6 transitions to represent all re-

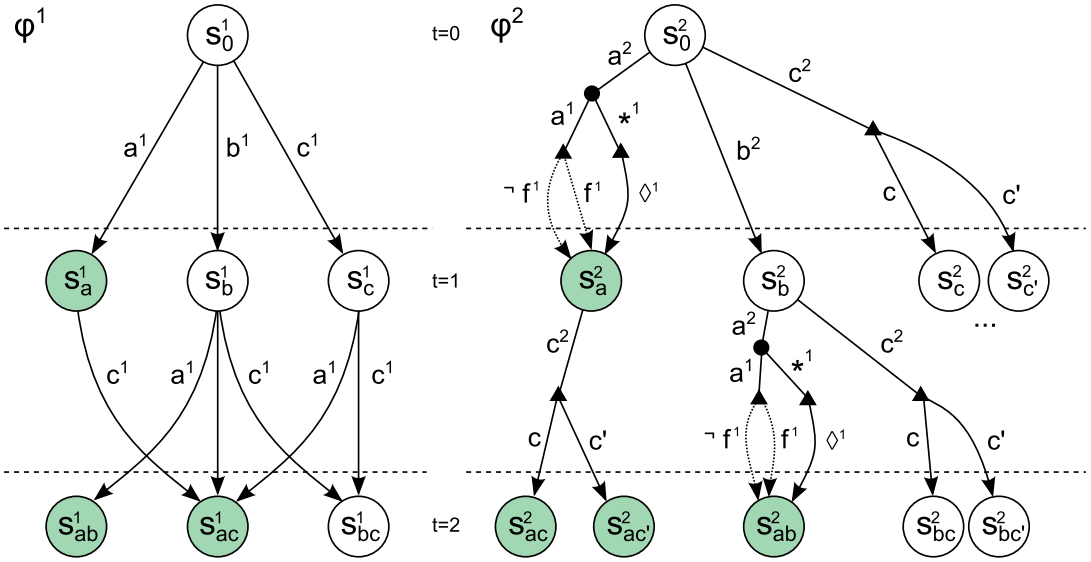


Figure 2: The CRGs of the two agents. We omit the branches for a^2 and b^2 from states s^2_c and $s^2_{c'}$. The highlighted states are locally reward independent (reward arcs are omitted).

wards. The execution sequence θ_h that is evaluated 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 the corresponding transitions to states s^1_b and s^2_b are shown. Now for $t = 1$ CoRe is evaluating joint action $\{a^1, a^2\}$ that is reward-interacting and thus the value of state feature f^1 is required to determine the transition in ϕ^2 (here chosen arbitrarily as $\neg f^1_x$). The corresponding execution sequence (of agent 2) is therefore $\theta^2_2 = [s^2_0, \{b^1, b^2\}, s^2_b \cup \{f^1?\}, \{a^1, a^2\}, s^2_{ba} \cup \{\neg f^1\}]$ If agent 1 had chosen action c^1 instead, we would traverse the branches $*^1$ and $\diamond^1$ leading to state s^2_{ba} without reward interactions.

Branch-and-bound is shown (in blue) for state $s_{b,b}$, with the rewards labelled on transitions and their bounds at the nodes. The bounds for joint actions $\{a^1, a^2\}$ and $\{a^1, c^2\}$ are $[13, 16]$ and $[12.5, 12.5]$, respectively, found by summing the CRG bounds, hence $\{a^1, c^2\}$ can be pruned. Note that we can compute the expected value of $\{a^1, c^2\}$ 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. Regardless, we can bound the return of action a^2 over all possible feature values, stored in ϕ^2, and they can be updated as the probabilities become known during policy search.

Conditional reward independence occurs in the green states of the policy search tree. After joint action $\{b^1, a^2\}$, the agents will no longer interact (a^2 is completed and will not be available anymore) and thus the problem is decoupled. From state $s_{b,a}$ CoRe finds optimal policies $\pi^*_1(s^1_b)$ and $\pi^*_2(s^2_a)$ and combines them into the optimal joint policy $\pi^*(s_{b,a}) = \langle \pi^*_1(s^1_b), \pi^*_2(s^2_a) \rangle$ for the planning problem remaining from independent state $s_{b,a}$.

6 Evaluation

In our experiments we find optimal policies for the *maintenance planning problem* (MPP, see the introduction) that minimise the (time-dependent) maintenance costs and economic losses due to traffic hindrance. In this problem agents represent contractors that participate in a mechanism and thus it is essential that the planning is done optimally.

The problem can be modelled as an MDP, as is explained in full detail in [Scharpff et al., 2013].

