Scalable Anytime Planning for Multi-Agent MDPs

Shushman Choudhury*, Jayesh K. Gupta*
Stanford University
{shushman,jkg}@cs.stanford.edu

Peter Morales
Microsoft
pmorales@microsoft.com

Mykel J. Kochenderfer
Stanford University
mykel@stanford.edu

ABSTRACT

We present a scalable tree search planning algorithm for large multi-agent sequential decision problems that require dynamic collaboration. Teams of agents need to coordinate decisions in many domains, but naive approaches fail due to the exponential growth of the joint action space with the number of agents. We circumvent this complexity through an anytime approach that allows us to trade computation for approximation quality and also dynamically coordinate actions. Our algorithm comprises three elements: online planning with Monte Carlo Tree Search (MCTS), factored representations of local agent interactions with coordination graphs, and the iterative Max-Plus method for joint action selection. We evaluate our approach on the benchmark SysAdmin domain with static coordination graphs and achieve comparable performance with much lower computation cost than our MCTS baselines. We also introduce a multi-drone delivery domain with dynamic, i.e., state-dependent coordination graphs, and demonstrate how our approach scales to large problems on this domain that are intractable for other MCTS methods. We provide an open-source implementation of our algorithm at https://github.com/JuliaPOMDP/FactoredValueMCTS.jl.

ACM Reference Format:
Shushman Choudhury*, Jayesh K. Gupta*, Peter Morales, and Mykel J. Kochenderfer. 2021. Scalable Anytime Planning for Multi-Agent MDPs. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online., May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Coordination is crucial for effective decision-making in cooperative multi-agent systems with a shared objective. Various real-world problems like formation control [31], package delivery [12], and firefighting [32] require a team of autonomous agents to perform a common task. Such cooperative sequential decision-making problems can be modeled as a multi-agent Markov decision process (MMDP) [7], an extension of the Markov decision process (MDP) [20]. MMDPs can be reduced to centralized single-agent MDPs with a joint action space for all agents. Such reductions often make large problems intractable because the action space grows exponentially with the number of agents. Solving independent MDPs for all agents yields suboptimal behavior in problems where reasoning about the effects of joint actions is necessary for better performance [28].

Many previous MMDP approaches have tried to balance these extremes of optimality and efficiency. In the *offline* setting, these include ad hoc function decomposition approaches, such as Value Decomposition Networks [39] and QMIX [36], or parameter sharing in decentralized policy optimization [19]. Guestrin et al. [17] introduced the concept of a coordination graph to reason about joint value estimates from a factored representation, while Kok and Vlassis [23] used approximations to scale these ideas to larger problems. Monte Carlo Tree Search (MCTS) [8], a common approach to *online* planning, has been combined with coordination graphs in Factored Value MCTS [1]. However, Factored Value MCTS coordinates actions with an exact Variable Elimination (Var-El) step, which negates the anytime nature of MCTS planning.

The key idea of this paper is to *recover the anytime nature of MCTS planning for MMDPs requiring coordination and also scale to larger teams of agents.* To that end, we propose combining Max-Plus action selection, introduced by Vlassis et al. [41], with the factored value MCTS of Amato and Oliehoek [1]. We do so for many reasons. Unlike Var-El, which is exact, Max-Plus is an iterative procedure and allows for truly anytime behavior that can trade computation for approximation quality. The representation of Max-Plus is much more efficient than that of Var-El for using dynamic, i.e., state-dependent, coordination graphs (state-dependent data-structures are a key benefit of online planning for MDPs). Finally, Max-Plus can scale to much larger and denser coordination graphs than Var-El, and it can be distributed for additional scalability [24].

We present a scalable anytime MMDP planning algorithm called Factored Value MCTS with Max-Plus. On the standard SysAdmin benchmark domain [17], with static coordination graphs, we demonstrate that our approach performs as well as or better than Factored Value MCTS with Var-El [1] and is much faster for the same tree search hyperparameters. We also introduce a new MMDP domain, Multi-Drone Delivery, with dynamic coordination graphs. On the second domain, we show how our approach scales to problem sizes that are entirely intractable for other MCTS variants, while also achieving better performance on smaller problem sizes.

2 BACKGROUND AND RELATED WORK

We first review MDPs and their multi-agent formulation. We then describe how coordination graphs can efficiently exploit locality of interactions in multi-agent problems. Finally, we discuss how to use coordination graphs to solve multi-agent MDPs.

2.1 Multi-Agent Markov Decision Processes

An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ is the transition function, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The objective for solving an MDP is to obtain a *policy*, $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ that specifies a probability distribution over actions for the agent to take from its current state to maximize its *value*, i.e. its expected cumulative reward. An action-value function $Q(s,a)$ defines the

* Equal contribution.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online..

expected cumulative reward after taking action a in state s before following the specified policy.

We focus on decision-making settings where multiple agents cooperate to achieve a shared objective [7]. Such problems are multi-agent Markov decision processes (MMDPs), where each agent takes an individual action and the controller policy observes the states of all agents. In principle, we can solve an MMDP as a standard MDP with a joint action space $\mathcal{A} = \prod_i \mathcal{A}_i$ [35]. There exist both offline and online methods for computing such MDP policies [3].

Offline methods pre-compute a policy over the entire state space (exactly or approximately) and query the policy during execution. Various exact offline methods exist, but reinforcement learning has emerged as an attractive solution technique due to the complexity of planning in large MMDPs [40]. Reinforcement Learning approaches attempt to compute an effective value function $Q(s, a)$ or a policy $\pi(a \mid s)$ through repeated interaction with the environment model. They still have difficulty with the size of the joint action space, which is exponential in the number of agents. A common strategy is to *decentralize* the policy or value function, such that each only depends on the actions of a single agent [19, 36, 39]. Unfortunately, such decentralization approaches are often suboptimal for coordination and encounter exploration bottlenecks due to uncooperative random actions from the agents [6].

Online methods use an alternative strategy to deal with the complexity of multi-agent planning; they interleave planning and execution by focusing only on states that are reachable for the current state, while computing the next action to take. Monte Carlo Tree Search (MCTS) is the predominant framework for online planning and has succeeded in a variety of domains [8], including in multi-player contexts [30, 44]. The *anytime* nature of MCTS (search depth and number of simulations) allows us to trade computation time for approximation quality. However, the exponentially large action space of MMDPs can still be a bottleneck for the naive application of MCTS techniques [9]. Dec-MCTS tries to work around this bottleneck but does not apply to action-dependent stochastic transitions of an MDP, as it directly chooses the next state [4].

