gorithm. In particular, it terminates with a decentralized Markov policy within $\varepsilon = \bar{v}_0(\eta_0) - \underline{v}_0(\eta_0)$ of the optimal decentralized Markov policy. Indeed, the upper bound value functions $\bar{v}_\tau$ never underestimate the exact value at any state occupancy $\eta_\tau$. This is because we update the upper bound value at each state occupancy based upon a greedy decision rule for this state occupancy. On the other hand, the exhaustive variant algorithm requires the exhaustive enumeration of all possible decentralized Markov decision rules at each backup step (Algorithm 1, line 5). In MDP techniques, the exhaustive enumeration is not prohibitive since the action space is often manageable. In decentralized MDP planning, however, the space of all decentralized Markov decision rules increases exponentially with increasing observations and agents. As such, the exhaustive variant can scale only to problems with a moderate number of observations (local states) and two agents.

### 5.2 The constraint optimization formulation

To overcome the memory limitation of the exhaustive variant, we use constraint optimization instead of the exhaustive backup operation. More precisely, our constraint optimization program returns a greedy decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$ for each state occupancy $\eta_\tau$ visited, but without performing the exhaustive enumeration.

In our constraint optimization formulation, variables are associated with decision rules $\sigma^i(z^i)$ for all agents $i = 1, \ldots, n$ and all local observations $z^i \in Z^i$. The domain for each variable $\sigma^i(z^i)$ is action space $A^i$. For each state $s \in S$, we associated a single soft constraint $c_\tau(s, \cdot): A \mapsto \mathbb{R}$. Each of these assigns a value $c_\tau(s, a) = r(s, a) + \sum_{s', p(s, a, s') \cdot C_{\text{mdp}, \tau} + 1, \ldots, C_{\text{mdp}, \tau} - 1}(s')$ to each joint action $a \in A$. Value $c_\tau(s, a)$ denotes the reward accrued at horizon $\tau$ when taking joint action $a$ in state $s$ and then following the underlying MDP joint policy for the remaining planning horizons. For each decentralized Markov decision rule $\sigma_\tau \in \Psi_\tau(\eta_\tau)$, we also associate a single soft constraint $g_\tau(\cdot)$. Each of these assigns value $g_\tau(\sigma_\tau) = \hat{q}_\tau(\eta_\tau, \sigma_\tau) - \bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$, where $\bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau) = \sum_{\sigma_\tau}(s, \sigma_\tau)(s, \sigma_\tau)$. The objective of our constraint optimization program is to find an assignment $\sigma_{\text{greedy},\tau}$ of actions $a^i$ to variables $\sigma^i(z^i)$ such that the aggregate value is maximized. Stated formally, we wish to find $\sigma_{\text{greedy},\tau} = \arg \max_{\sigma_\tau} g_\tau(\sigma_\tau) + \sum_{\eta_\tau}(s, \sigma_\tau)(s, \sigma_\tau)$. To better understand our constraint optimization program, note that by the definition of mapping $\bar{q}_{\text{mdp}}$ we have that $\sum_{\eta_\tau}(s, \sigma_\tau)(s, \sigma_\tau) = \hat{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$. Hence, if we use $\bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$ instead of $\sum_{\eta_\tau}(s, \sigma_\tau)(s, \sigma_\tau)$, we get $\sigma_{\text{greedy},\tau} = \arg \max_{\eta_\tau} \hat{q}_\tau(\eta_\tau, \sigma_\tau)$. Thus, our constraint optimization program returns a decentralized Markov decision rule with the highest upper-bound value. Techniques that solve our constraint optimization formulation abound in the literature of constraint programming [9], allowing many different approaches to be utilized.

#### Theoretical guarantees

The constraint optimization variant yields the same guarantees as the exhaustive variant without the major drawback of exhaustive enumeration of all decentralized Markov decision rules. Instead, it uses a constraint optimization formulation that returns a greedy decentralized Markov decision rule, which will often be much more efficient than exhaustive enumeration. And hence, we retain the property that stopping the algorithm at any time, the solution is within $\varepsilon = \bar{v}_0(\eta_0) - \underline{v}_0(\eta_0)$ of an optimal decentralized Markov policy.

#### Comparison to COP based algorithms

There is a rich body of work that replaces the exhaustive backup operation by a constraint optimization formulation in decentralized control settings [15, 14, 19]. These constraint optimization programs compute a decentralized history-dependent policy for a given belief state. While MPS also takes advantage of a constraint optimization formulation, it remains fundamentally different. The difference lies in both the COP formulation and the heuristic search. In existing COP based algorithms for decentralized control, authors try to find the best assignment of sub-policies to histories. Instead, in our case the COP formulation aims at mapping local observations to local actions. This provides considerable memory and time savings. Moreover, existing algorithms proceed by backing up policies in a backward direction (i.e., from last step to first) using a set pre-selected belief states. In contrast, the MPS algorithm proceeds forward, expanding the state occupancy distributions and selecting greedily decision rules. Finally, the MPS algorithm returns an optimal solution, whereas other COP based algorithms for Dec-POMDPs return only locally optimal solutions [15, 14]. Approximate solutions are returned by the other algorithms because they plan over (centralized) belief states, which do not constitute a sufficient statistic for Dec-POMDPs (or Dec-MDPs).

#### 6 Empirical Evaluations

We evaluated our algorithm using several benchmarks from the decentralized MDP literature. For each benchmark, we compared our algorithms with state-of-the-art algorithms for solving Dec-MDPs and Dec-POMDPs. Note that we do not compare with ND-POMDP methods. Since our benchmarks allow all agents to interact with all teammates at all times, there is no reason to expect the optimal ND-POMDP method (GOA [19]) to outperform the algorithms presented here. We report on each benchmark the optimal value $v_0(\eta_0)$ together with the running time in seconds for different planning horizons.

The MPS variants were run on a Mac OSX machine with 2.4GHz Dual-Core Intel and 2GB of RAM available. We solved the constraint optimization problems using the aolib library\(^1\). The bilinear programming approach (listed as

\(^1\)The aolib library is available at the following website: