Decentralized Task Collaboration for Multiagent System with Compact Coupled Tasks under LTL Specifications

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

***Abstract*—We consider the collaboration under compact coupled task specifications of a multiagent system that consists of heterogeneous groups of homogeneous agents. To alleviate the massive computational complexity of centralized multi-agent planning, we propose a decentralized solution, where couple-edge is proposed to eliminate the coupling among compact task and transform the collaboration issue into a local task planning problem. Meanwhile, we adapt the online collaboration scheme based on real-time exchange of request and reply messages to ensure the satisfaction of task specifications without central collaboration. This adapted online collaboration scheme facilitates decreasing the total execution cost in general collaboration scenario. Towards scenarios where partial agents are unstable, union agent model based on previous decentralized solution is proposed to dispatch agents to strive to complete task when part of agents failure. Proposed approach is demonstrated by a simulated scenario in a warehouse and agents are assigned compact coupled task specifications.**

***Index Terms*—multiagent system, linear temporal logics (LTL-s), decentralized collaboration.**

I. Introduction

TEMPORAL logic based motion planning method has gained attention significantly over the last decade because it can provide complete solution for motion planning under specific environment and task specifications. The temporal logic, such as linear temporal logic (LTL) has usage as offering formal language which specify the task in a high level. Meanwhile, model checking algorithms are applied to find the discrete plan according to task specification and the abstraction of environment[principle of ...]. Then the plan is implemented through the low-level physical system.

LTL has been employed by multiagent system to plan the motion path under complex task specifications. In [1],[3] and [4], the essential LTL is employed by a multiagent system to perform coordinated behaviors in Robotic Urban-Like Environment(RULE). Navigation function is combined with LTL in [6] to tackle task specifications. Due to dynamic workspace, agents need to revise their motion path to conform the task specification. [9],[19] propose a motion path revision method based on knowledge transferring and real time Dijstra algorithm. Towards infeasible task, [8] synthesizes the motion plan that fulfills the infeasible task specifications most and relax the infeasible task specifications. LTL is employed in many aspects of research of multiagent system. In [7], the motion of each agent is modeled as a weighted transition system, and the goal is to minimize a cost function which captures the maximum time between successive satisfactions of the optimizing proposition while guaranteeing the formula is satisfied. Discrete Average Space Robustness which is composed of Signal Temporal Logic and Model Predictive Control is proposed in [40]. Besides the robust control, the hybrid control [24,28,29] and game theorem [13] in multiagent systems are also combined with LTL. In addition, human-in-the-loop mixed initiative control is studied in [44] and [43] demonstrates the event trigger scheme of LTL in multiagent systems.

A key challenge of applying LTL to the motion planning of multiagent systems is the computational complexity. In centralized method, all the agents need to product their agent models to synthesize the motion paths which satisfying all the task specifications[add ref], and the complexity exponential increases with the number of states. To alleviate the massive amount of calculation, many methods are proposed. These methods can be classified into two aspects. On the one hand, a receding horizon framework consists of a goal generator, a trajectory planner and a continues controller is proposed in [5], where the goal generator reduces the computational complexity of trajectory generation and meanwhile the desired system-level temporal properties are preserved. This framework is then employed in multiagent system in [10]. On the other hand, the decomposition and decentralization of LTL are studied widely. A two-phase automata-based solution is proposed in [30], where the planning procedure is systematically decoupled for the specifications that constraints agents' trace and expresses the agents' task. [31] demonstrates an automata-based approach to decompose LTL specifications into sets of independently executable task specifications and the hierarchical decomposition of LTL is introduced in [37]. Compared to the decomposition of LTL specifications, decentralization methods tend to reduce the computational complexity more efficiently. [15] presents a decentralized hybrid supervisory control approach. Local LTL tasks are employed in [14] where the connectivity constraints are considered simultaneously. With the preliminary decentralization methods, the reconfiguration of distributed plan is studied in [16]. Recently, a decentralization framework using Tableau approach is proposed in [Efficient Dec...]. Most of these frameworks can't handle coupled task, which requires

M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

J. Doe and J. Doe are with Anonymous University.

Manuscript received April 19, 2005; revised August 26, 2015.

to product all the agent models of multiagent system and results massive computational complexity. In this regard, [jing chang yin] presents a product-free framework towards loosely coupled task. However, the execution of motion is sequential in compact coupled tasks, the dependencies among motions can't be defined clearly, especially for complex and nested compact coupled task specifications. Regarding compact coupled tasks specifications, we proposed a decentralization collaboration framework of multiagent system. At first we decouple the compact coupled task, then through combining with the simplified communication scheme, the computational complexity is reduced significantly. Finally, considering the failures may occur during the work, we proposed a new agent model to assign agents to inherit the task of broken agent with small computational complexity.

