Supporting materials

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

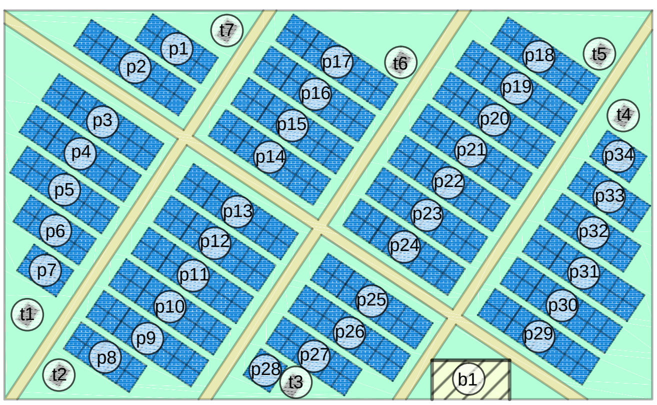

Fig. 1. Work space

I. MILP

Mathematical Models: Given 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 temporal order between the subtasks. $\omega_1 \preceq_\varphi \omega_2$ means subtask ω_2 should begin after the start of ω_1 and $\omega_1 \neq_\varphi \omega_2$ means the execution time of ω_1, ω_2 should not have intersection. An agent swarm is required to execute these subtasks under the constrains of $\preceq_\varphi, \neq_\varphi$ in a 300×500 work space 1. Additionally, the agents are heteroid as there are three different agent type V_f, V_l, V_s with different velocity and different functions. For any subtask $\omega_j \in \Omega$, it needs a particular combination of different type of collaborators as showed in table I. The optimal function is to minimum the max execution time of subtasks Ω. To go further, we give the table of definition for the symbols in this mathematical models.

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	$j \leqslant o$
Ω	set of subtasks	
q_{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
VARIABLE DEFINITION

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

s.t.

$\preceq_\varphi$ constraint of tasks:

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

$\neq_\varphi$ constraint of tasks:

$$t_{j_1} + p_{j_1} + q_{j_1,j_2} T_b \leqslant t_{j_2} \quad \forall (j_1, j_2) \in \neq_\varphi \tag{3}$$

$$t_{j_2} + p_{j_2} + (q_{j_1,j_2} - 1) T_b \leqslant t_{j_1} \quad \forall (j_1, j_2) \in \neq_\varphi \tag{4}$$

provide enough serves for the task j

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

one agent can only provide the serve it has:

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

One agent can execute one task no more than once:

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

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

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

One agent can execute $k+1th$ task only if it execute kth task.

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

Even agent need to obey the motion constrain.

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

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

With the constraints mentioned above, we defined this Mixed interger linear programming(MILP) question. Unfortunately, duo to the number of bool variables is MN^2O, the complexity of this question is exploding as agent number M or task number N increase.

II. Lower Bound method

Here we put out another Lower Bound method which is more complex than the one in article but is more accurate. This method preforms better in the small agent swarm and subtasks that $MN \leqslant 50$ because it can return a more accurate lower bound but it's performance descent tepidly in larger scale system as $MN \geqslant 100$. Comparing to the original question 1 11, we use a simplified mathematical model. Instead of calculating the distance cost caused by different tasks order, we use the time lower bound t_{low} instead. t_{low} is the minimum time for any agent to go to the goal place. The lower bound is consisted of two parts, the first is to calculate the exact solution of current node. The second part is to estimate the makespan based on current boundary condition. The first part is an algorithm of P-hard. We only need to consider the partial order in the assigned tasks and motion constrains. For the partial order constrain:

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

s.t.

$$t_{j_a1} + p_{j_a1} < t_{j_a2} \quad \forall j_{a1}, j_{a2} \in P, j_{a1} \in N_a, j_{a2} \in N_a \quad (12)$$

When task j is the first task of agent i, the motion constrain is :

$$dis_{i,j_a}/v_i + p_{j_a} < t_{j_a} \quad (13)$$

When task j_{a2} is the next task of task t_{a1} in agent i, the motion constrain is:

$$dis_{j_{a1},j_{a2}}/v_i + p_{j_{a1}} + t_{j_{a1}} - t_{j_{a2}} < 0 \quad (14)$$

Then we can generate the t_{i0} of each agent, which means the time agent finished assigned tasks and can begin to execute the left unassigned tasks.t_{j_i} is the assigned task to the agent i. We defined that the task set assigned to agent i is T_i.

$$t_{i0} = \max\{t_{j_a}\} \quad j_a \in T_i \quad (15)$$

To the unassigned task, we propose an algorithm with much simplified to get the final lower bound. Instead of consider the order relationship of task, we use the lower bound of motion cost to unify the executing time for one task in any order. That is rebuilding the task executing time p_j with a minimum possible motion cost as p'_j.

$$p'_j = \min\{\frac{dis_{i,j}}{v_i}, \frac{dis_{j_1,j_2}}{\max v_i}\} + p_j \quad \forall j \in N_u, \forall i \in \mathcal{M} \quad (16)$$

Thus, the optimal function and constrains are in the following:

$$\min_{r_{i,j},t_i} \quad \max(t_i) \quad (17)$$

s.t.

Enough executor constrain:

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

Relaxed motion and task executing time constrain:

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

The complexity of first part is $O(N_a)$, and the complexity of the second part is $O(N_u * M)$. Comparing to 1, we ignore the order relations between tasks so that the execution of each task is compact and the poset constrains is neglected. Also, we relaxed the sub-task type constrains as 5,6,7 and only require enough agent execute a cooperative task as 18. Also, we do not require a cooperation task to execute at same time in different agents thus the optimal value is certainly smaller than the MILP.

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