2.2 Coordination Graphs and Variable Elimination

Several real-world multi-agent systems demonstrate *locality of interaction*, i.e. the outcome of an agent's action depends only on the actions of a subset of other agents. The coordination graph (CG) structure is often used to encode such interactions [17, 18]. A CG for a multi-agent system has one node per agent, and edges connect agents if their payoffs depend on each other. For now, we assume a stateless or single-shot decision setting (rather than sequential). The CG structure induces a set of payoff components, *where each component is associated with a clique*, i.e., a subset of agents that are all mutually connected.

For CGs in multi-agent settings, we assume that we can factor the global payoff for a joint action as the sum of local component payoffs, i.e. $Q(\overline{a}) = \sum_c Q_c(\overline{a}_c)$, where $\overline{a}$ is the global joint action, and $\overline{a}_c$ is the local component action (the projection of $\overline{a}$ corresponding to component c). Given this factored representation and the local component payoffs, we can compute exactly the best joint

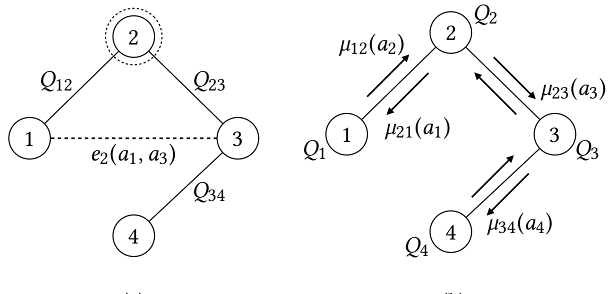


Figure 1: A coordination graph for an MMDP with 4 agents. (a) Eliminating agent 2 in Var-El introduces an edge between nodes (agents) 1 and 3 and a new payoff function e_2 (b) In Max-Plus, agents passes messages along the graph edges; the messages are functions of the actions of the receiving agent, e.g., agent 1 sends $\mu_{12}(a_2)$ to agent 2.

action, $\operatorname{argmax}_{\overline{a}} Q(\overline{a})$, with the Variable Elimination (Var-El) algorithm originating from the probabilistic inference literature [18]. Computing the optimal joint action in a CG is equivalent to obtaining the maximum a posteriori configuration in an undirected probabilistic graphical model [41].

Consider the 4-agent CG in Figure 1a. Here, $Q(\overline{a}) = Q_{12}(a_1, a_2) + Q_{23}(a_2, a_3) + Q_{34}(a_3, a_4)$, where a_i is the action variable for agent i. In Var-El, we *eliminate*, i.e., maximize over variables one at a time by collecting the local payoffs that depend on them. For instance, if we start with agent 2, then

$$\max_{a_1,a_3,a_4} Q_{34}(a_3, a_4) + \max_{a_2} [Q_{12}(a_1, a_2) + Q_{23}(a_2, a_3)] \qquad (1)$$

is the first elimination. The optimal choice for agent 4 depends on a_2 and a_3. The internal max expression is summarized by a new intermediate payoff function $e_2(a_1, a_3) = \max_{a_2} [Q_{12}(a_1, a_2) + Q_{23}(a_2, a_3)]$ and a new edge between 1 and 3, after which the algorithm continues with Q_{34} and e_2. After all eliminations, we recover the action for each agent by maximizing the conditional functions in reverse, finally obtaining the optimal joint action. Var-El is exponential in the induced width of the CG, which depends on the elimination order [13].

Although most works in the literature assume a domain-dependent static coordination graph structure, some incorporate state-dependent or dynamic CGs [43], including learning the CG structures [22, 27].

2.3 Scalable MMDP Methods with Coordination Graphs

In the *offline* context of tabular RL methods, Kok and Vlassis [23] explored action inference with predefined static coordination graphs over factorized value functions; Böhmer et al. [6] extended these ideas to the neural network function approximation regime. We focus on *anytime online* planning approaches to solving MMDPs. Amato and Oliehoek [1] provide an online planning solution by combining the idea of coordination graphs and factored values with MCTS. Although they apply their algorithm to partially observed MDPs, the key ideas are the same for the fully observed case. Monte Carlo planning *estimates* quantities by exploring from the current state and gathering relevant statistics through interactions with a simulated generative model of the environment [38]. These statistics typically track the average simulated reward obtained for trying an individual or joint action, the frequency of action attempts for Upper Confidence Bound or UCB exploration [21], and the number of occurences of the individual or joint state.

Amato and Oliehoek [1] maintain *local component statistics*, i.e. the mean payoff of a local component action $\overline{a}_e$ and the number of times it was attempted in that component; they call this *mixture of experts optimization*, albeit with a simple maximum likelihood estimator expert. For instance, during tree search from the current joint state $\overline{s} \equiv \{s_i\}$ (where s_i is the state of agent i), suppose the system simulates a joint action $\overline{a}$ and obtains a reward vector $\overline{r}$. Then, in a particular CG component e and the corresponding local subset of the joint action $\overline{a}_e$, they augment the local component action frequency statistic $N(\overline{s}, \overline{a}_e)$ by 1 and update the local component payoff statistic $Q_e(\overline{s}, \overline{a}_e)$ as

$$Q_e(\overline{s}, \overline{a}_e) \doteq Q_e(\overline{s}, \overline{a}_e) + \frac{\overline{r}_e - Q_e(\overline{s}, \overline{a}_e)}{N(\overline{s}, \overline{a}_e)}, \qquad (2)$$

which is a standard running average update. The UCB exploration step uses the current statistics to select joint actions, i.e.,

$$\max_{\overline{a}} \sum_e U_e(\overline{s}, \overline{a}_e) = \max_{\overline{a}} \sum_e Q_e(\overline{s}, \overline{a}_e) + c \cdot \sqrt{\frac{\log N(\overline{s})}{N(\overline{s}, \overline{a}_e)}}, \qquad (3)$$

where $N(\overline{s})$ is the visit frequency for state $\overline{s}$. Given these local component payoffs, i.e., the Q_e functions, their method computes the best joint action at the next time-step through Variable Elimination over the CG, as in Section 2.2. *Consequently, it loses the anytime property of MCTS because exact variable elimination cannot be stopped at an intermediate step.* Although Vlassis et al. [41] explored various anytime algorithms for action selection with coordination graphs, they did not investigate their interaction with online planning algorithms like MCTS.

3 ANYTIME FACTORED-VALUE MONTE CARLO TREE SEARCH

We now discuss our method for anytime multi-agent MDP planning with coordination graphs, *Factored-Value Monte Carlo Tree Search with Max-Plus*. To apply the mixture of experts optimization to each node of the search tree, we must define the factored statistics to maintain for each node. Given a potentially state-dependent undirected *coordination graph* (CG), $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, we factor the CG-induced global payoff at the current state, $\overline{s}$, as follows:

$$Q(\overline{a}) = \sum_{i \in \mathcal{V}} Q_i(a_i) + \sum_{(i,j) \in \mathcal{E}} Q_{ij}(a_i, a_j). \qquad (4)$$

Here, Q_{ij} is a local payoff function for agents i and j connected by edge (i, j), while Q_i is an individual utility function for agent i, if applicable to the domain. *All state-dependent quantities in this section's equations are implicitly for the current joint state* $\overline{s}$.