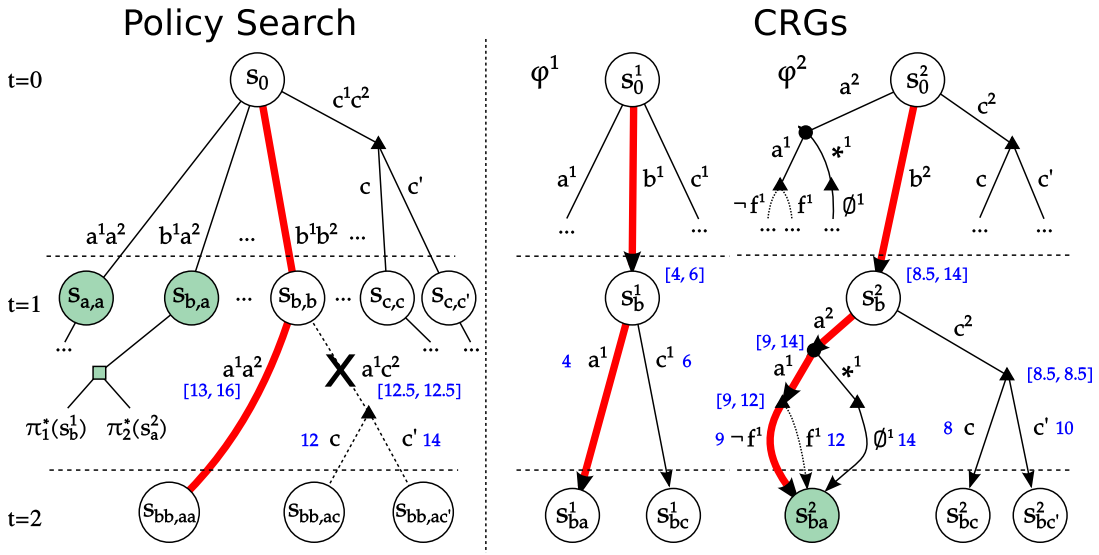
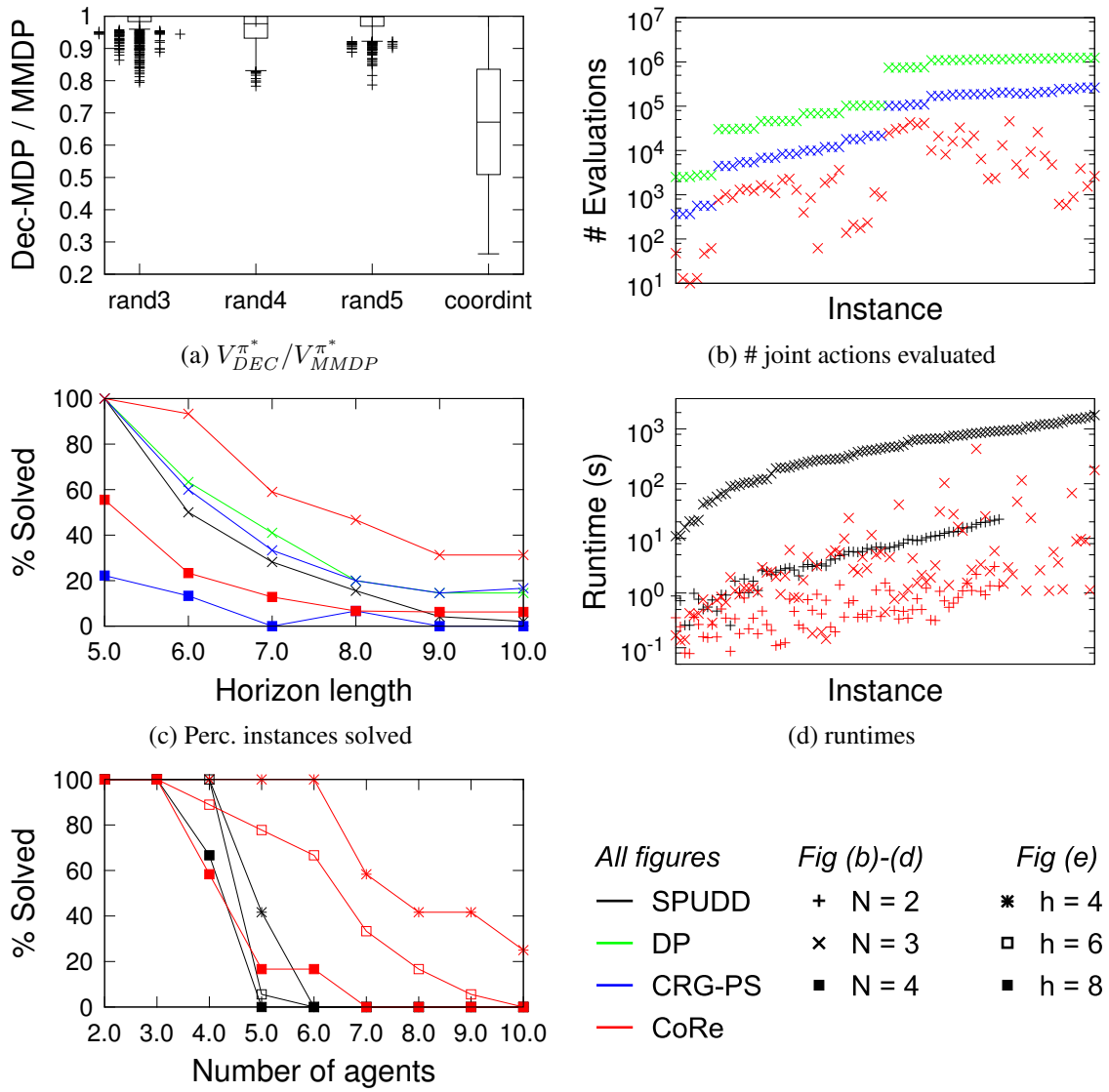


