Dec-SGTS: Decentralized Sub-Goal Tree Search for Multi-Agent Coordination

Minglong Li,[*1] Zhongxuan Cai,[*1] Wenjing Yang,[†1] Lixia Wu,[2] Yinghui Xu,[2] Ji Wang[1]

[1] Institute for Quantum Information & State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, China

[2] Artificial Intelligence Department, Zhejiang Cainiao Supply Chain Management Co., Ltd., China

liminglong10@nudt.edu.cn

Abstract

Multi-agent coordination tends to benefit from efficient communication, where cooperation often happens based on exchanging information about what the agents intend to do, i.e. intention sharing. It becomes a key problem to model the intention by some proper abstraction. Currently, it is either too coarse such as final goals or too fined as primitive steps, which is inefficient due to the lack of modularity and semantics. In this paper, we design a novel multi-agent coordination protocol based on subgoal intentions, defined as the probability distribution over feasible subgoal sequences. The subgoal intentions encode macro-action behaviors with modularity so as to facilitate joint decision making at higher abstraction. Built over the proposed protocol, we present Dec-SGTS (Decentralized Sub-Goal Tree Search) to solve decentralized online multi-agent planning hierarchically and efficiently. Each agent runs Dec-SGTS asynchronously by iteratively performing three phases including local sub-goal tree search, local subgoal intention update and global subgoal intention sharing. We conduct the experiments on courier dispatching problem, and the results show that Dec-SGTS achieves much better reward while enjoying a significant reduction of planning time and communication cost compared with Dec-MCTS (Decentralized Monte Carlo Tree Search).

Introduction

The capability to coordinate multiple agents across a wide variety of complex tasks is a critical concern in AI community. It becomes increasingly important to ensure that agents are able to operate not just as individuals but as members of a cohesive team (Smith et al. 2019). Most coordination approaches are centralized. It means that they often follow a principle: centralized offline planning or learning, decentralized online execution of planning result. They usually perform better than the decentralized methods. However, the performance of the centralized ones are strongly limited by the scalability, robustness, fault tolerance etc. Moreover, centralized methods stand in stark contrast with human collaboration: in most context we plan individually, and in parallel with other humans (Czechowski and Oliehoek 2020). To deal with the cases of multi-agent teamwork, the decentralized planning is highly expected.

Under a decentralized framework, each agent should plan cooperatively considering teammates for a global objective. To solve coordination in that context, the agents should know their teammates' current behavior intentions (plannings/decisions). One feasible way is by prediction via teammates' behavior models, e.g. behavior cloning (Czechowski and Oliehoek 2020). However, it relies on tedious offline learning using a lot of training examples. Moreover, once the task changes, the prediction on previous learned models may fail. Another promising way for agents to understand teammates is by communication. It means agents communicate frankly to exchange information about what they intend to do, i.e. intention sharing. Decentralized Monte Carlo Tree Search (Dec-MCTS) is developed upon that methodology, where intentions are modeled based on feasible primitive action sequences with low-level granularity, i.e. step-by-step actions (Best et al. 2019, 2018; Best, Huang, and Fitch 2018; Sukkar et al. 2019; Li et al. 2019). However, the negotiating inefficiency with heavy communication and less semantics severely limit the performance of Dec-MCTS.

In general, human communication always uses protocols which model the intentions with enriched semantics (aka. ontology) when cooperating. This high-level intention abstraction can not only exchange possible future behavior intentions in a highly-compressed style, but also be extremely helpful for local search and optimization. Therefore, in analogy with human coordination, it is valuable to design an abstract high-level communication protocol with enriched semantics for decentralized problem solving.

To model the intentions at proper abstraction, it needs to satisfy two requirements: modularity and semantics. For *modularity*, it means that the abstraction can model the intention of individual agents at a proper granularity. It is neither too fined such as primitive steps nor too coarse such as final goal. It is preferred with modular composition in a hierarchical way. For *semantics*, it means that the abstraction should model the feasible behaviors of individual agents and can facilitate the multi-agent coordination. With the above requirements, we model intentions on *subgoals* or *subtasks* (Sutton, Precup, and Singh 1999; Dietterich 2000). For instance, in Courier Dispatching Problem (CDP) simplified as grid world (Fig. 1), a courier may view a gate, a bridge, a

[*] Equal contribution.

[†] Corresponding author.

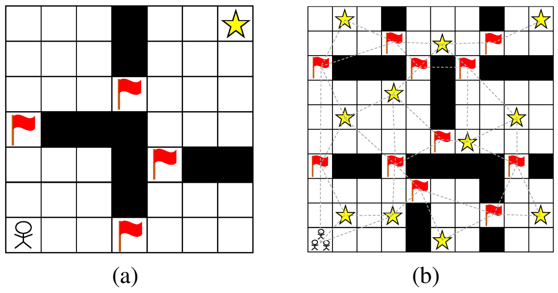


Figure 1: Simplified CDP environments: (a) A courier picks up a package. (b) Three couriers pick up eleven packages.