Exploiting the duality between computing the maximum a posteriori configuration in a probabilistic graphical model and the optimal joint action in a CG, Vlassis et al. [41] introduced the Max-Plus algorithm for computing the joint action via message passing. Each node, i.e., agent, iteratively dispatches messages to its neighbours $j \in \Gamma(i)$ in the CG (Figure 1b). A message from agent i is a scalar-valued function of the action space of *receiving agent* j, i.e.,

$$\mu_{ij}(a_j) = \max_{a_i} \Big\{ Q_i(a_i) + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus \{j\}} \mu_{ki}(a_i) \Big\}, \qquad (5)$$

where $\Gamma(i)$ is the set of neighbors of i. Agents exchange messages until convergence or for a maximum number of rounds. Finally, each agent computes its optimal action individually, i.e.,

$$a_i^* = \operatorname*{argmax}_{a_i} \Big\{ Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) \Big\} \qquad (6)$$

Max-Plus is equivalent to belief propagation in graphical models [33] and its time complexity scales linearly with the CG size (the number of edges); it is more suitable for real-time systems and more tractable for large numbers of agents than Var-El.

Similar to Factored Value MCTS with Var-El, our method with Max-Plus (that we illustrate in Figure 2) is more efficient than a naive application of MCTS with the joint action space, since it retains fewer statistics and performs efficient action selection. For the rest of this section, we will discuss the key differences from the prior work of Amato and Oliehoek [1], which underscore how our approach is more suitable than it for large MMDPs.

3.1 UCB Exploration with Max-Plus

The key implementation issue for extending MCTS to factored value functions and coordination graphs is that of action exploration as per the Upper Confidence Bound (UCB) strategy. In the Var-El case, Amato and Oliehoek [1] added the exploration bonus using component-wise statistics during each elimination step in Equation (3). We cannot apply this strategy with Max-Plus as it does not use components. In contrast, it has two distinct phases of computation. The first is message passing per edge in Equation (5), followed by action selection per node in Equation (4). We use these two phases to define how our algorithm explores.

Edge Exploration: Analogous to the edge payoff statistics Q_{ij}, we keep track of corresponding frequency statistics N_{a_i,a_j} (for pairwise actions). The natural exploration strategy over edges is to add the bonus to Equation (5) as follows:

$$\mu_{ij}(a_j) = \max_{a_i} \Bigg\{ Q_i(a_i) + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus \{j\}} \mu_{ki}(a_i) + c\sqrt{\frac{\log(N+1)}{N_{a_i,a_j}}} \Bigg\}. \qquad (7)$$

Adding this bonus during the message passing rounds can cause divergence for cyclic graphs with any cycle of length less than the number of rounds. Figure 3 illustrates intuition for this divergent behavior with a simple triangle graph. The bonuses accumulate in successive rounds for messages in either direction along the cycle, making the effective bonus proportional to the total number of rounds (divided by cycle length). Therefore, we only augment each message once *after the final round of message passing*.

Node Exploration: We maintain individual action frequency statistics N_{a_i} and modify Equation (4) to add a node exploration

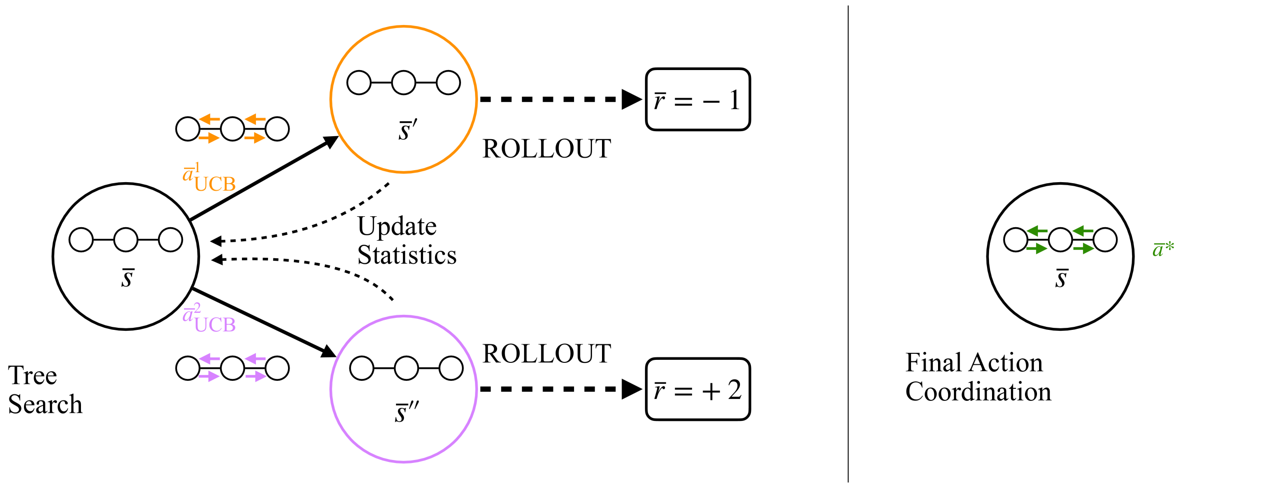


Figure 2: Our anytime MMDP planning algorithm, Factored Value MCTS with Max-Plus, computes the best joint action $\bar{a}^*$ for the current joint state $\bar{s}$. The tree search uses an Upper Confidence Bound (UCB) exploration bonus during action selection, while the final action coordination does not.

bonus during the action selection:

$$a_i^* = \operatorname*{argmax}_{a_i} \left\{ Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) + c\sqrt{\frac{\log(N+1)}{N_{a_i}}} \right\} \qquad (8)$$

Note that the joint-action payoff $Q(\bar{a})$ can be factorized over the CG nodes and edges as in Equation (4), but the joint-action exploration bonus $c\sqrt{\frac{\log N(\bar{s})}{N(\bar{s},\bar{a})}}$ cannot. Therefore, the node and edge exploration strategies we have defined here are heuristic choices that we make and will evaluate empirically through an ablation. Our exploration strategies differ from the component-wise exploration of Equation (3) in the previous work that uses Var-El, because we do not consider cliques/components in the CG, only nodes and edges.

3.2 Other differences from FV-MCTS with Variable Elimination

Convergence: For graphs without cycles, Max-Plus converges to a fixed point in finitely many iterations [33]. For cyclic graphs, there are no such guarantees in general [42]. However, cyclic message passing can work well in practice [29].