Our contribution can be summarized as follows: (1) A formal definition of compact coupled task specifications is introduced and the method decoupling the compact coupled tasks among agents is proposed. (2) A new agent model called union agent model is proposed.

The remaining of this paper is organized as follows: Some preliminaries are introduced in Sec.II. Sec.III states formally the problem. Sec.IV presents the decentralized synthesis of initial plan of each agent. The decentralized collaboration of agents is described in Sec.V. The agent scheduling system is demonstrated in Sec.VI. In Sec.VII we show the overall structure of our framework and case studies are shown in Sec.VIII. Finally, a summary and future work are given in Sec.IX.

II. Preliminaries

A. Linear Temporal Logic

Atomic propositions (AP) are boolean variables that can be either true or false and the ingredients of an linear temporal logic are a set of atomic propositions and several boolean and temporal operators, which are specified according to the following syntax[add ref]: $\varphi ::= \top | a | \varphi_1 \wedge \varphi_2 | \neg\varphi | \bigcirc \varphi | \varphi_1 \cup \varphi_2 | \Box\varphi | \Diamond\varphi | \implies \varphi$, where $\top$ (*True*), $a \in AP$, $\bigcirc$ (*Next*), $\cup$ (*Until*), $\Box$(*Always*), $\Diamond$(*Eventually*) and $\implies$ (*Implicate*). We refer the readers to [add ref] to grasp the detailed semantics of LTL. We consider $\cup$, $\implies$ and $\wedge$ in this paper which can combine the atomic propositions of different agents, thus generating the compact coupling task.

B. Büchi Automaton

Given a LTL formula φ, we can always translate it to a Nondeterministic Büchi Automaton (NBA) $\mathcal{B}_\varphi$ that accepts all the languages that satisfy φ. It is defined as $\mathcal{B}_\varphi = (Q, 2^{AP}, \delta, Q_0, Q_F)$, where Q is a finite set of all the states of Büchi Automaton, 2^{AP} is the set of alphabets, $\delta: Q \times 2^{AP} \to 2^Q$ is the transition relation and $Q_F \subseteq Q$ is the set of accepting states of Büchi Automaton. There are fast translation algorithms[add ref] to obtain $\mathcal{B}_\varphi$ from LTL formula φ.

III. Problem Formulation

We consider a multiagent system organized by $N \geq 1$ agents. Each agent has the capabilities of navigating within the workspace and performing various motions. The agent index is denoted by $\mathcal{I} = \{p_i, i = 1, 2, ..., N\}$. Then, we divide all the agents into different subsets according to their local LTL task specifications, which implies that only agents with the same local LTL task specifications can be divided into one subset. Assume there are M subsets after dividing, denoted by $\mathcal{G} = \{g_j, j = 1, 2, ..., M\}$. $\mathcal{G}$ satisfies: $g_s \subseteq \mathcal{I}$, $\cup_{g_t \subseteq \mathcal{G}} g_t = \mathcal{I}$ and $g_s \cap g_t = \emptyset, \forall g_t, g_s \subseteq \mathcal{G}$.

A. Motion Abstraction

Each agent only knows a part of the full workspace which guarantees the completeness of local task and collaboration task. Towards each agent, its workspace consists of L partitions, denoted by $\Pi^{p_i} = \{\pi_1, \pi_2, ..., \pi_L\}$, assume all the partitions are known by agents a priori. Besides, there is a set of atomic propositions $\digamma^{p_i}$ describing the properties of the workspace of agent p_i. According to [add ref], the motion of p_i is modeled as a finite transition system (FTS) as following[add ref]:

$$\mathcal{F}^{p_i} = (\Pi^{p_i}, \to^{p_i}, \Pi_0^{p_i}, L^{p_i}, \digamma^{p_i})$$

where $\to^{p_i} \subseteq \Pi \times \Pi$ is the transition relation, $\Pi_0^{p_i} \subseteq \Pi^{p_i}$ is the initial region, $L^{p_i}: \Pi^{p_i} \to 2^{\digamma^{p_i}}$ is the labeling function which indicates the properties held by each region. A path of $\mathcal{F}^{p_i}$ is a sequence of regions $\pi_0\pi_1...\pi_s$ where $(\pi_k, \pi_{k+1}) \in \to^{p_i}$, $\forall k = 0, 1, ..., L-1$.