Figure 3: Example of policy evaluation. The left graph shows (a part of) the policy search tree with joint states and joint actions, and the right graph the CRGs per agent. One possible execution sequence θ_h is highlighted in thick red.

Here we only briefly outline the problem. Agents maintain a state with start and end times of their maintenance tasks. Each of these tasks can be performed only once and agents can perform exactly one task at a time (or do nothing). The individual rewards are given by maintenance costs that are task and time dependent, while interaction rewards model network hindrance due to concurrent maintenance. Maintenance costs are task and time dependent, while interaction rewards model network hindrance due to concurrent maintenance. In 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.

First, we compare with a decentralised baseline by treating the problem as a (transition and observation independent) Dec-MDP [Becker et al., 2003] in which agents can only observe their local state. Although the (TI-)Dec-MDP model is fundamentally different from the TI-MMDP – in the latter decisions are coordinated on *joint* (i.e., global) observations – the advances in Dec-MDP solution methods [Dibangoye et al., 2013] may be useful for TI-MMDP problems if they can deliver sufficient quality policies. That is, since they assume less information available, the value of Dec-MDP policies will *at best* equal that of their MMDP counterparts, but in practice the expected value obtained from following a decentralised policy may be lower. We investigate if this is the case in our first experiment, which compares the expected value of optimal MMDP policies found by CoRe with optimal Dec-MDP policies, as found by the GMAA-ICE* algorithm [Oliehoek et al., 2013a].

For this experiment we use two benchmark sets: `rand[h]`, 3 sets of 1000 random two-agent problems with horizons $h \in [3, 4, 5]$, and `coordint`, a set of 1000 coordination-intensive instances. The latter set contains tightly-coupled agents with dependencies constructed in such a way that maintenance delays inevitably lead to hindrance unless agents coordinate their decisions when such a delay becomes known, which is ar the first time step after starting the maintenance task. Figure 4a shows the ratio $V^{\pi^*}_{DEC}/V^{\pi^*}_{MMDP}$ of the expected value of the optimal Dec-MDP policy $V^{\pi^*}_{DEC}$ versus that of the optimal MMDP policy $V^{\pi^*}_{MMDP}$. In the random instances the expected values of both policies equal in approximately half of the instances. For coordination-intensive instances `coordint` decentralised policies result in worse results – on average the reward loss is about 33%, but it can be 75% – demonstrating that decentralised policies are inadequate for our purposes. As this experiment only served to establish that decentralised methods are indeed not applicable, from now on only centralised methods are considered.



(a) $V^{\pi^*}_{DEC}/V^{\pi^*}_{MMDP}$

(b) # joint actions evaluated

(c) Perc. instances solved

(d) runtimes

(e) Perc. instances solved

Figure 4: Experimental results: (a) The ratio of expected reward of the optimal Dec-MDP policy versus the expected reward of the optimal MMDP policy for 1000 2-agent, 2-activity instances of four sets of problems (3 sets of random problems `rand[h]` with horizon 3, 4 and 5 and a set of coordination intensive instances `coordint`), (b) Number of joint actions evaluated by the dynamic programming algorithm (DP), CRG-enabled policy search (CRG-PS) and CoRe for the `mpp` instances (log scale), (c) percentage of instances from `mpp` that have been solved by each of the algorithms within 30 minutes, (d) runtime comparison of SPUDD and CoRe for all `mpp` instances that were successfully solved by SPUDD (also log scale) and (e) the percentage of `pyra` instances solved within the 30 minute time limit.