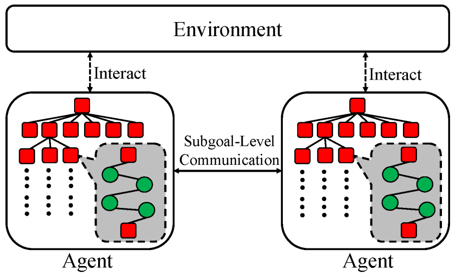


Figure 2: Overview of Dec-SGTS. Multiple agents coordinate by intention sharing with the Subgoal-based Protocol.

crossroad as subgoals (red flags) to help planning a path to the customer (yellow star) (Fig. 1 (a)). These subgoals can be seen as key waypoints in feasible future tracks, i.e., subgoal predicate (Kurzer, Zhou, and Zöllner 2018; Chen et al. 2019a). The subgoals have modularity of state-space division, and thus accelerating planning. Extending to a three-agent case (Fig. 1 (b)), the final goal is to pick up all packages from all customers as soon as possible. The customers (yellow stars) and the door-like positions (red flags) can both be seen as subgoals. For cooperation, each agent should consider teammates and may have several alternative subgoal sequences with different preference. The sequences and the preferences can be combined into subgoal intentions with enriched semantics. Agents can thus plan and coordinate by exchanging their subgoal intentions more efficiently.

In this paper, we design a novel Subgoal-based Protocol (SP) for multi-agent coordination on subgoal intention, defined as the probability distribution over feasible subgoal sequences. The subgoal intentions encode macro-action behaviors with modularity and semantics so as to facilitate joint decision making at higher abstraction. SP includes subgoal predicate, connecting subgoals, evaluating subgoal pairs, encoding and sharing subgoals. Built on SP, we propose Dec-SGTS (Decentralized Sub-Goal Tree Search) by fully reconstructing Dec-MCTS (Best et al. 2019) in a previously unexplored hierarchical manner. Each agent runs Dec-SGTS asynchronously by iteratively performing three phases including local sub-goal tree search, local subgoal intention update and global subgoal intention sharing. As in Fig. 2, inside the agents, they grow sub-goal trees and update their subgoal intentions considering teammates. Outside the agents, they share subgoal intentions by communication.

To perform our idea, the major challenge is *how to refine the Subgoal-based Protocol to grow Sub-Goal Trees of the agents for joint decision?* To solve that challenge, we carefully design the algorithm which: (1) integrates subgoal predicate, subgoal connection, subgoal-pair evaluation with tree expansion in a dynamic and demand-driven style; (2) implements subgoal encoding sharing by employing asynchronous subgoal-based multi-agent interactive model; (3) leverages World Utility and D-UCT in local subgoal tree search to deal with the uncertainty from the received teammates' subgoal intentions; (4) adapts the distributed lagrangian steepest descent framework into local subgoal intention update for distributed optimization of joint decision.

Our main contributions are summarized as follows:

- We design a Subgoal-based Protocol as a general methodology for multi-agent coordination on the subgoal level.
- We propose Dec-SGTS with newly-designed algorithms for sub-goal tree search, subgoal intention update and subgoal intention sharing in the decentralized online setting.
- We compare Dec-SGTS with MCTS (Kocsis and Szepesvári 2006), S-MCTS (Gabor et al. 2019) and Dec-MCTS (Best et al. 2019) on CDP benchmark from real environment. The experimental results show Dec-SGTS achieves an almost 110% higher coordination performance with nearly 45% lower communication cost.

Background

Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a promising approach to online planning (Kocsis and Szepesvári 2006). By using Monte Carlo simulations to sample thousands of possible trajectories quickly, it can achieve good approximations of the values of possible action sequences. MCTS iteratively executes the four steps as follows until a budget runs out.

Selection Starting from root node as current state, the search tree is traversed by selecting nodes until a leaf node with an exploration-exploitation method, i.e. Upper Confidence Trees (UCT) (Kocsis and Szepesvári 2006). Each parent node i selects its child j with the largest $UCT(j)$.

$$UCT(j) = w_j + \sqrt{\frac{2 \ln n_i}{n_j}} \tag{1}$$

where w_j denotes the average value estimation of the state represented by the child j. n_i is the total number that the parent node i has been visited so far. n_j is the total number that the child node j has been visited so far.

Expansion The selected leaf node is expanded by one or more child nodes representing the possible next states. Then, the children will be evaluated by rollout.

Rollout Given an environment model, rollout using a simulation policy, e.g. random sampling, is performed from the leaf to a maximum search depth or a terminal state.

Backpropagation The simulated reward by rollout is used to update the value estimates and visit counts of each node in the path from the leaf node backward to the root node.

Decentralized Monte Carlo Tree Search

Decentralized Monte Carlo Tree Search (Dec-MCTS) (Best et al. 2019) is constructed based on intention sharing, i.e. the probability distribution over feasible primitive action sequences. Formally, we denote $\mathcal{X}^i$ as the set of feasible primitive action sequences $\mathbf{x}^i$ for agent i, where $\mathbf{x}^i :=< a_0^i, a_1^i, a_2^i, ..., a_t^i >, \mathbf{x}^i \in \mathcal{X}^i$. a_t^i is a primitive action of agent i at time step t, which lasts for only one time step.

Definition 1. *The intention of agent i is defined as a probability distribution q^i over feasible action sequences $\mathcal{X}^i$ denoted by $(\mathcal{X}^i, q^i)$. The teammates' intentions except for agent i are denoted by $(\mathcal{X}^{(i)}, q^{(i)})$.*

In Dec-MCTS, each agent runs three phases iteratively: (1) Local Tree Search: it leverages the power of MCTS to select an effective and compact sample space of feasible action sequences $\mathcal{X}^i$; (2) Local Intention Update: a distributed gradient descent method is leveraged to further optimize the probability distribution q^i over $\mathcal{X}^i$ considering current teammates' intentions $(\mathcal{X}^{(i)}, q^{(i)})$; (3) Global Intention Sharing: the updated local intention $(\mathcal{X}^i, q^i)$ is published, and the current teammates' intentions $(\mathcal{X}^{(i)}, q^{(i)})$ are received.