Remark 1: Action models are essentially finite transition systems, therefore the method proposed in this paper is applicable for compact coupled action models. Since the principle is same, the application on action model will not be described in detail.

B. Communication Model

Similar to the communication model in [add ref], the communication network $\mathcal{V} = (\mathcal{N}, \mathcal{E}(t))$ at $t > 0$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}(t) \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges, which has two layers that $\mathcal{E}(t) = \mathcal{E}_1(t) \cup \mathcal{E}_2(t)$. The first layer $\mathcal{E}_1(t) = (p_i, p_j), \forall i, j \in g_t$ and $\mathcal{E}_2(t) = (p_i, p_j), \forall i \in g_t, \forall j \in g_s, t \neq s$. The first layer $\mathcal{E}_1(t)$ is static, which indicates that all the agents belong to the same group g_t can always communicate with each other. The second layer $\mathcal{E}_2(t)$ is dynamic, which indicates that it is enabled when there are decentralized collaboration occurs.

C. Complete Agent Model

Given the NBA $\mathcal{B}^{p_i}$ associated with φ^{p_i} and FTS $\mathcal{F}^{p_i}$, the complete agent model $\mathcal{C}^{p_i}$ of p_i is defined as following:

$$\mathcal{C}^{p_i} = \mathcal{B}^{p_i} \otimes \mathcal{C}^{p_i} = (Q^{p_i,*}, \delta^{p_i}, Q_0^{p_i,*}, Q_F^{p_i,*}, W^{p_i,*}, \tau^{p_i})$$

where $Q^{p_i,*} = \Pi^{p_i} \times Q^{p_i}$, $q^{p_i,*} =< \pi^{p_i}, q^{p_i} >$, $\forall \pi^{p_i} \in \Pi^{p_i}$, $\forall q^{p_i} \in Q^{p_i}$; $(< \pi_m^{p_i}, q_s^{p_i} >, < \pi_n^{p_i}, q_t^{p_i} >) \in \delta^{p_i}$ if $\pi_m^{p_i} \to^{p_i} \pi_n^{p_i}$ and $(q_s^{p_i}, L^{p_i}(\pi_m^{p_i}), q_t^{p_i}) \in \delta^{p_i}$; $Q_0^{p_i,*} = \Pi_0^{p_i} \times Q_0^{p_i}$ is the set of initial states; $Q_F^{p_i,*} = \Pi^{p_i} \times Q_F^{p_i}$ is the set of accepting states; $W^{p_i,*}: \delta^{p_i} \times Q^{p_i,*} \to \mathbb{R}^+$. $W^{p_i,*}(< \pi_m^{p_i}, q_s^{p_i} >, < \pi_n^{p_i}, q_t^{p_i} >)$ is the distance between $\pi_m^{p_i}$ and $\pi_n^{p_i}$, where $< \pi_n^{p_i}, q_t^{p_i} > \in \delta^{p_i}(< \pi_m^{p_i}, q_s^{p_i} >, L^{p_i}(\pi_m^{p_i}))$. τ^{p_i} indicates whether $(< \pi_m^{p_i}, q_s^{p_i} >, < \pi_n^{p_i}, q_t^{p_i} >)$ is a couple edge which is going to be introduced next.

D. Compact Coupled Task Specification

In this paper we consider the compact coupled task specification. Compared to loosely coupled task specifications, compact coupled task specifications relevant to the motions of two or more agents tightly, thus can't be translated to Büchi Automaton only according to the local agent model and action specification as previous method[add ref]. The compact coupled task specification takes an recursive definition as following.

Definition 1: Given a LTL formula φ contains temporal operators such as $\cup$, $\Longrightarrow$ and $\wedge$, we denote the terms around each temporal operators by t_i^{left} and t_i^{right}, respectively. if t_i^{left} and t_i^{right} belongs to AP of different agents or one of them is a compact coupled task specification, then we say φ is a compact coupled task specification.

In other words, a compact coupled task specification should combine the motions of different agents through $\cup$, $\Longrightarrow$, $\wedge$ so that we can't translate it to Büchi Automaton and synthesize an initial motion plan for each agent. In summary, we consider following problem.

Problem 1: Given the complete agent model $\mathcal{C}^{p_i}$ and a serious of task specifications φ which contain compact coupled task specifications, design a decentralized collaboration scheme such that φ is fulfilled for all $p_i \subseteq \mathcal{I}$.

Remark 2: The loosely coupled task specifications can be transformed into compact coupled task specifications. Such as the task specificated as $\Diamond(\mathsf{pick}_2^2 \wedge \Diamond(R_4 \wedge \mathsf{drop}_2^2))$ in [jing chang yin], where pick_2^2 and drop_2^2 need assisting actions hpick_2^2 and hdrop_2^2, respectively, it can be specificated in a compact coupled manner as $\Diamond((\neg\mathsf{pick}_2^2 \cup \mathsf{hpick}_2^2) \wedge \Diamond(R_4 \wedge (\neg\mathsf{drop}_2^2 \cup \mathsf{hdrop}_2^2)))$, then proposed method is applicable to loosely coupled tasks as well.

IV. Decentralized Initial Plan Synthesis

In this section, we will demonstrate how we decouple the compact coupled task specifications through a set of special edge called couple-edge. Then we use the decoupled Büchi Automaton to synthesize an initial motion plan for each agent.

A. Decoupling the Compact Coupled Task

We consider the basic compact coupled task such as $\varphi_1 \cup (\varphi_2 \wedge \varphi_3)$ and $\varphi_1 \Longrightarrow \bigcirc(\varphi_2 \wedge \varphi_3)$, the nested compact coupled task can be transform to basic terms through utilizing auxiliary words. For example, we can transform $\varphi_1 \cup (\varphi_2 \Longrightarrow \bigcirc\varphi_3)$ into $\varphi_1 \cup \kappa$ and $\kappa = \varphi_2 \Longrightarrow \bigcirc\varphi_3$.

Towards basic compact coupled task specification, we notice that compact coupling only exists in the task cooperation process and neither the motions nor states of agent directly participate in the construction of complete models of other agents. For example, consider compact coupled task specification $\neg\varphi_1 \cup \varphi_2$, according to definition 1, φ_1 and φ_2 belongs to different complete agent models. Consider the local complete agent model $\mathcal{C}^{p_i}$ and denote all the atomic propositions associated with φ is denoted by $sym(\varphi)$. Then $sym(\varphi_1) \in Q^{p_i}|_{\mathcal{F}^{p_i}}$, and $sym(\varphi_2) \notin Q^{p_i}|_{\mathcal{F}^{p_i}}$. There exists a subset of Q^{p_i} denoted by ϵ that $sym(\varphi_1) \in \epsilon|_{\mathcal{F}^{p_i}}$ and $sym(\varphi_1) \notin Q^{p_i} \setminus \epsilon|_{\mathcal{F}^{p_i}}$. Then there is no edge from $Q^{p_i} \setminus \epsilon$ to ϵ so that we can't synthesize an initial motion plan fulfill φ.

Definition 2: Given a complete agent model $\mathcal{C}^{p_i}$, If two nodes q_i, q_j satisfy following properties:
(1) $q_i \in \epsilon$, $q_j \in Q^{p_i} \setminus \epsilon$ or vice versa.
(2) $\delta(q_i, q_j) \subseteq sym(\varphi_c)$ or $\delta(q_j, q_i) \subseteq sym(\varphi_c)$.
where φ_c denote the compact coupled part of task specification φ. We call each edge which connects q_i and q_j the couple-edge. At first we can assume that all the collaborative agents have provided assistance and add all the couple-edge to the local complete agent model, then we can synthesize an initial motion plan for each agent.

We adapt previous local product algorithm[add ref] which combines the FTS and Büchi Automaton of single agent for finding out all the couple-edge with respect to the compact coupled task specification φ_c and add them into the origin complete agent model. Applying Algorithm 1 we can get a new complete agent model $\mathcal{C}_n^{p_i}$ for each agent, which contains $\mathcal{C}^{p_i}$ and all the couple-edges.

```
Algorithm 1: Construct New Complete Agent Model $\mathcal{C}_n^{p_i}$
   Input: the local FTS $\mathcal{F}^{p_i}$, the local Büchi Automaton
          $\mathcal{B}^{p_i}$ and the compact coupled task specification $\varphi_c$
   Output: complete agent model $\mathcal{C}_n^{p_i}$
1  foreach $\pi_l \in \Pi^{p_i}, q_m \in Q^{p_i}$ do
2      $q_s^* =< \pi_l, q_m >\in Q^{p_i,*}$
3      if $\pi_l \in \Pi_0^{p_i}$ and $q_m \in Q_0^{p_i}$ then
4          $q_s^* \in Q_0^{p_i,*}$
5      if $q_m \in Q_F$ then
6          $q_s^* \in Q_F^{p_i,*}$
7      foreach $\pi_z \in Post(\pi_l), q_n \in Post(q_m)$ do
8          $q_g^* =< \pi_z, q_n >\in Q^{p_i,*}$
           $d = CheckTranB(q_m, L^{p_i}(\pi_l), q_n, \mathcal{B}^{p_i})$
9          if $d \geqslant 0$ then
10             $q_g^* \in \delta(q_s^*)$
11             $W^{p_i,*}(q_s^*, q_g^*) = dist(\pi_l, \pi_z)$
12             $\tau(q_s^*, q_g^*) = None$
13         else
14             if $L^{p_i}(\pi_l) \in sym(\varphi_c)$ then
15                 $q_g^* \in \delta(q_s^*)$
16                 $W^{p_i,*}(q_s^*, q_g^*) = dist(\pi_l, \pi_z)$
17                 $\tau(q_s^*, q_g^*) = L^{p_i}(\pi_l)$
18 return $\mathcal{C}_n^{p_i}$
```

Lemma 1: By adding all the couple-edges found out by applying Algorithm 1 to $\mathcal{C}^{p_i}$, we can synthesize an initial accepted motion plan for each agent.

proof: According to algorithm 1, the nodes in $Q^{p_i} \setminus \epsilon$ are connected because these nodes are correspond to independent task specifications. Through adding couple-edges into $\mathcal{C}^{p_i}$, each node in ϵ becomes reachable and $\mathcal{C}_n^{p_i}$ becomes fully connected. Consequently, there must exist spanning trees for $\mathcal{C}_n^{p_i}$ and there must exist a path from any initial node in $Q_0^{p_i}$ to any accepted node in $Q_F^{p_i}$. So we can synthesize an initial motion plan fulfilling φ_c according to $\mathcal{C}_n^{p_i}$.

B. Plan Synthesis and Reconstitution

Given $\mathcal{C}_n^{p_i}$ and φ^{p_i}, there exists a finite path satisfying φ^{p_i} if and only if $\mathcal{C}_n^{p_i}$ has a finite path from an initial state to an accepting state. This finite path can be projected back to $\mathcal{F}^{p_i}$ as a finite path. Let $\varrho^{p_i} = q_0^{p_i,*} q_1^{p_i,*} ... q_s^{p_i,*}$ be a finite path, where $q_0^{p_i,*} \in Q_0^{p_i,*}$, $q_s^{p_i,*} \in Q_F^{p_i,*}$ and $q_z^{p_i,*} \in Q^{p_i,*}$ and $(q_z^{p_i,*}, q_{z+1}^{p_i,*}) \in \delta^{p_i}$. The *cost* of ϱ^{p_i} is denoted by $cost(\varrho^{p_i}, \mathcal{C}_n^{p_i}) = \sum_{k=0}^{s} W^{p_i,*}(q_k^{p_i,*}, q_{k+1}^{p_i,*})$.[add ref] Apply [add ref, Algorithm 1] to find ϱ^{p_i}, which utilizes Dijkstra algorithm [add ref] to find the shortest path from any initial state in $Q_0^{p_i,*}$ to every reachable accepting state in $Q_F^{p_i,*}$.

Lemma 2: This decentralized initial plan synthesis method has solutions as long as original centralized plan synthesis method has solutions.