In our remaining experiments we used a random test set `mpp` with 2, 3 and 4-agent problems (400 each) with 3 maintenance tasks, horizons 5 to 10, random delay probabilities and binary reward interactions. We compare CoRe against several other algorithms to investigate the performance of the algorithm. The current state-of-the-art approach to solve MPP, presented in [Scharpff et al., 2013], uses the value iteration algorithm SPUDD [Hoey et al., 1999] and solves an efficient MDP encoding of the problem. SPUDD uses algebraic decision diagrams to compactly represent all rewards and is in this sense somewhat similar to our work, however it does not implicitly partition and decouple rewards. Besides the SPUDD solver we included a dynamic programming algorithm (DP) that uses a depth-first approach to maximise the Bellman equation of Equation 2. In addition to basic value iteration we implemented some domain knowledge in this algorithm to quickly identify and prune sub-optimal and infeasible branches during evaluation. Finally, to analyse the impact that branch-and-bound can have in a task-based planning domain such as MPP we added also a CRG-enabled policy search algorithm (CRG-PS), a variant of our CoRe algorithm that uses CRGs but not branch-and-bound pruning.

Using these algorithms, we first study the impact of using CRGs on the number of joint actions that need to be evaluated. SPUDD is not considered in this experiment because because it does not report this information. Figure 4b shows the search space size reduction by CRGs in this domain. Our CRG-enabled algorithm (CRG-PS, blue) approximately decimates the number of evaluated joint actions compared to the DP method (green). Furthermore, when value bounds are used (CoRe, red), this number is reduced even more, although its effect varies per instance.

Having observed the policy search space reduction that CoRe can achieved, we are interested in the scalability of the algorithm in terms of number of agents and planning horizon. Figure 4c shows the percentage of problems from the `mpp` test set that are solved within 30 minutes per method (all two-agent instances were solved and hence omitted). CoRe solves more instances than SPUDD (black) of the 3 agent problems (cross marks), and only CRG-PS and CoRe solve 4-agent instances. This is because CRGs successfully exploit the conditional action independence that decouples the agents *for most of the planning decisions*. Only when reward interactions may occur actions are coordinated, whereas SPUDD always coordinates every joint decision. Notice also that the use of branch-and-bound enables CoRe to solve more instances, compared to the CRG-enabled policy search.

Next we investigate the runtime that was required by CoRe versus that by the current best known method based on SPUDD. As CoRe achieves a greater coverage than SPUDD, we compare runtimes only for instances successfully solved by the latter (Figure 4d). We order the instances on their SPUDD runtime (causing the apparent dispersion in CoRe runtimes) and plot runtimes of both. Note that as a result the horizontal axis is not informative, it is the vertical axis plotting the runtime that we are interested in. CoRe solves almost all instances faster than SPUDD, both with 2 and 3 agents, and has a greater solving coverage: CoRe failed to solve 3.4% of the instances solved by SPUDD whereas SPUDD failed 63.9% of the instances that CoRe solved.

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 solved instances from the `pyra` test for various problem horizons. Whereas previous state-of-the-art solved 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.

7 Conclusions and Future Work

In this work, we focus on optimally (and centrally) solving fully-observable, stochastic planning problems where agents are dependent only through interaction rewards. We partition individual and interaction rewards per agent in *conditional return graphs*, a compact and efficient data structure when interactions are sparse and/or non-recurrent. We propose a conditional return policy search algorithm

(CoRe) that uses reward bounds based on CRGs to reduce the search space size, shown to be by orders of magnitude in the maintenance planning domain. This enables CoRe to overall decrease the runtime required compared to the previously best approach and solve instances previously deemed unsolvable. The reduction in search space follows from three key insights: 1) when interactions are sparse, the number of unique returns per agent is relatively small and can be stored efficiently, 2) we can use CRGs to maintain bounds on the return, and thus the expected value, and use this 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 the impact of reduction is by orders of magnitude in the maintenance planning domain. This enables CoRe to solve instances that were previously deemed unsolvable. In addition, to scaling to larger instances, CoRe almost always produces solutions faster than the previously best approach. Moreover, CoRe is able to scale up to 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 also illustrate that using a decentralised MDP approach, a line of research that has seen many scalable approaches in terms of agents and reward structures, leads to suboptimal expected policy values.