Problem Formulation

Environment

The environment can be generally modeled by an undirected graph (V, E), and $n_v = |V|$, $n_e = |E|$. In grid world, each grid is a vertex, and every two adjacent grids form an edge.

Multi-Agent Markov Decision Process

We formulate the multi-agent coordination problem as deterministic Multi-Agent Markov Decision Process (MMDP) represented by a tuple $< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, h >$, where

- $\mathcal{I}$ is the set of agents and the team size $|\mathcal{I}| = n_i$. Agents can communicate with teammates to share information.
- $\mathcal{S}$ is the set of joint states. It includes positions of the agents and other domain-specific states of environment.
- $\mathcal{A}$ is the set of joint actions. All agents select adjacent vertexes to visit as a joint action in a single time step.
- T is the probabilities of transition between states for particular choices of actions: $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. We assume that the transition is deterministic and only determined by the destinations of the agent joint movements.
- R is the immediate joint reward function $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. An agent gets a negative reward when moving for one step, and gets a positive reward when finishing some subtask.
- h is the horizon of the problem. Each agent can move for at most h time steps during the whole task.

Planning Objective

Recall $\mathbf{x}^i :=< a_0^i, a_1^i, a_2^i, ...a_h^i >$ is an action sequence of agent i during horizon h. We denote $\mathbf{x}$ as the set of action sequences selected by all agents $\mathbf{x} := \{\mathbf{x}^1, ..., \mathbf{x}^{n_i}\}$. The team goal is to choose the best joint action sequences $\mathbf{x}$ to maximize an global objective function $o(\mathbf{x})$ by summing up the joint reward R accumulated over n_i agents and h time steps.

Subgoal-based Protocol for Coordination

Subgoal Predicate

For subgoal-based coordination, agents should first have consensus on how to choose subgoal state, i.e. subgoal predicate. We use lowercase s to denote the state of a single agent, i.e. the factored state of $\mathcal{S}$ in MMDP. We denote $g(s)$ as subgoal predicate. When the agent is at a subgoal state, $g(s) = 1$, otherwise, $g(s) = 0$. In general, subgoals have two types. One is separated directly from final goal with some splited reward, as the packages in Fig. 1 (yellow stars). The other is *auxiliary subgoals/subtasks* only for accelerating planning without any reward, which follows the terminology in Reinforcement Learning (Sutton and Barto 2018; Jaderberg et al. 2016), as the doors in Fig. 1 (red flags). The domain-specific g can be designed by learning-based offline training (Jaderberg et al. 2016; Eysenbach, Salakhutdinov, and Levine 2019) or rule-based online reasoning (Kurzer, Zhou, and Zöllner 2018; Chen et al. 2019a; Gabor et al. 2019). We design rule-based g for fully-online planning.

Subgoal Connection

Given subgoal predicate g, agent can determine whether its successive state is a neighbor subgoal state. As the coordination goes on, agents can gradually build a subgoal-state graph cooperatively (grey dotted lines in Fig. 1 (b)). It is highly-compressed compared with primary graph (V, E).

Subgoal-to-Subgoal Evaluation and Policy

For an agent i, there is a set of feasible subgoal sequences $\mathbf{g}^i \in \mathcal{G}^i$, where $\mathbf{g}^i :=< s_0^i, s_1^i, ... >$ and $\forall \tau, g(s_\tau^i) = 1$, where the subscript τ of s_τ^i is a macro time step lasting for more than or equal to one primitive step. We denote $\mathbf{g}$ as the set of subgoal state sequences selected by all agents $\mathbf{g} := \{\mathbf{g}^1, \mathbf{g}^2, ..., \mathbf{g}^{n_i}\}$. An agent i should know the subgoal-pair action policy for moving denoted by $\mathbf{x}(s_\tau^i, s_{\tau+1}^i) :=< a_t^i, a_{t+1}^i, ... >$ and the subgoal-pair distance evaluation for planning denoted by $|\mathbf{x}(s_\tau^i, s_{\tau+1}^i)|$. To get them online, instead of accurate but time-consuming methods, an approximate but elaborately-designed fast method should be better.

Encoding and Sharing Subgoals

Subgoals to share can be abstracted as subgoal intentions.

Definition 2. *We define the subgoal intention of agent i as a probability distribution q^i over feasible subgoal state sequences $\mathcal{G}^i$ denoted by $(\mathcal{G}^i, q^i)$. The subgoal intentions of teammates except for agent i are denoted as $(\mathcal{G}^{(i)}, q^{(i)})$.*

Besides, when an agent shares its current subgoal intention, it should also share the subgoal-pair evaluations in that intention, so that teammates can evaluate when receiving it.

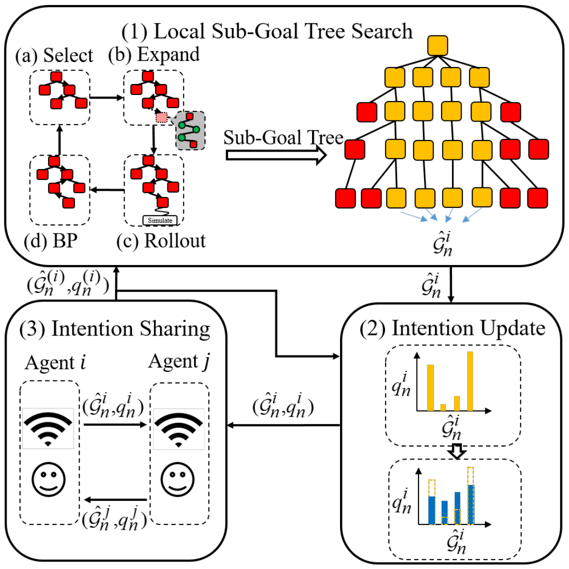


Figure 3: Dec-SGTS runs three phases including local SGTS, intention update and intention sharing iteratively.

