Supplementary Material for "Time Minimization and Online Synchronization for Multi-agent Systems under Collaborative Temporal Tasks"

Zesen Liu, Meng Guo and Zhongkui Li

TABLE I
DESCRIPTION OF RELATED REGIONS AND AGENT ACTIONS.

Proposition	Description	Duration [s]
$\mathtt{p}_1, \cdots, \mathtt{p}_{34}$	34 PV panels.	\
$\mathtt{b}$	Base stations for all agents to park and charge.	\
$\mathtt{t}_1, \cdots, \mathtt{t}_7$	7 transformers.	\
$\mathtt{temp}_{\mathtt{p}_i,\mathtt{t}_i}$	Measure temperature of panel $\mathtt{p}_i$ and transformer $\mathtt{t}_i$. Requires one V_f.	10
$\mathtt{sweep}_{\mathtt{p}_i}$	Sweep debris around any panel $\mathtt{p}_i$. Requires one V_s.	190
$\mathtt{mow}_{\mathtt{p}_i,\mathtt{t}_i}$	Mow the grass under panel $\mathtt{p}_i$ or transformer $\mathtt{t}_i$. Requires one V_s.	190
$\mathtt{fix}_{\mathtt{t}_i}$	Fix malfunctional transformer $\mathtt{t}_i$. Requires one V_l and one V_s	72
$\mathtt{repair}_{\mathtt{p}_i}$	Repair broken panel $\mathtt{p}_i$. Requires one V_s to repair and two V_f to guide.	576
$\mathtt{wash}_{\mathtt{p}_i}$	Wash the dirt off panel $\mathtt{p}_i$. Requires one V_l to wash and one V_f to monitor the progress.	565
$\mathtt{scan}_{\mathtt{p}_i,\mathtt{t}_i}$	Build 3D models of panel $\mathtt{p}_i$ or transformer $\mathtt{t}_i$ for inspection. Requires three V_f.	95

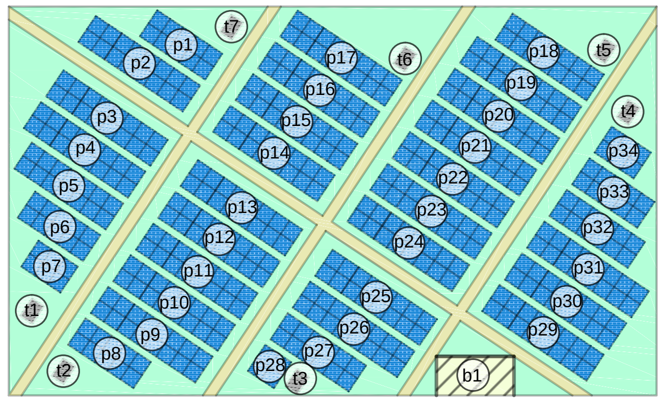

Fig. 1. The workspace layout of the simulated PV farm.

I. MILP FORMULATION

Give a poset $P = (\Omega, \preceq_\varphi, \neq_\varphi)$, where Ω is a sequence of subtasks, and $\preceq_\varphi$, $\neq_\varphi$ are the partial relations to describe the relative ordering between the subtasks. Furthermore, $\omega_1 \preceq_\varphi \omega_2$ means subtask ω_2 should be started after ω_1 is started, and $\omega_1 \neq_\varphi \omega_2$ means that the execution of ω_1, ω_2 should not have intersection. A team of agents is deployed to execute these subtasks under the constraints of $\preceq_\varphi, \neq_\varphi$ in the given workspace, e.g., as shown in Fig. 1. Additionally, the agents are heterogeneous with three different types V_f, V_l, V_s, e.g., different velocities and functionalities. For any subtask $\omega_j \in \Omega$, it needs a particular combination of different collaborators as specified in Table I. The objective function is to minimize the maximum execution time of all subtasks in Ω, i.e., the makespan of the complete plan. In particular, given the definition of variables in Table II, the Mixed Integer Linear Program (MILP) for solving this problem is formulated as follows. Similar formulation can be found in [1], [2], but with different objective function and different way of modeling the relative constraints. The objective function is given by:

$$\min_{r_{i,j,k,l}, t_j, q_{j_1,j_2}} \max(t_j + p_j), \tag{1}$$

The authors are with the State Key Laboratory for Turbulence and Complex Systems, Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China.