Proof: In algorithm 1 we assume that all the collaborative agents have provided assistance and add all the couple-edges into $\mathcal{C}^{p_i}$. Then towards $\mathcal{C}_n^{p_i}$, φ_c is no longer a constraint. The constraint becomes $\varphi \setminus \varphi_c$. Assume we synthesize an initial motion plan ϱ^{p_i} applying original centralized plan synthesis method, ϱ^{p_i} fulfills φ and then ϱ^{p_i} must fulfill $\varphi \setminus \varphi_c$, so all the solutions generated by original centralized method are contained in the solutions generated by decentralized method. Consequently, the proposed decentralized initial plan synthesis method has solutions as long as original centralized plan synthesis method has solutions.

ϱ^{p_i} constructed by [zhao na ge suan fa] is composed of prefix $\varrho_p^{p_i}$ and suffix $\varrho_s^{p_i}$. In order to adapt to next algorithms, ϱ^{p_i} is reconstituted as following:

(1)Catenate $\varrho_p^{p_i}$ and $\varrho_s^{p_i}$ and use γ to indicate the separation point.

(2)If two state nodes $q_s^{p_i,*}$ and $q_t^{p_i,*}$ satisfy the properties in *Definition 2*, add $\tau(q_s^{p_i,*}, q_t^{p_i,*})$ to the attribute req^{p_i} of $q_s^{p_i,*}$.

V. Decentralized Task Collaboration

In previous section, we assume that all the collaborative agents have provided assistance and add all the couple-edges into $\mathcal{C}^{p_i}$, which permits us to synthesize an initial motion plan for each agent. If each agent runs on its own local path independently without any communication with other agents, there may be no guarantee that the collaborative task will be completed, which may result in a violation of the task specification φ. In response to this issue, we adapt the online collaboration scheme to ensure the completion of the collaborative task.

A. Planning In Speed Matching Horizon

In order to reduce the complexity of task collaboration, we apply a receding horizon method for each agent. Consider the initial reconstituted motion plan ϱ^{p_i} previous, assume the current position of agent p_i is π^{p_i} which is the lth element of ϱ^{p_i} namely $\varrho^{p_i}[l]$. Assign a receding time T_H for each agent, note that different agents have different velocity so each agent will adjust its velocity according to its speed. Assume that each agent could estimate the approximate time according its current velocity, departure position and target position, then the segment the agent expected to execute in T_H is denoted by $\varrho_H^{p_i}$, $\varrho_H^{p_i} = \varrho^{p_i}[l:t]$, where the index t is the solution of the following optimization problem: min t, subject to $\sum_{k=l}^{t} T^{p_i}(\varrho^{p_i}[k], \varrho^{p_i}[k{+}1]) \geq T_H$, this issue can be solved by iterating through the ϱ^{p_i} and computing the accumulated time which is compared to T_H. If previous optimization problem doesn't have a solution, it means that in T_H the agent can execute the rest of the suffix of ϱ^{p_i}, namely $\varrho^{p_i}[l:]$. Since the split point of ϱ^{p_i} has been indicated previously, original optimization problem can be transformed into a new optimization problem: min t, subject to $\sum_{k=\gamma}^{t} T^{p_i}(\varrho^{p_i}[k], \varrho^{p_i}[k+1]) \geq T_H - \sum_{k=l}^{|\varrho^{p_i}|} T^{p_i}(\varrho^{p_i}[k], \varrho^{p_i}[k+1])$, where $|\varrho^{p_i}|$ denote the length of ϱ^{p_i} and γ denote the split point index. We can always find a solution to above problem iteratively.

B. Online Collaboration Scheme

Through speed matching receding horizon planning, the truncation index is founded and the receding horizon motion plan $\varrho_H^{p_i}$ is synthesized. To ensure the fulfill of task specification φ, each agent needs to check whether it needs others' collaboration within $\varrho_H^{p_i}$. Since we have reconstituted the motion plan of each agent, we can just find the first item in $\varrho_H^{p_i}$ which has req^{p_i} attribute. After one agent confirms that it needs others' collaboration, a **Request**^{p_i} is sent to others. More specifically, for the first item $< q_c^{p_i}, req^{p_i} > \in \varrho_H^{p_i}$ satisfying $req^{p_i} \neq \emptyset$, where c denote the index of $< \pi_c^{p_i}, q_c^{p_i}, req^{p_i} >$ in $\varrho_H^{p_i}$. Agent p_i needs to broadcast the request message to $\mathcal{G} \setminus g_s, p_i \in g_s$, through its communicate network. **Request**^{p_i} has the following format:

$$\textbf{Request}^{p_i} = (\chi^{p_i}, \pi_c^{p_i}, q_c^{p_i}, T_e^{p_i})$$

where χ^{p_i} denotes all the items in req^{p_i}, which indicates the collaborative motion agent p_i needs in $\pi_c^{p_i}$, and $T_e^{p_i}$ denotes the estimated time spent from $\varrho^{p_i}[l]$ to $\varrho^{p_i}[c]$, namely $T_e^{p_i} = \sum_{k=l}^{c} T^{p_i}(\varrho_H^{p_i}[k], \varrho_H^{p_i}[k+1])$. Since $\varrho_H^{p_i}[c]$ is the predecessor of the first request state in $\varrho_H^{p_i}$, $\varrho_H^{p_i}[c]$ can be always reached within finite time, so $T_e^{p_i}$ is a finite positive number.

If agent p_i proposes a request message and there is no reply message transitorily, all the collaborations p_i requests are stored in $\mathcal{R}_c^{p_i}$, the current state $\varrho_r^{p_i}$ is stored.

The algorithm finding out the first request in receding horizon motion plan $\varrho_H^{p_i}$ is as Algorithm 2.

Once there are request messages transmitted among agents, some agents should revise their motion plan to provide assistance. The procedure of determining which agent to respond to the request is quite similar with the algorithm proposed in [add ref], we only present a brief overview here and we refer the readers to [add ref] to grasp more detail. In order to provide collaboration, the collaborative agent needs to revise its motion plan temporarily. Assume current state of agent p_j is $\varrho^{p_j}[l]$ and next accepting state is $\varrho^{p_j}[f]$, original motion plan from $\varrho^{p_j}[l]$ to $\varrho^{p_j}[f]$ is $R_-^{p_j} = \varrho^{p_j}[l:f]$ and the motion plan after revision is denoted by $R_+^{p_j}$, the index of the position in where agent p_j provides collaboration is m. We require each agent to calculate the cost of providing assistance in order to decide which agent to collaborate. The *balanced cost* is

Algorithm 2: Plan in Horizon and Request

```
Input: $\varrho^{p_i}, \pi^{p_i}, T_H, \gamma$
Output: $\varrho^{p_i}_H$, Request$^{p_i}$
1  $\varrho^{p_i}[l] = \pi^{p_i}, m = 0, T_m = 0,$ Request$^{p_i} = \emptyset$
2  while $T_m < T_H$ do
3      $m = m + 1$
4      $n = m$
5      if $m > |\varrho^{p_i}|$ then
6          $m = \gamma - l$
7      $T_m = T_m + T^{p_i}(\varrho^{p_i}[l + n - 1], \varrho^{p_i}[l + m])$
8      $< \pi^{p_i}_c, q^{p_i}_c, req^{p_i} >= \varrho^{p_i}[l + m]$
9      if $req^{p_i} \neq \emptyset$ and Request$^{p_i} = \emptyset$ then
10         forall the $\chi^{p_i} \in req^{p_i}$ do
11             add $(\chi^{p_i}, \pi^{p_i}_c, q^{p_i}_c, T_m)$ to Request$^{p_i}$
12 if $m > n$ then
13     $\varrho^{p_i}_H = \varrho^{p_i}[l : l + m]$
14 else
15     $\varrho^{p_i}_H = \varrho^{p_i}[l :] \oplus \varrho^{p_i}[\gamma : m]$
16 return $\varrho^{p_i}_H$, Request$^{p_i}$;
```

defined as following:

$$
\begin{aligned}
BalCost(R^{p_j}_+, T^{p_i}_e, \mathcal{C}^{p_j}_n) = \\
|(\sum_{k=l}^{m} T^{p_j}(\varrho^{p_j}[k], \varrho^{p_j}[k+1]) - T^{p_i}_e)| \quad (1)\\
+ \beta_{p_j} \times (Cost(R^{p_j}_+, \mathcal{C}^{p_j}_n) - Cost(R^{p_j}_-, \mathcal{C}^{p_j}_n))
\end{aligned}
$$

where the first part denotes the time gap between the agent which sends a request message arriving the position where needs collaboration and corresponding agent arrive the position where assistance is provided. Simply, the first part denotes the waiting time of multiagent system, which contains both the waiting time of request agent and reply agent. The second part denotes the additional cost of $R^{p_i}_+$ compared with $R^{p_i}_-$, and β_{p_j} is a designed parameter as relative weighting. Instead of formulating the agent choice issue as an integer programming problem,[add ref], we transmit the balanced cost of each agent among the group g_t, $p_j \in g_t$, since each agent calculates its balanced cost in the same way, there must exist a consensus agent p_{min} whose balanced cost is minimal and p_{min} will be elected to reply to corresponding **Request**^{p_i}.