Algorithm 1 Dec-SGTS for agent i

```
Input: $o, g, (V, E)$                ▷ Functions and the environment
Parameter: $b_0, b_1, b_2$           ▷ Iteration budgets for the loops
Output: $\mathbf{x}^i$               ▷ Planned action sequences for agent $i$
1:  $SGT$←InitialiseSubGoalTree();
2:  for $n \leftarrow 1$ to $b_0$ do
3:      ▷ Sparse Representation of Current Tree
4:      $\hat{\mathcal{G}}_n^i$←SelectSetOfSubgoalSequences($SGT$);
5:      for $m \leftarrow 1$ to $b_1$ do
6:          ▷ Phase 1: Local sub-goal tree search
7:          $SGT \leftarrow$ SGTS($SGT, (\hat{\mathcal{G}}_n^{(i)}, q_n^{(i)}), o, g, b_2$);
8:          ▷ Phase 2: Local subgoal intention update
9:          $(\hat{\mathcal{G}}_n^i, q_n^i)$←IntentionUpdate($\hat{\mathcal{G}}_n^i, q_n^i), \hat{\mathcal{G}}_n^{(i)}, q_n^{(i)}$);
10:         ▷ Phase 3: Global subgoal intention sharing
11:         $(\hat{\mathcal{G}}_n^{(i)}, q_n^{(i)})$←IntentionSharing($(\hat{\mathcal{G}}_n^i, q_n^i)$);
12:     end for
13: end for
14: $\mathbf{g}^i \leftarrow \arg\max_{\mathbf{g}^i \in \hat{\mathcal{G}}_n^i} [q_n^i(\mathbf{g}^i)]$;   ▷ Best subgoal sequence
15: return $\mathbf{x}^i \leftarrow$ GetPrimitiveActionSequence($\mathbf{g}^i$,$SGT$));
```

Refine Subgoal-based Protocol in Dec-SGTS

Overview of Dec-SGTS

Built on SP, we propose Dec-SGTS by fully reconstructing Dec-MCTS (Best et al. 2019). Dec-SGTS runs on each agent by iteratively executing three phases including local sub-goal tree search, local subgoal intention update and global subgoal intention sharing. As shown in Fig. 3 and Alg. 1, agent i initialises a sub-goal tree denoted by SGT, where each tree node represents a subgoal state of agent i (Line 1). The feasible subgoal sequences $\hat{\mathcal{G}}_n^i$ of agent i are selected from SGT using sparse representation (Line 4). In phase 1 (Line 7), agent i grows SGT for b_2 iterations considering teammates' subgoal intentions $(\hat{\mathcal{G}}_n^{(i)}, q_n^{(i)})$, called Sub-Goal Tree Search (SGTS). In phase 2 (Line 9), the probability distribution q_n^i over $\hat{\mathcal{G}}_n^i$ is updated using a distributed optimization method considering $(\hat{\mathcal{G}}_n^{(i)}, q_n^{(i)})$. In phase 3 (Line 11), agent i publishes its current intention $(\hat{\mathcal{G}}_n^i, q_n^i)$ and receives teammates' intentions $(\hat{\mathcal{G}}_n^{(i)}, q_n^{(i)})$. Finally, best subgoal sequences $\mathbf{g}^i$ can be found with max $q_n^i(\mathbf{g}^i)$ and decoded to primitive actions $\mathbf{x}^i$ by inquiring SGT (Lines 14&15).

Phase 1: Local Sub-Goal Tree Search

Sparse Representation: Periodically Selecting a Set of Subgoal Sequences from Current Tree $\mathcal{G}_n^i$ has an exponential cardinality. We leverage a sparse representation by periodically selecting the sample space $\hat{\mathcal{G}}_n^i \in \mathcal{G}_n^i$ as the most promising subgoal sequences found by SGTS so far (Line 4, Alg. 1). We select the sequences in $\hat{\mathcal{G}}_n^i$ that have higher reward than the others in $\mathcal{G}_n^i$. The size of $\hat{\mathcal{G}}_n^i$ is set empirically as a tunable parameter. (In initial iterations, the selected sequences can not reach the terminal state of horizon h, so we extend them using a default policy, e.g. random. As Dec-SGTS goes on, the selected subgoal sequences can reach h.)

Local Utility Function: Simulating Reward for Guiding the Growth of Local Sub-Goal Tree Follow Dec-MCTS (Best et al. 2019), rather than optimising directly for a global object function o, each agent i instead optimises with respect to a local utility function f^i to grow tree:

$$f^i(\mathbf{g}) := o(\mathbf{g}^i \cup \mathbf{g}^{(i)}) - o(\mathbf{g}_\emptyset^i \cup \mathbf{g}^{(i)}) \qquad (2)$$

where $\mathbf{g}_\emptyset^i$ is a default 'no reward' subgoal sequence for agent i and would typically be an empty sequence. Although the agents counld use the global utility o directly, optimising with respect to f^i instead results in faster convergence since f^i is less affected by the unknown plans of teammates, aka. World Utility (Rahmattalabi et al. 2016).