Here only optimal solutions are considered, but CRGs can be combined with approximation in several ways. First, the reward structure of the problem itself may be approximated. For instance, the reward-function approximation of [Koller and Parr, 1999] can be applied to increase reward sparsity, or CRG paths with relatively small reward differences may be grouped, trading off a (bounded) reward loss for compactness. Secondly, the CRG bounds directly lead to a bounded-approximation variant of CoRe, usable in for instance the approximate multi-objective method of [Roijers et al., 2014]. Lastly, the CRG structure can be implemented in any (approximate) TI-MMDP algorithm or, vice versa, any existing approximation scheme for MMDP that preserves TI can be used within CoRe.

Although we focused on transition-independent MMDPs, CRGs may be interesting for general MMDPs when transition dependencies are sparse. This would require including dependent-state transitions in the CRGs similar to reward-interaction paths and is considered to be future work. Another interesting avenue is to exploit conditional reward independence during joint action generation.

Acknowledgements

This research is supported by the NWO DTC-NCAP (#612.001.109), Next Generation Infrastructures, Almende BV and NWO VENI (#639.021.336) projects.

Appendix: Proof of Theorem 2

In this appendix, we prove the correctness of the CoRe algorithm (Theorem 2).

We define several notational shorthands for convenience. For two (sub)sets of agents $A, B \subseteq N$, $\mathcal{R}^e_{AB} \subseteq \mathcal{R}$ is the set of all rewards for which $A \cap B \cap e \neq \emptyset$. $\mathcal{R}^e_{A\not B} \subseteq \mathcal{R}$ is the set of rewards such that $A \cap e \neq \emptyset$ and $B \cap e = \emptyset$. Observe that the individual rewards for all agents $a \in A$ are thus contained within $\mathcal{R}^e_{A\not B}$ (and similarly all rewards R^b are included in $\mathcal{R}^e_{\not AB}$). Furthermore, two agent sets A and B are conditionally reward independent, denoted $CRI(A, B, \theta^t)$, as result of history θ_t iff $\forall a \in A, b \in B\colon CRI(a, b, \theta_t)$. Finally, $\tau^e = (\{s^j\}_{j\in e}, \{\vec{a}^{\,j}\}_{j\in e}, \{\hat{s}^j\}_{j\in e})$ denotes a transition local to agents e and a global transition, i.e. $e = N$, is denoted by τ.

Lemma A.1. *Given an execution history $\theta_t = [s_0, \vec{a}_0, \ldots, s_u, \ldots, s_t]$ up to time t that can be partitioned into two histories, $\theta_u = [s_0, \ldots, s_u]$ and $\theta_{u'} = [s_u, \ldots, s_t]$, and a disjoint partitioning of agents $N = N_1 \cup N_2 \cup \ldots \cup N_k$ such that for every pair $N_a, N_b \in N$ it holds that $CRI(N_a, N_b, \theta_u)$ when $a \neq b$.*

The return $Z(\theta_t)$ (as defined in Equation 3) can be decoupled as:

$$Z(\theta_u) + \sum_{i=1}^{k} Z_{N_i}(\theta_{u'}^{N_i}) \tag{A.13}$$

where $\theta_{u'}^{N_i}$ is the execution history only containing the states and actions of the agents in the set $N_i \subseteq N$, starting from time u.

Proof. Recall from Definition 5 that two agents $i, j \in N$ are CRI iff $\forall R^e \in \mathcal{R}\ s.t.\ \{i,j\} \subseteq e : \sum_{x=t}^{h} R^e(s_x, \vec{a}_x, s_{x+1}) = 0$ for every pair of states s_t, s_{t+1} and all joint actions $\vec{a}_t$, given execution history θ_t. Moreover, recall that the MMDP rewards $\mathcal{R}$ w.l.o.g. are structured as $R(\tau) = \sum_{R^e \in \mathcal{R}} R^e(\tau^e)$.

Let A and B be disjoint subsets of agents such that $A \cup B = N$ and let the reward functions be accordingly partitioned as disjoint sets: $\mathcal{R} = \mathcal{R}^e_{AB\!\!\not}\cup\mathcal{R}^e_{\not AB}\cup R^e_{AB}$. Now assume that for a given execution history $\theta_t = \theta_u \cup \theta_{u'}$ we have $CRI(A, B, \theta_u)$. From the state s_u resulting from the execution history θ_u, all future rewards can only be local with respect to subsets A and B because every reward R^e_{AB} must be zero by definition of CRI. Therefore we can rewrite the (future) global reward $R(\tau)$ of every possible transition τ (in every possible $\theta_{u'}^{N_i}$) as:

$$\sum_{R^e \in \mathcal{R}^e} R^e(\tau) = \mathcal{R}^e_{AB\!\!\not}(\tau^A) + \mathcal{R}^e_{\not AB}(\tau^B) + \mathcal{R}^e_{AB}(\tau^{AB}) \tag{A.14}$$