Agent Utilities: The Max-Plus global payoff in Equation (4) includes a utility function Q_i for each individual agent. The FV-MCTS with Var-El has no such individual utility (unless a node has degree 0 in the CG). If such agent utilities were known or learned *independent of the payoffs*, we would naturally use them during action coordination. However, in FV-MCTS we estimate all statistics from the rewards obtained during tree search with a simulated environment model; the environment model returns precisely one reward vector for each joint state-action pair.

We already account for the simulated rewards in tree search through the Q_{ij} local payoff statistics in Equation (2). We do not receive independent per-agent rewards, so utility statistics would be derived from the same information we use for the payoff statistics.

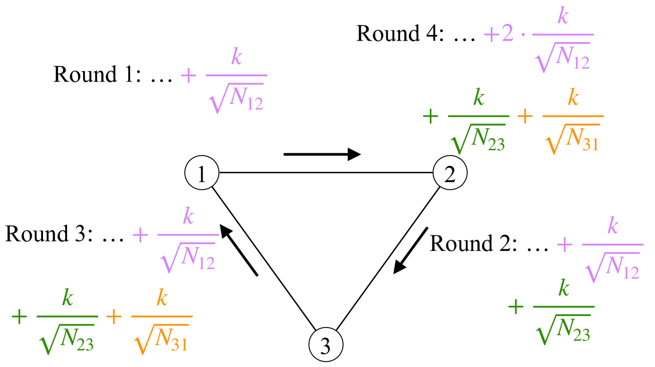

Round 1: $\dots + \frac{k}{\sqrt{N_{12}}}$
Round 4: $\dots + 2 \cdot \frac{k}{\sqrt{N_{12}}} + \frac{k}{\sqrt{N_{23}}} + \frac{k}{\sqrt{N_{31}}}$
1
2
3
Round 3: $\dots + \frac{k}{\sqrt{N_{12}}} + \frac{k}{\sqrt{N_{23}}} + \frac{k}{\sqrt{N_{31}}}$
Round 2: $\dots + \frac{k}{\sqrt{N_{12}}} + \frac{k}{\sqrt{N_{23}}}$

Figure 3: For coordination graphs with cycles, adding an edge exploration bonus to the messages *at every round* can lead to divergent behavior. The exploration bonuses accumulate over rounds from one node to the next (clockwise or anti-clockwise along the cycle). If the number of rounds is greater than the length of a cycle (usually true), the effective relevant exploration bonus for each edge gets compounded each time the messages loop back around. We abuse some notation for convenience, i.e., N_{ij} is a counting function for the pairwise actions of agents i and j.

Our experiments compare the benefit of these derived individual agent (node) utilities, in addition to local edge payoffs. We maintain separate statistics N_i and Q_i for the per-agent frequencies and utilities respectively and estimate them from the joint rewards during tree search; the corresponding updates are $N_i(\bar{s}, \bar{a}_i) = N_i(\bar{s}, \bar{a}_i) + 1$ and $Q_i(\bar{s}, \bar{a}_i) = Q_i(\bar{s}, \bar{a}_i) + \frac{\bar{r}_i - Q_i(\bar{s},\bar{a}_i)}{N_{\bar{a}_i}}$ for an agent i. The results in Section 4.1 demonstrate how *including derived node utilities* enables better empirical performance.

Algorithm 1 Factored Value MCTS with Max-Plus

Require: time limit, depth, exploration constant c, state $\bar{s}$

```
Initialize $N_i, Q_i$                                   ▷ Node statistics
Initialize $N_{ij}, Q_{ij}$                             ▷ Edge statistics

function FV-MCTS-MP($\bar{s}$, depth)
    while time limit not reached
        Simulate($\bar{s}$, depth)
    $\bar{a}^* \leftarrow$ MaxPlus(0)                    ▷ No exploration here
    return $\bar{a}^*$                                   ▷ Best joint action
function Simulate($\bar{s}$, depth)
    if depth = 0
        return 0
    $\bar{a} \leftarrow$ MaxPlus($c$)
    $\bar{s}', \bar{r} \sim T(\bar{s}, \bar{a}), R(\bar{s}, \bar{a})$        ▷ Generative model
    $\bar{q} \leftarrow \bar{r} + \gamma \cdot$ Simulate($\bar{s}'$, depth − 1)
    UpdateStats($\bar{s}, \bar{a}, \bar{q}$)
function UpdateStats($\bar{s}, \bar{a}, \bar{q}$)
    for every agent $i$
        $N_i(\bar{s}, a_i)$ += 1
        $Q_i(\bar{s}, a_i)$ += $\frac{q_i - Q_i(\bar{s}, a_i)}{N_i(\bar{s}, a_i)}$
    for every edge $(i, j) \in \mathcal{G}(\bar{s})$
        $N_{ij}(\bar{s}, a_i, a_j)$ += 1
        $q_e \leftarrow q_i + q_j$
        $Q_{ij}(\bar{s}, a_i, a_j)$ += $\frac{q_e - Q_{ij}(\bar{s}, a_i, a_j)}{N_{ij}(\bar{s}, a_i, a_j)}$
```

Algorithm 2 MaxPlus Action Selection

Require: Coordination Graph $\mathcal{G}(s) = \langle \mathcal{V}, \mathcal{E} \rangle$; state node statistics N, Q; max iterations M; flags (exploration; normalization)

```
function MaxPlus(c)
    for $t \leftarrow 1$ to $M$
        $\mu_{ij}(a_j) = \mu_{ji} = 0$ for $(i, j) \in \mathcal{E}, a_i \in \mathcal{A}_i, a_j \in \mathcal{A}_j$
        for every agent $i$
            for all neighbors $j \in \Gamma(i)$
                Compute $\mu_{ij}(a_j)$ via Equation (5)
                if message normalization
                    $\mu_{ij}(a_j)$ −= $\frac{1}{|\mathcal{A}_j|} \sum_{a_j \in \mathcal{A}_j} \mu_{ij}(a_j)$
                send message $\mu_{ij}(a_j)$ to agent $j$
                if $\mu_{ij}(a_j)$ close to previous message
                    break
        for every agent $i$
            if edge exploration
                for all neighbors $j \in \Gamma(i)$
                    Compute $\mu_{ij}(a_j)$ via Equation (7)
            $q_i(a_i) = Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)$
            if node exploration
                $q_i(a_i)$ += $c\sqrt{\frac{\log(N+1)}{N_i(a_i)}}$
            $a_i' = \arg\max_{a_i} q_i(a_i)$
        if time limit reached
            break
    return $\bar{a}'$
```

Dynamic Coordination Graphs: Recall that MCTS (and online MDP planning in general) can use computational structures that vary with the current state. *For FV-MCTS with Var-El, state-dependent or dynamic CGs are not feasible* because eliminating an agent can change the intermediate CG topology during action coordination (by adding edges). It is not tractable to maintain statistics for the set of all possible CG components, the size of which is exponential in the number of agents [45]. On the other hand, Max-Plus only maintains statistics for at most all CG edges, over which the messages are sent. Therefore, dynamic CGs can be used seamlessly [43].