Selection: Discounted UCT in Sub-Goal Tree We leverage an exploration-exploitation method to traverse Sub-Goal Tree called Discounted UCT (D-UCT) (Best et al. 2019). Agent i alternately plans locally (local SGTS and intention update) and communicates cooperatively (intention sharing). We use q_n to denote the joint distribution of the current plannings of all agents, that is, the combination of q_n^i (updated locally) and $q_n^{(i)}$ (received globally). Between two communication round, q_n may change abruptly. It may oscillate the convergence of tree growth. D-UCT leverages a discounted factor γ to discount the past, and $0 < \gamma < 1$. It is based on the principle that the most recent rollouts are more relevant since they are obtained by sampling the most recent distributions. Formally, the selection rule is the same as $UCT(j)$ (Eqn. 1), but the update of w_j, n_i, n_j in this function is different. In detail, when SGTS finishes a new rollout to get a simulated reward r calculated by World Utility (Eqn. 2), it backpropagates to update the value estimate w_j and the visit count n_j of each node j in the path as follows, $w_j = (w_j n_j + r/\gamma^{n_c})/(n_j + 1/\gamma^{n_c})$ and $n_j = n_j + 1/\gamma^{n_c}$, where n_c is the total communication rounds currently, i.e. $n_c = nm$ (n and m are presented in Lines 2&5 in Alg. 1).

Algorithm 2 Expansion with Subgoal States

```
Input: $LeafNode, g$
Parameter: $\sigma, h$
Output: $NewChild$
 1: $c \leftarrow 0$                                ▷ count for action coverage
 2: repeat
 3:     $s_t \leftarrow LeafNode$.GetState();
 4:     $\mathbf{x}_t \leftarrow <>$
 5:     ▷ Sample until another subgoal state or horizon
 6:     repeat
 7:         $a_t \sim \mathcal{A}_i$                        ▷ sample an action
 8:         $\mathbf{x}_t \leftarrow \mathbf{x}_t +\!\!+ < a_t >$      ▷ add this sampled action
 9:         $s_t \leftarrow$ StateTransition$(s_t, a_t)$
10:     until $(g(s_t) = 1)$
11:     $Flag \leftarrow$False        ▷ whether $s_t$ existing in children
12:     for $Child$ in $LeafNode$.GetChildrenList() do
13:         ▷ Successive subgoal state is rediscovered
14:         if $s_t$=$Child$.GetState() then
15:             $Flag \leftarrow$True
16:             $c \leftarrow c + 1$
17:             ▷ Distance from leaf to current child
18:             $d \leftarrow Child$.GetSubgoalPairDistance()
19:             ▷ Update leaf-child subgoal-pair evaluation
20:             if $|\mathbf{x}_t| \le d$ then
21:                 ▷ Store the updated in current child
22:                 $Child$.UpdateSubgoalPairDistance($|\mathbf{x}_t|$)
23:                 $Child$.UpdateSubgoalPairPolicy($\mathbf{x}_t$)
24:             end if
25:         end if
26:     end for
27:     ▷ Successive subgoal state is newly discovered
28:     if $Flag$=False then
29:         $NewChild \leftarrow LeafNode$.AddChild($s_t$)
30:         $NewChild$.UpdateSubgoalPairDistance($|\mathbf{x}_t|$)
31:         $NewChild$.UpdateSubgoalPairPolicy($\mathbf{x}_t$)
32:     end if
33: until $c > \sigma$
34: $LeafNode$.IsFullyExpanded←True;
35: return $NewChild$;
```

Proposition 1. *Although the joint probability distribution, i.e. q_n, is changing (and converging), D-UCT maintains an exploration–exploitation trade-off for child selection and achieves a polynomial convergence rate*[1].

Expansion: Expansion with Subgoal States Algorithm 2 shows the pseudocode of expansion in an online and demand-driven style. This is a key of Dec-SGTS, we merge three parts of SP in tree expansion inlcuding: (1) choosing and connecting subgoals; (2) subgoal-to-subgoal distance; (3) subgoal-to-subgoal policy as primitive action sequence.

Overall, the expansion is to find several neighbor successive subgoal states through approximate sampling until confident, and add them as children to the leaf node. Along with that, the subgoal-to-subgoal distance and policy are stored and updated in added children. We use a parameter σ to denote the amount of sampling trials for discovering successive subgoal states (Lines 3-33). First, we get the current subgoal state s_t from leaf node (Line 3). We simulate agent randomly walking from current subgoal state to any successive subgoal state s_t decided by g in horizon h (Lines 6-10). Then, we check if the successive subgoal state s_t is already discovered and added in the tree as an existing child (Lines 11-14). If so, we update the existing child with the newly-discovered subgoal-pair distance and policy if they are better than the old. Each node stores the subgoal-pair distance and action sequence from its parent to itself (Lines 15-24). At the same time, action coverage count c increases (Line 16). If not, which means the successive subgoal state is newly found, we add it as a new child of leaf node, and store current subgoal-pair distance and policy in the new child (Lines 28-32). Finally, if the action coverage count c reaches the threshold σ, the loop ends and the leaf node is set as fully expanded. The new child is returned (Lines 33-35).

Proposition 2. *Given a proper σ, Algorithm 2 can find the best subgoal-pair distances and policies asymptotically when expanding the tree with successive subgoal states*[2].

Rollout: Subgoal-based Interactive Simulation Model Considering Teammates' Intentions We get $\mathbf{g}^{(i)}$ by sampling from shared teammates' intentions $(\mathcal{G}_n^{(i)}, q_n^{(i)})$. We get $\mathbf{g}^i$ from current traversed tree path. A rollout heuristics can extend $\mathbf{g}^i$ to the terminal state, e.g. random. So we can simulate multi-agent interaction locally to get a reward calculated by Eqn. 2 considering current teammates' plannings.