$$= \sum_{R^e \in \mathcal{R}^e_{AB\!\!\not}} R^e(\tau^A) + \sum_{R^e \in \mathcal{R}^e_{\not AB}} R^e(\tau^B) \tag{A.15}$$

where the transition decoupling is possible due to transitional independence. Remember that we can write the returns for an execution history θ_h as (Eq. 3)

$$Z(\theta_t) = \sum_{x=0}^{t-1} \sum_{R^e \in \mathcal{R}} R^e(\tau^e_{\theta,x}) \tag{A.16}$$

in which τ^e_θ denotes the transition in the execution history θ_t local to agents e. Then, for two disjoint agent subsets $A \cup B = N$ that have $CRI(A, B, \theta_u)$ as a result of θ_u:

$$Z(\theta_t) = Z(\theta_u) + Z(\theta_{u'}) \tag{A.17}$$

$$= Z(\theta_u) + \sum_{x=u}^{t-1} \left[\mathcal{R}^e_{AB\!\!\not}(\tau^A_{\theta,x}) + \mathcal{R}^e_{\not AB}(\tau^B_{\theta,x})\right] \tag{A.18}$$

$$= Z(\theta_u) + \sum_{x=u}^{t-1} \mathcal{R}^e_{AB\!\!\not}(\tau^A_{\theta,x}) + \sum_{x=u}^{t-1} \mathcal{R}^e_{\not AB}(\tau^B_{\theta,x}) \tag{A.19}$$

$$= Z(\theta_u) + Z_A(\theta^A_{u'}) + Z_B(\theta^B_{u'}) \tag{A.20}$$

As a consequence, we can decouple the computation of returns for agent sets A and B from time u.

For now we have only considered two agent sets A and B, however we can apply this decoupling recursively, in order to obtain an arbitrary disjoint partitioning of agents such that $N_1 \cup N_2 \cup \ldots \cup N_k = N$ and $\forall N_a, N_b \in N :\ CRI(N_a, N_b, \theta_u)$. That is, without loss of generality, we choose $A = N_a$ and $B = N_2 \cup \ldots \cup N_k$ and decouple the return as $Z(\theta_u) + Z_A(\theta^A_{u'}) + Z_B(\theta^B_{u'})$. We now observe that we can rewrite Z_B itself as $Z_{N_2}(\theta^{N_2}_{u'}) + Z_{B\setminus N_2}(\theta^{B\setminus N_2}_{u'})$ by following the same steps, because both sets again satisfy conditional reward independence. By continuing this process we obtain Equation A.13. □

As a result of Lemma A.1, Equation 4 and transitional independence we can now also decouple the policy values of two sets of agents, N_a and N_a, from time u onwards, when it holds that $CRI(N_a, N_b, \theta_u)$.

Corollary A.1. *When at a timestep u, we have observed θ_u and there is a disjoint partitioning of agents $N = N_1 \cup N_2 \cup \ldots \cup N_k$ such that for every pair $N_a, N_b \in N$ it holds that $CRI(N_a, N_b, \theta_u)$ when $a \neq b$, the value of a given policy π, $V(s_t)$ can be decoupled as:*

$$V^{\pi}(s_t) = \sum_{i=1}^{k} V^{\pi}_{N_i}(s_t^{N_i}) \tag{A.21}$$

Proof. Starting from Equation 4, we first observe that the return $Z(\theta_{u'})$ from timestep u onwards is equal to $\sum_{i=1}^{k} Z_{N_i}(\theta_{u'}^{N_i})$ (Lemma A.1) and that, because of transition independence, each set N_i of agents has independent probability distributions over future execution histories $\theta_{u'}$. Thus we have the following equalities:

$$V(s_t) = \sum_{\theta_{u'}|\pi,\theta_t} Pr(\theta_{u'}) Z(\theta_{u'}) \tag{A.22}$$

$$= \sum_{\theta_{u'}|\pi,\theta_t} Pr(\theta_{u'}) \sum_{i=1}^{k} Z_{N_i}(\theta_{u'}^{N_i}) = \sum_{\theta_{u'}|\pi,\theta_t} \sum_{i=1}^{k} Pr(\theta_{u'}) Z_{N_i}(\theta_{u'}^{N_i}) \tag{A.23}$$

$$= \sum_{i=1}^{k} \sum_{\theta_{u'}|\pi,\theta_t} Pr(\theta_{u'}) Z_{N_i}(\theta_{u'}^{N_i}) = \sum_{i=1}^{k} \sum_{\theta_{u'}^{N_i}|\pi,\theta_t} Pr(\theta_{u'}^{N_i}) Z_{N_i}(\theta_{u'}^{N_i}) \tag{A.24}$$

$$= \sum_{i=1}^{k} V^{\pi}_{N_i}(s_t^{N_i}) \tag{A.25}$$

□

Lemma A.2. *The bounding heuristics $L(s_t^e)$ and $U(s_t^e)$ used to prune during branch-and-bound search are admissible with respect to the expected value $V(s_t^e)$ of state s_t^e for agents $e \subseteq N$ at time t.*

