As such, the $\tau$-th value function $v_{\tau}$ can be built from a $(\tau + 1)$-th value function $v_{\tau+1}$ as follows:

$$v_{\tau}(\eta_{\tau}) = \max_{\sigma_{\tau}} [L_{\tau}, v_{\tau+1}](\eta_{\tau}),$$

$$v_{\tau}^*(\eta_{\tau}) = 0.$$  \hfill (2)

In our setting, Equations (2) denote the optimality equations. It is worth noting that the decentralized Markov policy solution $\pi = \langle \sigma_0, \ldots, \sigma_{T-1} \rangle$ of the optimality equations is greedy with respect to value functions $v_0, \ldots, v_{T-1}$.

5 Markov Policy Search

In this section, we compute optimal decentralized Markov policy $\langle \sigma_0^*, \ldots, \sigma_{T-1}^* \rangle$ given initial state occupancy $\eta_0$ and planning horizon $T$. Note that while state occupancies are used to calculate heuristics in this algorithm, the final choices at each step do not depend on the state occupancies. That is, the result is a nonstationary policy for each agent mapping local observations to actions at each step.

We cast decentralized MDPs $(S, A, p, r)$ as continuous and deterministic MDPs where: states are state occupancy distributions $\eta_{\tau}$; actions are decentralized Markov policies $\sigma_{\tau}$; the update-rules $\chi_{\tau}(\cdot, \sigma_{\tau-1})$ define transitions; and mappings $r_{\tau}(\cdot, \sigma_{\tau})$ denote the reward function. So, techniques that apply in continuous and deterministic MDPs also apply in decentralized MDPs with independent transitions and observations. For the sake of efficiency, we focus only on optimal techniques that exploit the initial information $\eta_0$.

The learning real-time A* (LRTA*) algorithm can be used to solve deterministic MDPs [13]. This approach updates only states that agents actually visit during the planning stage. Therefore, it is suitable for continuous state spaces. Algorithm 1, namely Markov Policy Search (MPS), illustrates an adaptation of the LRTA* algorithm for solving decentralized MDPs with independent transitions and observations. The MPS algorithm relies on lower and upper bounds $\bar{v}_{\tau}$ and $\bar{v}_{\tau}$ on the exact value functions for all planning horizons $\tau = 0, \ldots, T - 1$.

We use the following definitions. Q-value functions $q_{\tau}(\eta_{\tau}, \sigma_{\tau})$ denote rewards accrued after taking decision rule $\sigma_{\tau}$ at state occupancy $\eta_{\tau}$ and then following the policy defined by upper-bound value functions for the remaining planning horizons. We denote $\Psi_{\tau}(\eta_{\tau}) = \{ \sigma_{\tau} \}$ to be the set of all stored decentralized Markov decision rules for state occupancy $\eta_{\tau}$. Thus, $v_{\tau}(\eta_{\tau}) = \max_{\sigma_{\tau} \in \Psi_{\tau}(\eta_{\tau})} q_{\tau}(\eta_{\tau}, \sigma_{\tau})$ represents the upper-bound value at state occupancy $\eta_{\tau}$. Formally, we have that $\bar{v}_{\tau}(\eta_{\tau}, \sigma_{\tau}) = [L_{\tau}, \bar{v}_{\tau+1}](\eta_{\tau})$.

Next, we describe two variants of the MPS algorithm. The exhaustive variant replaces states by state occupancy distributions, and actions by decentralized Markov decision rules in the LRTA* algorithm. The second variant uses a constraint optimization program instead of the memory demanding exhaustive backup operation that both the LRTA* algorithm and the exhaustive variant use.

5.1 The exhaustive variant

The exhaustive variant consists of three major steps: the initialization step (line 1); the backup operation step (line 5); and the update step (lines 6 and 8). It repeats the execution of these steps until convergence ($\bar{v}_0(\eta_0) - \bar{v}_0(\eta_0) \leq \epsilon$).

Algorithm 1: The MPS algorithm.

```
begin
  Initialize bounds $\bar{v}$ and $\bar{v}$.
  while $\bar{v}_0(\eta_0) - \bar{v}_0(\eta_0) > \epsilon$ do
    MPS-TRIAL($\eta_0$)
  MPS-TRIAL($\eta_\tau$) begin
    while $\bar{v}_{\tau}(\eta_\tau) - \bar{v}_{\tau}(\eta_\tau) > \epsilon$ do
      $\bar{v}_{\tau}(\eta_\tau) \leftarrow \arg \max_{\sigma_{\tau}} q_{\tau}(\eta_\tau, \sigma_{\tau})$
      Update the upper bound value function.
      MPS-TRIAL($\chi_{\tau+1}(\eta_\tau, \bar{v}_{\tau}(\eta_\tau))$)
    Update the lower bound value function.
  end
end
```

Initialization. We initialize lower bound $\bar{v}_{\tau}$ with the $\tau$-th value function of any decentralized Markov policy, such as a randomly generated policy $\pi_{\text{rand}} = \langle \sigma_{\text{rand},0}, \ldots, \sigma_{\text{rand},T-1} \rangle$, where $\bar{v}_{\tau} = v_{\sigma_{\text{rand},0}, \ldots, \sigma_{\text{rand},T-1}}$. We initialize the upper bound $\bar{v}_{\tau}$ with the $\tau$-th value function of the underlying MDP. That is, $\pi_{\text{mdp}} = \langle \sigma_{\text{mdp},0}, \ldots, \sigma_{\text{mdp},T-1} \rangle$, where $\bar{v}_{\tau} = v_{\sigma_{\text{mdp},0}, \ldots, \sigma_{\text{mdp},T-1}}$.

The exhaustive backup operation. We choose decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$, which yields the highest value $\bar{v}_{\tau}(\eta_\tau)$ through the explicit enumeration of all possible decentralized Markov decision rules $\sigma_{\tau}$. We first store all decentralized Markov decision rules $\sigma_{\tau}$ for each visited state occupancy $\eta_\tau$ together with corresponding values $\bar{q}_{\tau}(\eta_\tau, \sigma_{\tau})$. Hence, the greedy decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$ is arg max$_{\sigma_{\tau}} \bar{q}_{\tau}(\eta_\tau, \sigma_{\tau})$ at state occupancy $\eta_\tau$.

Update of lower and upper bounds. We update the lower bound value function based on decentralized Markov policies $\pi_{\text{greedy}} = \langle \sigma_{\text{greedy},0}, \ldots, \sigma_{\text{greedy},T-1} \rangle$ selected at each trial. If $\pi_{\text{greedy}}$ yields a value higher than that of the current lower bound, $\bar{v}_{\tau}(\eta_\tau) < v_{\pi_{\text{greedy}}}(\eta_\tau)$, we set $\bar{v}_{\tau} = v_{\pi_{\text{greedy},0}, \ldots, \pi_{\text{greedy},T-1}}$ for $\tau = 0, \ldots, T - 1$, otherwise we leave the lower bound unchanged. We update the upper bound value function based on decentralized Markov decision rules $\sigma_{\text{greedy},\tau}$ and the $(\tau + 1)$-th upper-bound value function $\bar{v}_{\tau+1}$, as follows $\bar{v}_{\tau}(\eta_\tau) = [L_{\tau}, \bar{v}_{\tau+1}](\eta_\tau)$.

Theoretical guarantees. The exhaustive variant of MPS yields both advantages and drawbacks. On the one hand, it inherits the theoretical guarantees from the LRTA* al-