Backpropagation: Backpropagate the new rollout reward and visit count multiplied by $1/\gamma^{n_c}$ as mentioned in D-UCT.

Phase 2: Local Subgoal Intention Update

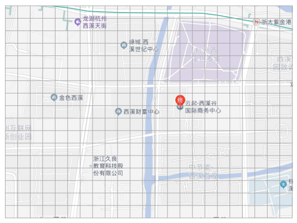
The intention update rule is from the theory of PC (Probability Collective) (Wolpert, Bieniawski, and Rajnarayan 2013). We leverage a distributed optimization method based on Lagrangian Steepest Descent proposed in (Wolpert and Bieniawski 2004) to modify the intention $(\hat{\mathcal{G}}_n^i, q_n^i)$. When a new $\hat{\mathcal{G}}_n^i$ is selected by sparse representation (Line 4, Alg. 1), the probability distribution q_n^i over it is initialised as uniform distribution. Each $q_n^i(\mathbf{g}^i)$ of q_n^i is updated once in a subgoal intention update phase (Line 9, Alg. 1) by the function:

$$q_n^i(\mathbf{g}^i) = q_n^i(\mathbf{g}^i) - \eta q_n^i(\mathbf{g}^i)[\frac{E_{q_n}[f^i] - E_{q_n}[f^i|\mathbf{g}^i]}{\beta} + H(q_n^i) + \ln(q_n^i(\mathbf{g}^i))] \tag{3}$$

where q_n is the joint distribution for all agents, E_{q_n} is the expectation of joint reward with respect to q_n. Considering the overhead, we use sampling methods to approximate those two expectations. H is the entropy. β gradually decreases. η is a designated step size. The above process can be seen as a bridge between game theory and information theory. Intuitively, it increases the probability that agent i selects $\mathbf{g}^i$ if that leads to an improved local utility, while also

[1]For formalization and proof, see supplementary material https://github.com/HPCL-micros/dec-sgts.

[2]For formalization and proof, see supplementary material https://github.com/HPCL-micros/dec-sgts.


(a)

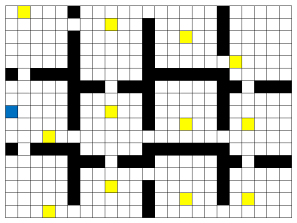

(b)

Figure 4: Instances of urban region of CDP. (a) a screenshot of CDP real environment from the application Amap. (b) an instance of CDP real environment simplified as grid world.

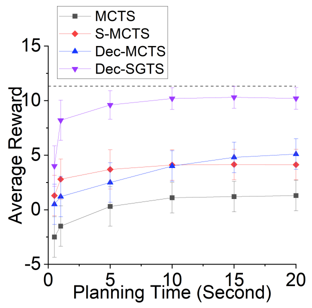


Figure 5: Average rewards for different planning time.

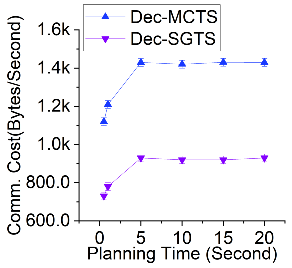


Figure 6: Average communication cost in bytes/second.

ensuring the entropy does not decrease too rapidly. (In fact, the sequence set $\hat{\mathcal{G}}_n^i$ as a sparse representation (Line 4, Alg. 1) is also gradually optimized as Dec-SGTS goes on).

Proposition 3. *Given an appropriate subset $\hat{\mathcal{G}}_n^i \in \mathcal{G}_n^i$, the above algorithm by iteratively implementing the subgoal intention update asymptotically converges to the joint distribution q_n that optimizes the joint subgoal sequences*[3].

Phase 3: Global Subgoal Intention Sharing

Agent i shares its current intention $(\hat{\mathcal{G}}_n^i, q_n^i)$ by communicating it with teammates. It also shares the subgoal-pair distances in that intention $(\hat{\mathcal{G}}_n^i, q_n^i)$. We design a message type composed of four fields including agent ID, time step, current intention and subgoal-pair distances in that intention. If no new messages are received from a teammate, then agent i continues to plan based on the most recent distribution of the teammate. If agent i is yet to receive any messages from a teammate, it may assume a default policy such as a random policy. Hence, Dec-SGTS is an asynchronous method.

Final Policy Generation

Agent i can select the best subgoal sequences $\mathbf{g}^i$ with max probability $q_n^i(\mathbf{g}^i)$ in current intention $(\hat{\mathcal{G}}_n^i, q_n^i)$. Since in tree expansion, each newly-added child node stores and updates the best primitive action sequence between its parent and itself, we can use current tree SGT to decode $\mathbf{g}^i$ to best primitive action sequence $\mathbf{x}^i$ for moving (Lines 14&15, Alg. 1).

Experiments & Results

Experimental Setup

Environment We implement the instances of Dec-SGTS in CDP (Figure 4). Figure 4 (a) is an example of a real environment map, we see the crossings, bridges and gates as auxiliary subgoals. Figure 4 (b) is an instance of CDP grid world with three couriers. The couriers start from a depot (blue grid) aiming to pick up all the packages (yellow grids) as soon as possible in horizon h. Grid world is one of the most studied domains in AI including courier dispatching (Chen et al. 2019b), warehouse commissioning (Claes et al. 2017) and ride hailing (Jin et al. 2019) etc.

[3]For formalization and proof, see supplementary material https://github.com/HPCL-micros/dec-sgts.