Proof. We proof the admissibility of the bounding heuristics by induction. For sake of brevity, we only show the upper bound proof, but the proof for the lower bound can be written down accordingly. Recall that $R = \bigcup_{i \in N} \mathcal{R}_i$ is the disjoint partitioning of reward functions over the CRGs.

First, consider the very last timestep, $h-1$, for which there is no future reward, i.e.,

$$V(s_{h-1}) = \max_{\vec{a}} \sum_{s_h \in S} T(\tau_{h-1}) R(\tau_{h-1}) = \max_{\vec{a}} \sum_{s_h \in S} T(\tau_{h-1}) \sum_{i \in N} \mathcal{R}_i(\tau_{h-1}^e) \tag{A.26}$$

$$\leq \max_{\vec{a}} \max_{s_h \in S} \sum_{i \in N} \mathcal{R}_i(\tau_{h-1}^e) \tag{A.27}$$

$$\leq \sum_{i \in N} \max_{\tau_{h-1}^e \in \phi_i(s_{h-1}^i)} \mathcal{R}_i(\tau_{h-1}^e) \tag{A.28}$$

$$= \sum_{i \in N} U(s_{h-1}^i) \tag{A.29}$$

Then, we show that if for a next stage $t+1$ we have a valid upper bound, the value for a state s_t is also upper bounded by $\sum_{i \in N} U(s_t^i)$. And therefore, that because $U(s_{h-1}^i)$ is a valid upper bound on

$V(s_{h-1})$, the upper bound is admissible for all stages before $h-1$:

$$V(s_t) = \max_{\vec{a}} \sum_{s^e_{t+1} \in S} T(\tau_t) \left(R(\tau_t) + V(s_{t+1}) \right) \tag{A.30}$$

$$= \max_{\vec{a}^e} \sum_{s_{t+1} \in S} T(\tau_t) \left(V(s_{t+1}) + \sum_{i \in N} \mathcal{R}_i(\tau^e_t) \right) \tag{A.31}$$

$$\leq \max_{\vec{a}^e} \sum_{s_{t+1} \in S} T(\tau_t) \sum_{i \in N} \left(\mathcal{R}_i(\tau^e_t) + U(s^i_{t+1}) \right) \tag{A.32}$$

$$\leq \sum_{i \in N} \max_{\tau^e_t \in \phi_i(s^i_t)} \left(\mathcal{R}_i(\tau^e_t) + U(s^i_{t+1}) \right) \tag{A.33}$$

$$= \sum_{i \in N} U(s^i_t) \tag{A.34}$$

From the bounds on the state-values,

$$L(s_t) = \sum_{i \in N} L(s^i_t) \leq V(s_t) \leq \sum_{i \in N} U(s^i_t) = U(s_t),$$

we can also distill admissible bounds on state-action values, $Q(s_t, \vec{a})$. Taking the standard MDP definition for $Q(s_t, \vec{a})$,

$$Q(s_t, \vec{a}) = \sum_{\hat{s}_{t+1} \in S} T(s_t, \vec{a}, \hat{s}_{t+1}) \left(R(s_t, \vec{a}, \hat{s}_{t+1}) + V(s_{t+1}) \right),$$

we replace $V(s_{t+1})$ by the corresponding lower or upper bounds:

$$B(s_t, \vec{a}) = \sum_{\hat{s}_{t+1} \in S} T(s_t, \vec{a}, \hat{s}_{t+1}) \left(R(s_t, \vec{a}, \hat{s}_{t+1}) + B(s_{t+1}) \right).$$

Using $B(s_t, \vec{a})$, CoRe can exclude a joint action $\vec{a}$ after execution history θ_t from consideration when there is another joint action $\vec{a}'$ for which $U(s_t, \vec{a}) \leq L(s_t, \vec{a}')$, as is standard in branch-and-bound algorithms. □

Concluding the proof of Theorem 2. As a direct consequence of Corollary A.1 agents can be decoupled during policy search without losing optimality. Moreover, Lemma A.2 shows that both the upper and lower bounds are admissible heuristic functions with respect to the expected policy value from a given state. In the main loop, the CoRe algorithm recursively expands and evaluates all possible extensions to the current execution path, except for those starting with actions that lead to a lower upper bound than another action's lower bound.