Memory Complexity of Statistics: Factored Value MCTS collects frequency and payoff statistics for each unique joint state encountered during tree search. Assume the same action set $\mathcal{A}$ for each agent, and a CG with $|\mathcal{V}|$ nodes (agents), $|\mathcal{E}|$ edges, and C local components or cliques. Then, the memory complexity of per-state statistics for Max-Plus is $O(|\mathcal{V}||\mathcal{A}| + |\mathcal{E}||\mathcal{A}|^2)$ (the first term only applies if we track per-node utilities). In contrast, the per-state memory for Var-El statistics is $O(\sum_{c \in C} |\mathcal{A}|^{|\mathcal{V}|_c} \cdot |\mathcal{V}|_c)$, where $|\mathcal{V}|_c$ is the size of local component c. For a CG that is connected (typically the case), the memory complexity for Var-El is at least $O(|\mathcal{E}||\mathcal{A}|^2)$, which is the dominant term for Max-Plus, and more generally is exponential in the largest clique. *Therefore, Max-Plus is more memory-efficient than Var-El.* The experiments in Section 4.2 empirically support this claim by showing how our algorithm solves problems that cause out-of-memory issues for the Var-El baseline.

Distributed Implementation: Unlike with Var-El, *we can execute Max-Plus in a distributed manner* by sending messages in parallel, albeit incurring additional communication complexity. Such an implementation can allow further scalability with available compute. Note that this is distinct from full decentralization wherein the agent actions can be computed independently.

We outline our approach in Algorithm 1 as well as the Max-Plus routine in Algorithm 2.

4 EXPERIMENTS AND RESULTS

We used cumulative discounted return as the primary metric to evaluate our approach, Factored Value MCTS with Max-Plus (FV-MCTS-MP). Our most relevant baseline is Factored Value MCTS with Variable Elimination (FV-MCTS-Var-El). We also compared against standard MCTS (with no factorization), independent Q-learning (IQL), and a random policy. Besides measuring performance, we examined the effect of different exploration schemes on the performance of FV-MCTS-MP (as we discussed in Section 3.2) and the problem size on MCTS computation time. The appendix of the extended version[10] provides performance results for FV-MCTS (both variants) with different hyperparameters. Both of our experimental domains represent a range of MMDP problems and underlying coordination graphs (CGs). The extended version and source-code for experiments are available at https://sites.google.com/stanford.edu/fvmcts/. All implementations and simulations are in Julia [5, 15].

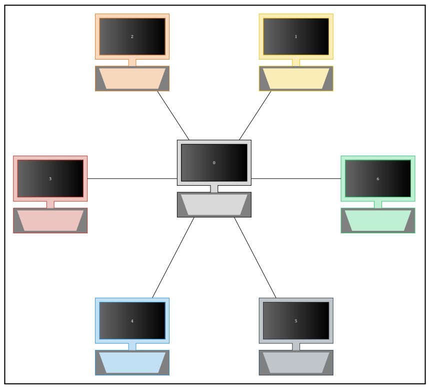
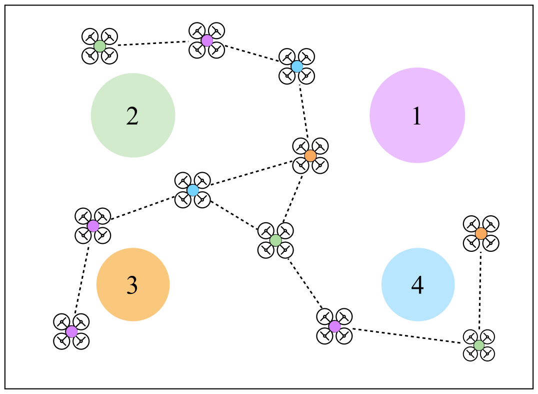



Figure 4: Our two experimental domains: on the left, SysAdmin with the star topology and on the right, Multi-Drone Delivery, where dotted lines illustrate a subset of the Coordination Graph edges for the current state (for clarity, we omit some edges between drones of the same color, i.e., assigned to the same goal).

We will show qualitatively how our approach recovers the true anytime nature of MCTS by using Max-Plus rather than Var-El. However, *there are many confounds for quantitatively evaluating the anytime property*. Our metric is the average discounted return over the episode, where the Max-Plus routine is called several times; typical anytime evaluation reports improving solution quality with more compute time for a single call to a method. MCTS itself has several parameters that affect the computation-vs-quality tradeoff, such as tree depth, exploration constant, and number of trials. With dynamic CGs as in our multi-drone delivery domain, the same Max-Plus parameters leads to different computation times. Note that our reference for Max-Plus does evaluate its anytime property in a one-shot decision-making domain that does not have any of the above confounds [41].

4.1 SysAdmin Domain

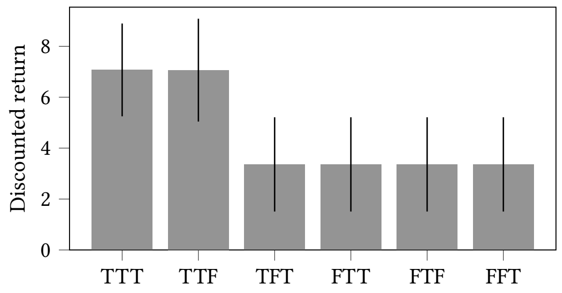
Our first domain is a standard MMDP benchmark: SysAdmin [18]. Each agent i represents a machine in a network with two state variables: Status $S_i \in \{\text{GOOD}, \text{FAULTY}, \text{DEAD}\}$, and Load $L_i \in \{\text{IDLE}, \text{LOADED}, \text{SUCCESS}\}$. A DEAD machine increases the probability that its neighbor also dies. The system gets a reward of 1 if a process terminates successfully, processes take longer when status is FAULTY, and a DEAD machine loses the process. Each agent must decide whether to reboot its machine, in which case the Status becomes GOOD and any running process is lost. The discount factor, γ used in all the experiments is 0.9. All evaluations have been averaged over 40 runs. Error bars indicate standard deviations. Figure 4 illustrates the star network topology for SysAdmin; we also use the ring topology as well as the ring-of-rings topology.