Variable	Variable definition	
Name	definition	range
P	partial order set	
m	number of agents	$\mathcal{N}$
n	number of tasks	$\mathcal{N}$
i	agent i	$i \leqslant n$
j	task j	$j \leqslant m$
k	order of tasks	$j \leqslant m$
o	number of serves type agent can provide	$\mathcal{N}$
l	order of tasks	$j \leqslant o$
r	the ID of tasks set in $\neq$	$\mathcal{N}$
Ω	set of subtasks	
S^r	a set of tasks belong to $\neq$	
$q^r_{j_1,j_2}$	task j_1 execute in front of j_2 or not	$\{0,1\}$
T_b	a pretty large number as time budget	$T_b > 0$
$r_{i,j,k,l}$	agent i execute task j providing serve l in the order k	$\{0,1\}$
t_j	begin time of task j	$t_j > 0$
p_j	continue time of task j	$p_j > 0$
$a_{j,l}$	number of servey l task j needed.	$\mathcal{N}$
$b_{i,l}$	type of servey l that agent i can provide	$\{0,1\}$
v_i	velocity of agent i	$v_i > 0$
dis_{j_1,j_2}	distance from task j_1 to task j_2	$dis_{j_1,j_2} > 0$
$dis_{i,j}$	distance from initial i to task j	$dis_{i,j} > 0$

TABLE II
DEFINITION OF VARIABLES USED IN THE MILP.

is the objective function. The following constraints should be satisfied. Firstly, the $\preceq_\varphi$ constraints are enforced by

$$t_{j_1} + p_{j_1} \leqslant t_{j_2}, \quad \forall (j_1, j_2) \in \preceq_\varphi; \tag{2}$$

the $\neq_\varphi$ constraints are enforced by: TODOLIU: update

$$t_{j_1} + p_{j_1} - q^l_{j_1,j_2} T_b \leqslant t_{j_2}, \forall S^l = \{j_1, j_2, \cdots, j_n\} \subseteq \neq_\varphi \tag{3}$$

$$\sum_{j_1,j_2\in S^l} q_{j_1,j_2} \leqslant |S^l|(|S^l|-1)-1, \forall S^l = \{j_1, j_2, \cdots, j_n\} \subseteq \neq \varphi \tag{4}$$

Furthermore, the required subtasks for each task j should be satisfied:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} r_{i,j,k,l} b_{i,l} = a_{j,l}, \quad \forall j, l; \tag{5}$$

Each agent can only provide the subtasks within its capability:

$$r_{i,j,k,l} \leqslant b_{i,l}, \quad \forall i, j, k, l; \tag{6}$$

Each agent can execute one task no more than once:

$$\sum_{k=1}^{n}\sum_{l=1}^{o} r_{i,j,k,l} \leqslant 1, \quad \forall i, j; \tag{7}$$

Each agent at any time can execute no more than one task:

$$\sum_{j=1}^{m}\sum_{l=1}^{o} r_{i,j,k,l} \leqslant 1, \quad \forall i, k; \tag{8}$$

Each agent can execute $k+1$-th task only after it has executed k-th task:

$$\sum_{j=1}^{m}\sum_{l=1}^{o} r_{i,j,k,l} - \sum_{j=1}^{m}\sum_{l=1}^{o} r_{i,j,k+1,l} \leqslant 0, \quad \forall i, k < m-1; \tag{9}$$

Each agent should obey its motion and action model:

$$\begin{aligned} &t_{i_2} - t_{i_1} - M\sum_{l=1}^{o} r_{i,j_1,k,l} - M\sum_{l=1}^{o} r_{i,j_2,k+1,l} \geqslant \\ &dis_{j_1,j_2}/v + p_{i_1} - 2M, \quad \forall i, j; \end{aligned} \tag{10}$$

$$t_i - M\sum_{l=1}^{o} r_{i,j,1,l} \geqslant dis_{i,j}/v - M, \quad \forall i, j; \tag{11}$$

With these constraints mentioned, the associated MILP is completed. Unfortunately, the required number of bool variables is MN^2, the complexity of solving the underlying MILP is exploding as M and N increase.

II. Alternative Lower Bound

In this section, another method to compute the lower bound is proposed here, which is more complex than the one proposed in the article. But it can provide more accurate estimate of the lower bound, particularly, it performs better when the number of agents and subtasks are small. Instead of the original conditions in (1) (11), a more relaxed version is used here. More specifically, the distance cost is replaced by the time bound t_{low} as the minimum time for any agent to reach any goal region. The lower bound consists of two parts: the first part is to calculate the exact finishing time of all currently assigned tasks; the second part is to estimate the makespan based on the current boundary condition. In particular, the set of used variables is summarized in Table III. Furthermore, the objective function is given by:

$$\min_{t_{j_a}} \quad \max(t_{j_a} + p_{j_a}),$$

Name	Variable definition	range
P	partial order set	
M	number of agents	$\mathcal{N}$
$\mathcal{M}$	agent set	
N_a	set of assigned tasks	
N_u	set of unassigned tasks	
T_i	assigned tasks for agent i	
n_u	number of unassigned tasks	$\mathcal{N}$
t_{j_a}	finished time of assigned tasks	$t_{j_a} > 0$
Ω	anchor function	
$r_{i,j}$	agent i execute task j	$\{0,1\}$
t_{i0}	begin time of agent i	$t_i > 0$
t_i	end time of agent i	$t_i > 0$
p_j	continue time of task j	$p_j > 0$
p_j'	estimate continue time of task j	$p_j > 0$
a_j	number agent task j needed.	$\mathcal{N}$
v_i	velocity of agent i	$v_i > 0$
dis_{j_1,j_2}	distance from task j_1 to task j_2	$dis_{j_1,j_2} > 0$
$dis_{i,j}$	distance from initial i to task j	$dis_{i,j} > 0$

TABLE III
VARIABLE DEFINITION

and the partial ordering constraints are given by:

$$t_{j_{a1}} + p_{j_{a1}} < t_{j_{a2}}, \quad \forall j_{a1}, j_{a2} \in P, j_{a1} \in N_a, j_{a2} \in N_a; \tag{12}$$

when task j is the first task of agent i, the motion constraint is given by:

$$dis_{i,j_a}/v_i + p_{j_a} < t_{j_a}; \tag{13}$$

when task j_{a2} is the subsequent task after task t_{a1} of agent i, the motion constraint is given by:

$$dis_{j_{a1},j_{a2}}/v_i + p_{j_{a1}} + t_{j_{a1}} - t_{j_{a2}} < 0; \tag{14}$$

then t_{i0} can be computed as the time each agent finishes executing all assigned tasks and starts to execute the remaining unassigned tasks. t_{j_a} is the time when the execution of task j_a is finished by agent i. Denote by T_i the set of tasks assigned to agent i, which yields the computation of t_{i0} as follows:

$$t_{i0} = \max\{t_{j_a}\}, \quad j_a \in T_i; \tag{15}$$

For the remaining tasks, the final lower bound is modified by replacing the task executing time p_j with the minimum possible motion cost p_j':

$$p_j' = \min\{\frac{dis_{i,j}}{v_i}, \frac{dis_{j_1,j_2}}{\max_i v_i}\} + p_j, \quad \forall j \in N_u, \forall i \in \mathcal{M}; \tag{16}$$

Thus, the objective function and constrains are reformulated as follows:

$$\min_{r_{i,j},t_i} \ \max\{t_i\} \tag{17}$$

the number of collaborators for each subtask is enforced by:

$$\sum_{i=1}^{m} r_{i,j} = a_j, \quad \forall j \in N_u; \tag{18}$$

the relaxed constraint on motion and task executing time is given by:

$$\sum_{j=1}^{n} r_{i,j} p_j' + t_i > t_{i0}, \quad \forall i \in \mathcal{M}, \tag{19}$$

and $r_{i,j}$ is a boolean variable which is 1 if task j is assigned to agent i, 0 otherwise:

$$r_{i,j} \in \{0, 1\}. \tag{20}$$

The complexity of the first part is $O(N_a)$, while the complexity of the second part is $O(N_u \cdot M)$. Compared with (1), the ordering relations among subtasks are ignored thus the poset constraint is neglected. In addition, for each subtask, the constraints in (5)-(7) are removed and only the constraint in (18) is kept. Also, a collaborative task is not required to be executed by the collaborators at same time, thus the actual completion time is smaller.

Consequently, the lower bound on the makespan of all solutions rooted from ν is given as the minimum of these two lower bounds above:

$$\underline{T}_\nu = \texttt{lower\_bound}(\nu, P_\varphi) = \min\{\underline{T}_{\nu,1}, \underline{T}_{\nu,2}\}, \quad (21)$$

which can be computed efficiently. It is worth noting that the task assignments associated with $\underline{T}_\nu$ above is infeasible as it either violates the partial ordering constraints or the current agent capacities.

III. Details for Numerical Simulation

A. Planner and Controller

Each agent is governed by a two-level path planner: the high-level planner relies on the $A^\star$ method to find the discrete sequence of waypoints from any initial position to the goal position; the low-level planner is a simple P-controller with RVO for collision avoidance. The agent synthesizes the discrete path first with high level planner, then uses the lower-level planner to navigate the robot between waypoints. The transition function between the regions is different among the agents, e.g., the UAV can move freely without considering any obstacles, while the UGVs are restricted from the areas around the PV panels and off-road obstacles. The collision avoidance mechanism via RVO is only activated when the distance between any two agents is smaller than a threshold.