As the policy search considers all possible execution histories, excluding pruned, non-optimal ones, the search will eventually return the optimal policy value V^*, and corresponding policy, thus proving Theorem 2. □

Moreover, as there is only a finite number of execution histories, the CoRe algorithm is also guaranteed to terminate in a finite number of recursions.

References

[Bakker et al., 2010] Bakker, B., Whiteson, S., Kester, L., and Groen, F. (2010). *Traffic Light Control by Multiagent Reinforcement Learning Systems*, chapter Interactive Collaborative Information Systems, pages 475–510. Studies in Computational Intelligence, Springer.

[Becker et al., 2004] Becker, R., Zilberstein, S., and Lesser, V. (2004). Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 302–309.

[Becker et al., 2003] Becker, R., Zilberstein, S., Lesser, V., and Goldman, C. V. (2003). Transition-independent decentralized Markov decision processes. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 41–48.

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Int. Conf. on Theoretical Aspects of Rationality and Knowledge*.

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

[De Hauwere et al., 2012] De Hauwere, Y.-M., Vrancx, P., and Nowé, A. (2012). Solving sparse delayed coordination problems in multi-agent reinforcement learning. In *Adaptive and Learning Agents*, pages 114–133. Springer.

[Dibangoye et al., 2013] Dibangoye, J. S., Amato, C., Doniec, A., and Charpillet, F. (2013). Producing efficient error-bounded solutions for transition independent decentralized MDPs. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*.

[Guestrin et al., 2002a] Guestrin, C., Koller, D., and Parr, R. (2002a). Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*. MIT Press.

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

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

[Kok and Vlassis, 2004] Kok, J. R. and Vlassis, N. (2004). Sparse cooperative Q-learning. In *Proceedings of the Int. Conf. on Machine Learning*, pages 481–488.

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

[Melo and Veloso, 2009] Melo, F. S. and Veloso, M. (2009). Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 2*, pages 773–780. International Foundation for Autonomous Agents and Multiagent Systems.

[Melo and Veloso, 2011] Melo, F. S. and Veloso, M. (2011). Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789.

[Messias et al., 2013] Messias, J. V., Spaan, M. T. J., and Lima, P. U. (2013). GSMDPs for multi-robot sequential decision-making. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 1408–1414.

[Meuleau et al., 1998] Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L. P., Dean, T. L., and Boutilier, C. (1998). Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 165–172.

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

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

[Oliehoek et al., 2013a] Oliehoek, F. A., Spaan, M. T. J., Amato, C., and Whiteson, S. (2013a). Incremental clustering and expansion for faster optimal planning in decentralized POMDPs. *Journal of Artificial Intelligence Research*, 46:449–509.

[Oliehoek et al., 2008] Oliehoek, F. A., Spaan, M. T. J., Whiteson, S., and Vlassis, N. (2008). Exploiting locality of interaction in factored Dec-POMDPs. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 517–524.

[Oliehoek et al., 2015] Oliehoek, F. A., Spaan, M. T. J., and Witwicki, S. J. (2015). Factored upper bounds for multiagent planning problems under uncertainty with non-factored value functions. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 1645–1651.

[Oliehoek et al., 2013b] Oliehoek, F. A., Whiteson, S., and Spaan, M. T. J. (2013b). Approximate solutions for factored Dec-POMDPs with many agents. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 563–570.

[Parr, 1998] Parr, R. (1998). Flexible decomposition algorithms for weakly coupled Markov decision problems. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 422–430. Morgan Kaufmann Publishers Inc.

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Roijers et al., 2014] Roijers, D. M., Scharpff, J., Spaan, M. T. J., Oliehoek, F. A., De Weerdt, M., and Whiteson, S. (2014). Bounded approximations for linear multi-objective planning under uncertainty. In *Proceedings of the Int. Conf. on Automated Planning and Scheduling*, pages 262–270.

[Scharpff et al., 2013] Scharpff, J., Spaan, M. T. J., de Weerdt, M., and Volker, L. (2013). Planning under uncertainty for coordinating infrastructural maintenance. In *Proceedings of the Int. Conf. on Automated Planning and Scheduling*, pages 425–433.

[Spaan et al., 2006] Spaan, M. T. J., Gordon, G. J., and Vlassis, N. (2006). Decentralized planning under uncertainty for teams of communicating agents. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*.

[Varakantham et al., 2007] Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., and Yokoo, M. (2007). Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems*.

[Witwicki and Durfee, 2010] Witwicki, S. J. and Durfee, E. H. (2010). Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *Proceedings of the Int. Conf. on Automated Planning and Scheduling*, pages 185–192.