Exploration Schemes for FV-MCTS-MP: The three knobs affecting exploration in FV-MCTS-MP are per-agent utility, node bonus, and edge bonus. We compared the discounted return of variants that either use or ignore per-agent utilities, and use either or both bonuses. Figure 5 demonstrates that the combination of *agent utilities and only node exploration (TTF) is enough*; including the edge bonus as well (TTT) does not have much effect. All other schemes are significantly poorer. Therefore, we used the TTF variant of FV-MCTS-MP to compare against the other baselines. Lack of significant difference between some of the exploration strategies has more to do with the small action space of the SysAdmin domain.

FV-MCTS-MP compared to baselines: For all three SysAdmin topologies (and corresponding fixed CGs), we varied the number of machines (agents) and compared the performance of all methods in Figure 6. With fewer agents, all MCTS methods perform similarly to each other and better than Q-Learning. However, with more agents, standard MCTS runs out of memory even on our 128GB RAM machine, as expected for large joint action spaces. Both Factored Value MCTS variants perform comparably on larger problems (on ring-of-rings our Max-Plus variant was better). However, as we discuss subsequently, FV-MCTS-Var-El is much slower than FV-MCTS-MP, e.g., taking approximately 35 s versus 16 s for 32 agents on a single-threaded implementation in the Ring topology. *Therefore our approach strictly dominates the Var-El baseline on the performance-time tradeoff.*

Effect of Hyperparameters: We performed ablation experiments, varying one of exploration constant c, tree search exploration depth d and number of Monte Carlo rollouts n, while keeping the rest of the hyperparameters constant. Low values of n adversely effected the performance a little for both FV-MCTS-Var-El and FV-MCTS-MP, while *low values of c vastly degraded the performance of FV-MCTS-Var-El only*. The difference in performance was not significant over a range of values of d. The appendix of the extended version[10], shows results for 32 agents. Similar results hold for the other domains as well as different numbers of agents.

Computation Time: For the same tree search hyperparameters with number of iterations fixed as 16000, exploration constant as 20 and tree search depth as 20, we compared the average time taken for each action for different number of agents in the coordination graphs. For a fair comparison, we used a single threaded implementation. As demonstrated in Figure 7, we found FV-MCTS-MP to be consistently faster than FV-MCTS-Var-El. Although MCTS



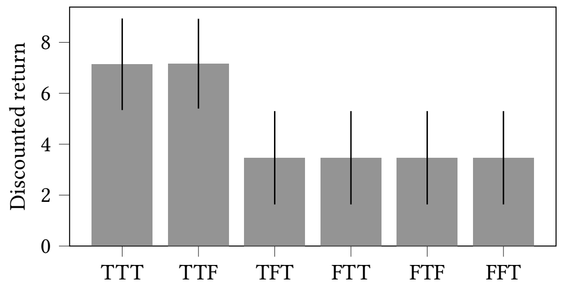
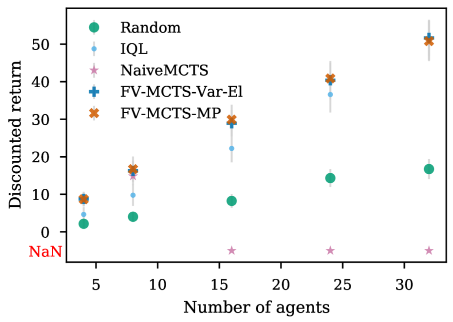
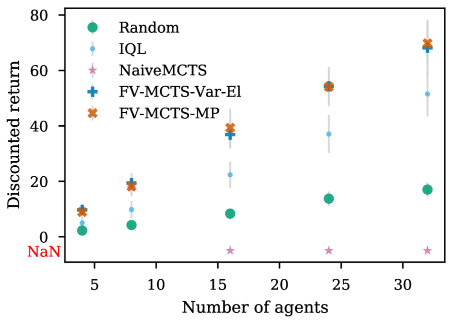


Figure 5: The performance of FV-MCTS-MP varies with different combinations of exploration strategies for the 4-agent Sysadmin on Ring (left) and Star (right) topologies. The True/False (T/F) labels correspond to Agent Utilities, Node Exploration and Edge Exploration in order, e.g. TTF implies agent utilities and only node (but not edge) exploration.





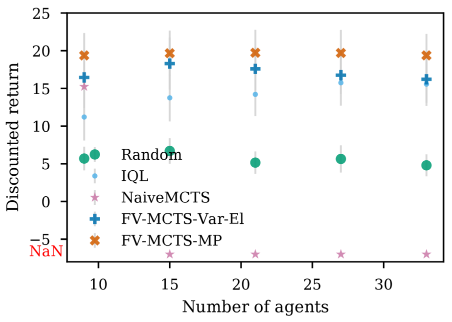


Figure 6: On SysAdmin topologies: Ring (left), Star (middle), and Ring-of-Rings (right), FV-MCTS with MaxPlus performs as well as or slightly better than Var-El, while being much more efficient for larger problems as in Figure 7. NaN indicates that the algorithm ran out of memory.

was faster when there were small number of agents, it ran out of memory as the number of agents increased.

4.2 Multi-Drone Delivery Domain

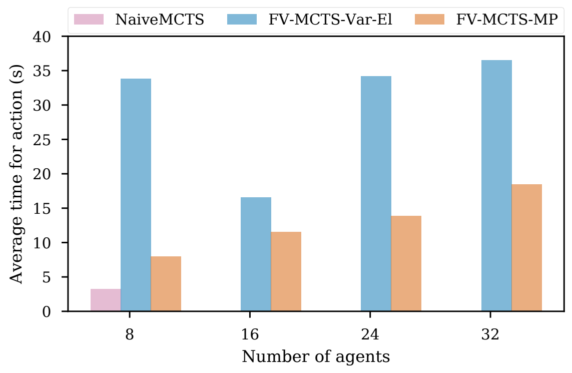
Besides the SysAdmin domain, previous multi-agent decision-making work has also used the Firefighter [1] and Traffic Control [25] domains for benchmarking. Underneath the differing high-level descriptions, however, *the MMDP details of all three domains are very similar*: a small state space and binary action space, the degrees of most nodes in the coordination graph are independent of the total number of agents (except the hub node for Star SysAdmin), and there is no scope in any of them for dynamic CGs.

We introduce and use a truly distinct domain for our second set of experiments. It simulates *a team of delivery drones navigating a shared operation space to reach their assigned goal regions.* We are motivated by recent advances in drone delivery technology, from high-level routing to low-level control [14, 26]; in particular, drones using ground vehicles as temporary modes of transit to save energy and increase effective travel range [11, 12]. Our domain models a key component of such drone-transit coordination: multiple drones assigned to board transit vehicles in close proximity to each other (within the same time window).