Settings Dec-SGTS is implemented with a platform of 12 cores, 3.7 GHz and 16 GB Memory. We ran each experiment for 50 times and present the average rewards with standard deviations. Agent moves for one grid with reward -0.01 and picks up a package with reward 1.0. For each experiment, we use different settings and parameters[4]. We compare Dec-SGTS with MCTS (Kocsis and Szepesvári 2006), S-MCTS (Gabor et al. 2019) and Dec-MCTS (Best et al. 2019).

Subgoal Predicate Heuristics Subgoal predicate function g is domain-specific. In an urban region of CDP, if an agent moves to a customer, it often passes a bridge, a zebra crossing, a community gate or a building door etc., which can be seen as a subgoal state denoted by $g(s) = 1$. Those grids can be see as 'door' positions with obstacles on opposing sides adjacent to it, where the agent can only move with two opposite legal actions. The subgoal predicate function g can be formulated as: if $|\mathcal{A}(s)| = 2$ or there is a package at state s, $g(s) = 1$, otherwise $g(s) = 0$, where $|\mathcal{A}(s)|$ are the number of legal following actions for an agent at state s.

Results and Discussion

Performance-Computation Tradeoff Figure 5 presents average rewards for different planning time in scenario of Fig. 4 (b) with 3 agents. Dec-SGTS significantly outperforms the others by more than 110%, and approximates optimality (dotted line) in less than 10 seconds. There is an upward trend of S-MCTS, however, the reward increases very slowly. Dec-MCTS increases slowly either. It does not converge in 20 seconds. Given only 20 seconds, MCTS can hardly make the agents find any valuable planning result.

Performance-Communication Tradeoff In Fig. 6, we record the communication cost represented as bandwith in bytes/second in the scenario of Fig. 4 (b) with 3 agents. The result show that Dec-MCTS suffers from heavier communication which is nearly twice as much as that of Dec-SGTS.

Rewards of Different Subgoal Predicate We test four subgoal predicate including exact definition, $|\mathcal{A}(s)| = 2$, $|\mathcal{A}(s)| \leq 2$, and $|\mathcal{A}(s)| \leq 3$. Customer locations are subgoals in default. Given planning time of 15 seconds. Table 1 presents the average rewards with regard to different kinds

[4]For detail parameter settings, see supplementary material https://github.com/HPCL-micros/dec-sgts.

Subgoal Predicate	Rewards
Exact Definition	10.65
$\lvert\mathcal{A}(s_t)\rvert = 2$	10.65
$\lvert\mathcal{A}(s_t)\rvert \leq 2$	7.65
$\lvert\mathcal{A}(s_t)\rvert \leq 3$	1.81

Table 1: The average rewards with regard to different kinds of subgoal predicate heuristics of Dec-SGTS.

σ	Rewards
40	10.65
30	7.89
20	1.2
10	-

Table 2: The average rewards for different action coverage thresholds σ of Dec-SGTS.

of subgoal predicate in the scenario of Fig. 4 (b). We observe that Dec-SGTS can find a near-optimal result using exact definition and $|\mathcal{A}(s_t)| = 2$. Coarser subgoal predicate brings much worse results for $|\mathcal{A}(s)| \leq 2$ and $|\mathcal{A}(s)| \leq 3$.

Action Coverage Threshold to Decide Neighbor We use the action coverage threshold σ to expand tree nodes with subgoal states. Table 2 presents the average rewards with regard to different σ in the scenario of Fig. 4 (b). Given a planning time of 15 seconds, when $\sigma = 40$, Dec-SGTS easily find a near-optimal result. As the σ goes down, the average rewards decrease fast. When $\sigma = 30$ and $\sigma = 20$, the rewards are 7.89 and 1.20 respectively. When $\sigma = 10$, the connected graph among the subgoal states can not be built, let alone the proper subgoal-pair evaluations, hence there is no meaningful result.

Discussion In Dec-SGTS, the huge search space for a joint policy is segmented with subgoals in a hierarchical way. Hence, the state space is modularized. Shared messages are also highly compressed by a macro protocol, and the message semantics is richer. Therefore, with the aid of these advatages, Dec-SGTS achieves much higher performance with less communication cost. Dec-MCTS is hindered by the drawback of primitive-level search and negotiation. S-MCTS has the natural disadvantage of centralized planning when the problem scales. MCTS is a centralized method on primitive level and is hindered by the above two drawbacks.

Threats to validity compromising our confidence are concerned with the design of the subgoal predicate, the size of the team and environment, the density degree of subgoals.

Related Work

In addition to the papers we previously discussed, first, we will mention other related decentralized MCTS methods for mulit-agent coordination, but different to Dec-SGTS. At a high level, our proposed Dec-SGTS is similar to Dec-MCTS (Best et al. 2019), but we differ in how we expand local tree dynamically with subgoal states using a subgoal predicate, integrate subgoal-pair connection and evaluation with tree expansion, share subgoal-level information with enriched semantics, and we propose solving the multi-agent coordination with an anytime, hierarchical and principled manner. Kurzer et al. propose a coordination method on macro actions in the domain of automated vehicles (Kurzer, Zhou, and Zöllner 2018). In (Claes et al. 2017), a decentralized MCTS method is implemented on warehouse commissioning task with a high efficiency. But the above two methods base on the assumption that each agent has a behavior model of each other in advance. It is unsuitable for most scenarios.