There are three types of agents: V_f, V_s, V_l: V_f is a quadrotor with model Crazyflie 2.0, which has a onboard navigation controller. All UAVs have the same height and can only move in the x, y plane; V_s, V_f are both four-wheel-driven car with MecanNum wheels, with a simple onboard navigation controller for rotation and translation.

B. Additional Figures

This section contains the figures illustrating the simulation results in Section VIII-A, of the original article, which were omitted for lack of space.

Fig. 2-4 are the poset graph, the BnB search process, and the final execution plan for task φ_1; while Fig. 5-7 are for task φ_2.

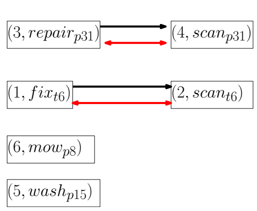

Fig. 2. Poset graph of task φ_1.

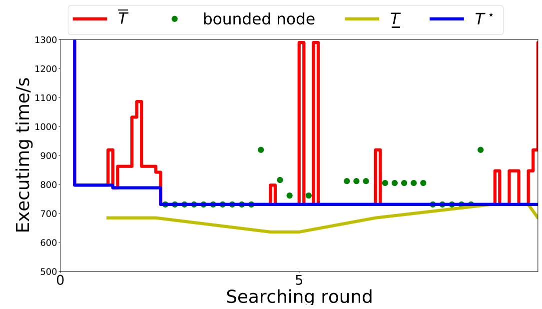

Fig. 3. BnB search process of φ_1.

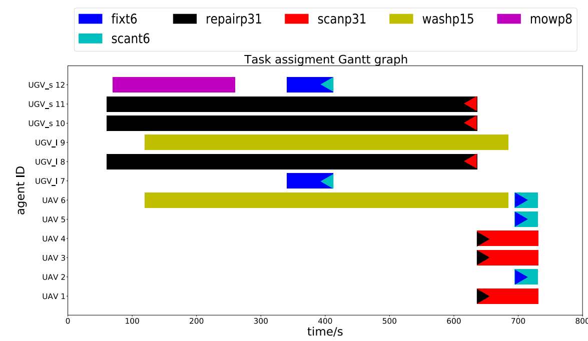

Fig. 4. Gantt graph of optimal task assignment in φ_1.

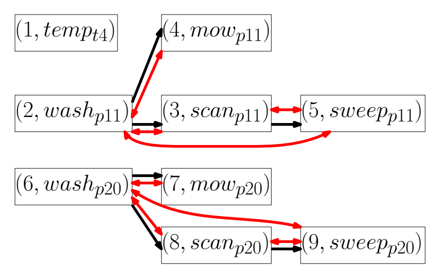

Fig. 5. Poset graph of task φ_2.

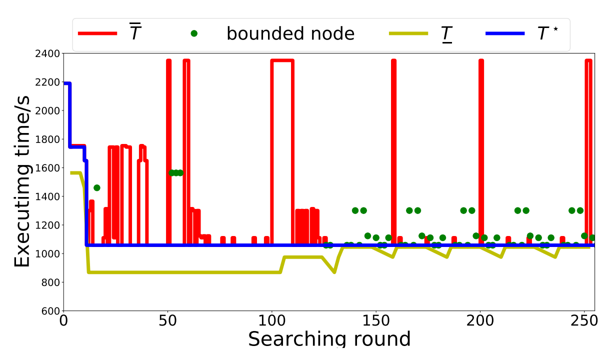

Fig. 6. BnB search process of φ_2.

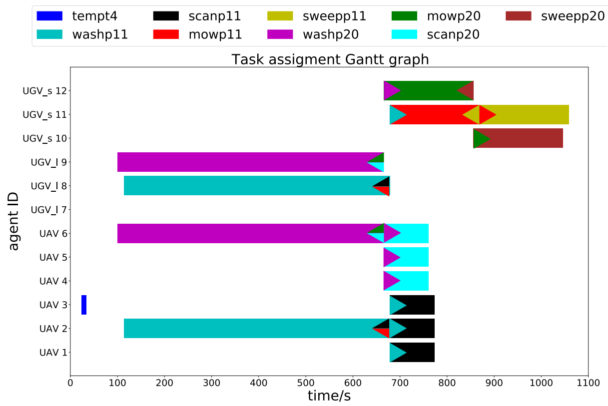

Fig. 7. Gantt graph of optimal task assignment in φ_2.

REFERENCES

[1] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multi-robot systems," *arXiv preprint arXiv:2101.05694*, 2021.

[2] A. M. Jones, K. Leahy, C. Vasile, S. Sadraddini, Z. Serlin, R. Tron, and C. Belta, "Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications," in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 1–16.