Domain details: Figure 4 illustrates our Multi-Drone Delivery domain; for convenience and consistency with MMDP benchmarks we discretize everything, but MCTS could accommodate a continuous state space. Each drone starts in a randomly sampled unique cell in a grid (we use larger grids for more drones in our simulations). There are four circular goal regions, one in each quadrant, that represent a transit vehicle; each goal region has a radius and maximum capacity of drones it can accommodate, since no two drones can occupy the same grid cell (we also vary goal radius with grid size). We allocate the drones to the goal regions at random such that at least two drones target every region.

Each drone has 10 actions in total: one for moving to each of the 8-connected grid neighbors, a NO-OP action for staying in place, and a BOARD action that is only valid when the drone is inside its assigned goal region. The MMDP is episodic and terminates only when all drones have reached their goals and executed BOARD inside them, thus boarding the transit vehicle and receiving a reward of 1000. Drones also receive an intermediate positive reward if they get closer to their assigned goals. Besides drone movement, the other sources of negative reward, i.e., cost, are penalties for two or more drones being too close to each other, attempting to enter the same cell (which makes them both stay in place), and attempting to board in the same goal region at the same time.

Unlike the typical MMDP domains used in prior work, *Multi-Drone Delivery motivates dynamic or state-dependent coordination graphs*; any two drones benefit from coordination only when they are close to each other. Therefore, at the current joint state, we assign a CG edge between any two drones whose mutual distance is



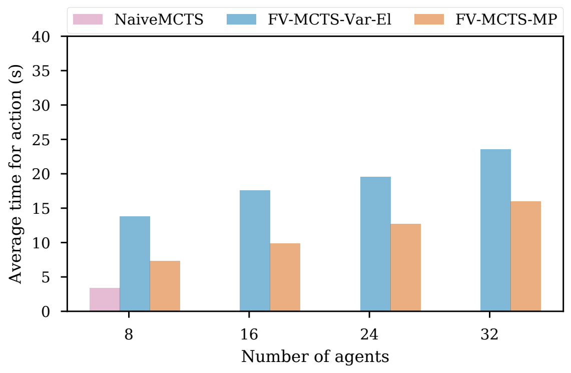


Figure 7: Runtime comparisons (lower is better) for the same tree search hyperparameters on SysAdmin with Ring (left) and Star (right) topologies. The NaiveMCTS baseline ran out of memory with more than 8 **agents.**

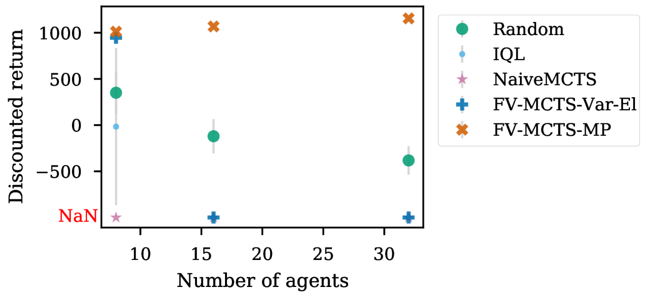


Figure 8: For Multi-Drone Delivery, FV-MCTS-MP vastly outperforms the baselines while effectively using dynamic CGs without any memory issues. NaN indicates that the algorithm ran out of memory.

Agents	XY axis res.	Noise	Expl. const.	Expl. depth	Iter
8	0.20	0.10	5	10	4000
16	0.10	0.05	10	10	8000
32	0.08	0.05	20	10	16000
48	0.05	0.02	30	10	24000

Table 1: Multi-Drone Delivery hyperparameters.

lower than a resolution-dependent threshold (depicted in Figure 4). We also add edges apriori between all drones assigned to the same goal region, as they need to coordinate while boarding.

For all experiments, we set the discount factor γ to 1, the goal reaching reward to 1000.0 units, and the collision penalty to 10.0 units. Table 1 describes the full set of varying problem resolutions and MCTS hyperparameters. The average degree of the dynamic coordination graphs ranged from 2.4 for 8 agents to 11.8 for 48 agents. We averaged all evaluations over 20 runs; error bars indicate standard deviations.

Performance of FV-MCTS-MP against Baselines: As with SysAdmin, we varied the number of drones (agents), discretizing the grid appropriately, and compared against all baselines (except Random) in Figure 8. We observed that FV-MCTS-Var-El and MCTS quickly ran out of memory, which is expected given the large action space per agent. Even on the problems where Var-El runs, its restriction to static CGs leads to slightly worse performance. On the other hand, FV-MCTS-MP can solve tasks even with 48 agents successfully. Moreover, even on the eight agent problem, FV-MCTS-MP is much faster, taking on average approximately 1 s instead of 40 s for FV-MCTS-Var-El for the same tree search hyperparameters. *FV-MCTS-MP scales to MMDP problem sizes that FV-MCTS-Var-El cannot even accommodate.*

5 CONCLUSION

We introduced a scalable online planning algorithm for multi-agent MDPs with dynamic coordination graphs. Our approach, FV-MCTS-MP, uses Max-Plus for action coordination, in contrast to the previously introduced FV-MCTS with Variable Elimination. Over the standard SysAdmin and the custom Multi-Drone Delivery domains, we demonstrated that FV-MCTS-MP performs as well as Var-El on static CGs, outperforms it significantly on dynamic CGs, and is far more computationally efficient (enabling online MMDP planning on previously intractable problems).

In the appendix of the extended version [10], we discuss how our approach can extend to multi-agent POMDPs. However, we still require a domain expert to pre-define the coordination graph for the problem. A predetermined CG can be particularly difficult with dynamic domains where the CG depends on the state. More work is required towards learning the dynamic coordination graph itself via interaction with the model. Similarly, extending ideas from Alpha-Zero [2, 37] would be particularly relevant for distilling the coordinated individual actions for the agents into decentralized policies [34] with FV-MCTS acting as a scalable policy improvement operator [16]. Finally, a theoretical analysis of the exploration strategies and their interaction with MaxPlus' convergence would improve our understanding of the performance.

ACKNOWLEDGMENTS

This research is supported by the Ford Motor Company and the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

REFERENCES

[1] Christopher Amato and Frans A Oliehoek. 2014. Scalable Planning and Learning for Multiagent POMDPs: Extended Version. *arXiv preprint arXiv:1404.1140* (2014). arXiv:1404.1140

[2] Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking Fast and Slow with Deep Learning and Tree Search. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5360–5370.

[3] Dimitri P Bertsekas. 2005. *Dynamic Programming and Optimal Control*. Vol. 1. Athena Scientific Belmont, MA.

[4] Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. 2019. Dec-MCTS: Decentralized Planning for Multi-Robot Active Perception. *International Journal of Robotics Research* 38, 2-3 (March 2019), 316–337.