Second, we will mention some subgoal-based methods that we draw inspiration from. Temporal Abstraction summarizes temporal sequences of primitive actions into macro-actions (Sutton, Precup, and Singh 1999; Amato, Konidaris, and Kaelbling 2014; Amato et al. 2019), which are based on the Options framework (Sutton, Precup, and Singh 1999). Dietterich et al. propose MAXQ Value Function Decomposition by defining subtasks or subgoals (Dietterich 2000). Recently, Gabor et al. proposed Subgoal-based MCTS (S-MCTS) (Gabor et al. 2019). However, the natural drawbacks of centralized methods hinder their performance.

Third, it is worthy to mention Ad Hoc Teamwork (Stone et al. 2010; Chen et al. 2019a), where an agent engages in collaborative tasks without relying on communication or pre-defined strategy. However, without communication, it may rely on much offline training instead for understanding teammates, e.g. Convolutional Neural Network for detecting behavior switching (Ravula, Alkoby, and Stone 2019).

Conclusions and Future Work

In this paper, we design a novel Subgoal-based Protocol (SP) for communication-enabled coordination on a lifted abstraction with modularity and enriched semantics. Built over SP, we propose Dec-SGTS for multi-agent coordination on subgoal level in a much more efficient way. The experimental results validate the efficiency and efficacy of Dec-SGTS. We believe rational communication with a high-efficiency protocol can improve the performance of multi-agent systems. Our work opens exciting directions for future research, including: (1) can our approach be extended to stochastic environments? (2) can the scalability be further improved by coalition-based methods? (3) can Dec-SGTS be further strengthened by merging it with learning?

Finally, except for courier dispatching problem we studied in this paper, we believe that our idea will be applicable for robotics, warehouse commissioning, ride hailing and other domains where subgoal-based multi-agent coordination is important.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (No. 2017YFB1301104, No. 2017YFB1001900), the Youth Talent Lifting Project (No. 17-JCJQ-QT-047), the National Natural Science Foundation of China (No. 91948303, 91648204, 61803375, 62032024 and 12002380), and the National Science and Technology Major Project.

References

Amato, C.; Konidaris, G.; Kaelbling, L. P.; and How, J. P. 2019. Modeling and planning with macro-actions in decentralized POMDPs. *Journal of Artificial Intelligence Research (JAIR)* 64: 817–859.

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized POMDPs. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1273–1280.

Best, G.; Cliff, O. M.; Patten, T.; Mettu, R. R.; and Fitch, R. 2019. Dec-MCTS: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research (IJRR)* 38(2-3): 316–337.

Best, G.; Forrai, M.; Mettu, R. R.; and Fitch, R. 2018. Planning-aware communication for decentralised multi-robot coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1050–1057. IEEE.

Best, G.; Huang, S.; and Fitch, R. 2018. Decentralised mission monitoring with spatiotemporal optimal stopping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4810–4817. IEEE.

Chen, S.; Andrejczuk, E.; Irissappane, A. A.; and Zhang, J. 2019a. ATSIS: achieving the ad hoc teamwork by subtask inference and selection. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 172–179. AAAI Press.

Chen, Y.; Qian, Y.; Yao, Y.; Wu, Z.; Li, R.; Zhou, Y.; Hu, H.; and Xu, Y. 2019b. Can Sophisticated Dispatching Strategy Acquired by Reinforcement Learning? In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1395–1403.

Claes, D.; Oliehoek, F.; Baier, H.; and Tuyls, K. 2017. Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 492–500.

Czechowski, A.; and Oliehoek, F. 2020. Decentralized MCTS via Learned Teammate Models. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*. AAAI Press.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research (JAIR)* 13: 227–303.

Eysenbach, B.; Salakhutdinov, R.; and Levine, S. 2019. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Gabor, T.; Peter, J.; Phan, T.; Meyer, C.; and Linnhoff-Popien, C. 2019. Subgoal-based temporal abstraction in Monte-Carlo tree search. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 5562–5568. AAAI Press.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* .

Jin, J.; Zhou, M.; Zhang, W.; Li, M.; Guo, Z.; Qin, Z.; Jiao, Y.; Tang, X.; Wang, C.; Wang, J.; et al. 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*, 282–293. Springer.

Kurzer, K.; Zhou, C.; and Zöllner, J. M. 2018. Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search. In *Proceedings of the Intelligent Vehicles Symposium (IV)*, 529–536. IEEE.

Li, M.; Yang, W.; Cai, Z.; Yang, S.; and Wang, J. 2019. Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 450–456. AAAI Press.

Rahmattalabi, A.; Chung, J. J.; Colby, M.; and Tumer, K. 2016. D++: Structural credit assignment in tightly coupled multiagent domains. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4424–4429. IEEE.

Ravula, M.; Alkoby, S.; and Stone, P. 2019. Ad hoc teamwork with behavior switching agents. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 550–556. AAAI Press.

Smith, A. J.; Best, G.; Yu, J.; and Hollinger, G. A. 2019. Real-time distributed non-myopic task selection for heterogeneous robotic teams. *Autonomous Robots* 43(3): 789–811.

Stone, P.; Kaminka, G. A.; Kraus, S.; and Rosenschein, J. S. 2010. Ad hoc autonomous agent teams: collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Sukkar, F.; Best, G.; Yoo, C.; and Fitch, R. 2019. Multi-robot region-of-interest reconstruction with Dec-MCTS. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 9101–9107. IEEE.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence (AI)* 112(1-2): 181–211.

Wolpert, D. H.; and Bieniawski, S. 2004. Distributed control by lagrangian steepest descent. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 1562–1567. IEEE.

Wolpert, D. H.; Bieniawski, S. R.; and Rajnarayan, D. G. 2013. Probability collectives in optimization. In *Handbook of Statistics*, volume 31, 61–99. Elsevier.