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* 59, 1 (2017), 65–98.

[6] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. 2019. Deep Coordination Graphs. *arXiv preprint arXiv:1910.00091* (2019).

[7] Craig Boutilier. 1996. Planning, Learning and Coordination in Multi-Agent Decision Processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*. Morgan Kaufmann Publishers Inc., 195–210.

[8] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[9] Guillaume M J-B Chaslot, Mark HM Winands, H Jaap van den Herik, Jos WHM Uiterwijk, and Bruno Bouzy. 2008. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation* 4, 3 (2008), 343–357.

[10] Shushman Choudhury, Jayesh K Gupta, Peter Morales, and Mykel J Kochenderfer. 2021. Scalable Anytime Planning for Multi-Agent MDPs. *arXiv preprint arXiv:2101.04788* (2021).

[11] Shushman Choudhury, Jacob P. Knickerbocker, and Mykel J. Kochenderfer. 2019. Dynamic Real-time Multimodal Routing with Hierarchical Hybrid Planning. In *IEEE Intelligent Vehicles Symposium (IV)*. 2397–2404.

[12] Shushman Choudhury, Kiril Solovey, Mykel J. Kochenderfer, and Marco Pavone. 2020. Efficient Large-Scale Multi-Drone Delivery Using Transit Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[13] Rina Dechter. 1999. Bucket Elimination: A Unifying Framework for Reasoning. *Artificial Intelligence* 113, 1-2 (1999), 41–85. https://doi.org/10.1016/S0004-3702(99)00059-4

[14] Kevin Dorling, Jordan Heinrichs, Geoffrey G Messier, and Sebastian Magierowski. 2016. Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47, 1 (2016), 70–85.

[15] Maxim Egorov, Zachary N Sunberg, Edward Balaban, Tim A Wheeler, Jayesh K Gupta, and Mykel J Kochenderfer. 2017. POMDPs. jl: A Framework for Sequential Decision Making under Uncertainty. *The Journal of Machine Learning Research* 18, 1 (2017), 831–835.

[16] Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. 2020. Monte-Carlo Tree Search as Regularized Policy Optimization. *arXiv preprint arXiv:2007.12509* (2020).

[17] Carlos Guestrin, Daphne Koller, and Ronald Parr. 2002. Multiagent Planning with Factored MDPs. In *Advances in Neural Information Processing Systems*, T G Dietterich, S Becker, and Z Ghahramani (Eds.). MIT Press, 1523–1530.

[18] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. 2003. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research* 19 (Oct. 2003), 399–468. https://doi.org/10.1613/jair.1000

[19] Jayesh K Gupta, Maxim Egorov, and Mykel J. Kochenderfer. 2017. Cooperative Multi-Agent Control using Deep Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Springer, 66–83.

[20] Mykel J. Kochenderfer. 2015. *Decision Making under Uncertainty: Theory and Application*. MIT Press.

[21] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning (ECML)*, Vol. 4212. Springer, 282–293. https://doi.org/10.1007/11871842_29

[22] Jelle R Kok, Eter Jan Hoen, Bram Bakker, and Nikos Vlassis. 2005. Utile Coordination: Learning Interdependencies Among Cooperative Agents. In *EEE Symposium on Computational Intelligence and Games, Colchester, Essex*. 29–36.

[23] Jelle R Kok and Nikos Vlassis. 2004. Sparse Cooperative Q-Learning. In *International Conference on Machine Learning (ICML)*. ACM, 61.

[24] Jelle R. Kok and Nikos A. Vlassis. 2005. Using the Max-Plus Algorithm for Multi-Agent Decision Making in Coordination Graphs. In *Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*. 359–360.

[25] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent Reinforcement Learning for Urban Traffic Control using Coordination Graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 656–671.

[26] Jaihyun Lee. 2017. Optimization of a Modular Drone Delivery System. In *IEEE International Systems Conference*. 1–8. https://doi.org/10.1109/SYSCON.2017.7934790

[27] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J. Kochenderfer. 2021. Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[28] Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. 2012. Independent Reinforcement Learners in Cooperative Markov Games: a Survey Regarding Coordination Problems. *The Knowledge Engineering Review* 27, 1 (2012), 1–31. https://doi.org/10.1017/S0269888912000057

[29] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann, 467–475.

[30] J. A. M. Nijssen and Mark H. M. Winands. 2011. Enhancements for Multi-Player Monte-Carlo Tree Search. In *Computers and Games*.

[31] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. 2015. A Survey of Multi-Agent Formation Control. *Automatica* 53 (2015), 424–440.

[32] Frans A. Oliehoek, Matthijs T. J. Spaan, Shimon Whiteson, and Nikos A. Vlassis. 2008. Exploiting Locality of Interaction in Factored Dec-POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 517–524.

[33] Judea Pearl. 1989. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[34] Thomy Phan, Kyrill Schmid, Lenz Belzner, Thomas Gabor, Sebastian Feld, and Claudia Linnhoff-Popien. 2019. Distributed Policy Iteration for Scalable Approximation of Cooperative Multi-Agent Policies. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2162–2164.

[35] David V. Pynadath and Milind Tambe. 2002. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of Artificial Intelligence Research* 16 (2002), 389–423. https://doi.org/10.1613/jair.1024

[36] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 4295–4304.

[37] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. *Science* 362, 6419 (2018), 1140–1144.

[38] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2164–2172.

[39] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-Decomposition Networks for Cooperative Multi-Agent Learning based on Team Reward. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2085–2087.

[40] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning - An Introduction*. MIT Press.

[41] N Vlassis, R Elhorst, and J R Kok. 2004. Anytime Algorithms for Multiagent Decision Making using Coordination Graphs. In *IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, Vol. 1. 953–957 vol.1.

[42] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. 2004. Tree Consistency and Bounds on the Performance of the Max-Product Algorithm and its Generalizations. *Statistical Computing* 14, 2 (2004), 143–166. https://doi.org/10.1023/B:STCO.0000021412.33763.d5

[43] Chao Yu, Xin Wang, Xin Xu, Minjie Zhang, Hongwei Ge, Jiankang Ren, Liang Sun, Bingcai Chen, and Guozhen Tan. 2020. Distributed Multiagent Coordinated Learning for Autonomous Driving in Highways Based on Dynamic Coordination Graphs. *IEEE Trans. Intell. Transp. Syst.* 21, 2 (2020), 735–748. https://doi.org/10.1109/TITS.2019.2893683

[44] Nicholas Zerbel and Logan Yliniemi. 2019. Multiagent Monte Carlo Tree Search. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2309–2311.

[45] Alexander Aleksandrovich Zykov. 1949. On Some Properties of Linear Complexes. *Matematicheskii Sbornik* 66, 2 (1949), 163–188.