After p_{min} is elected, we design an attribute of agent named $state^{p_i}$ which turns into $locked$ when agent p_i is providing collaboration, otherwise $state^{p_i}$ is $normal$. Meanwhile, due to the property of compact coupled task, all the request messages that have time overlapping can be replied by a group of heterogeneous agents. So towards each request message **Request**^{p_i}, if there is already a group of agents p_i whose $state^{p_i}$ are $locked$, p_i automatically become the collaborative agents and the corresponding request message becomes invisible to other agents.

Towards some compact coupled task, such as agents can't across region r_i until some agents reach region r_j, only a group of agents is needed to reply request messages from different agents. Hence if two request messages **Request**^{p_i} and **Request**^{p_j} have time overlapping and there is a group of agents ∂ replying one request messages, then ∂ is selected as the reply agents for the other request message **Request**$'$ and **Request**$'$ becomes invisible to other agents.

$R^{p_i}_+$ can be constructed by the $bidirectional\ Dijkstra$ algorithm which utilizes the function $bidirectional\_dijkstra()$ in $Networkx$, we refer the readers to [add ref] for more details. After the reply agent p_r is elected, when p_i reaches the requested replying position and provides collaboration, there is a reply message sent from the reply agent to request agent, whose format is as following:

$$\textbf{Reply}^{p_r} = (p_r, p_i, \xi)$$

where p_r denotes the grade of replying agent and ξ denotes the collaboration provided.

Finally, we adapt the purpose and structure of **Confirm**^{p_i} to suit the compact coupled task specification. The format of confirm message is as following:

$$\textbf{Confirm}^{p_i} = (p_i, p_r, \hat{\xi})$$

where p_r denotes the grades of all replying agents and $\hat{\xi}$ denotes that p_i has relieved collaboration requirements for action $\hat{\xi}$. It is sent when the agents which send the request message separate themselves from the area where the collaboration is required and enter next state and the confirm messages informs the replying agents to recovery their $state^{p_r}$ from $locked$ to $normal$. Denote all the request, reply and confirm messages which agent p_i received by **Request**$^*_{p_i}$, **Reply**$^*_{p_i}$, **Confirm**$^*_{p_i}$, respectively. The entire algorithm to handle request, reply and confirm messages is as Algorithm 3.

Only if $\mathcal{R}^{p_i}_c$ is empty and $state^{p_i}$ is $normal$, agent p_i can continue to execute its motion plan, otherwise p_i needs to wait for the completion of collaboration.

VI. Union Agent Model

In general collaboration scenarios, previous decentralized solution can complete the task collaboration. However, in some special scenarios such as unstable agents and task swapping between different agents, the initial motion path can't be synthesized through local planning. In this section, suppose there are several weak agents that are prone to breakdown and we introduce how to mobilize other agents to inherit the task of weak agents when failure occurs to maximize the completion of task and meanwhile reduce the computational complexity.

At first we define a virtual order $\mathcal{O}$ which indicates the importance of each individual task that each agent undertakes, towards agent p_i, if its task is more important, or the agents number in g_j where $p_i \in g_j$ is fewer, the $\mathcal{O}_{p_i}$ is larger. Assume $\mathcal{O}_{p_i} \succ \mathcal{O}_{p_j}$, then when p_i is breakdown, p_j needs to revise its motion plan to inherit the task of p_i, otherwise p_j still fixes on its own task.

When p_i is broken down, there is a **Warning**^{p_i} message sent to all the other agents, the format of **Warning**^{p_i} is as following:

$$\textbf{Warning}^{p_i} = (p_i, \mathcal{C}^{p_i}_n, \aleph^{p_i})$$

Algorithm 3: Handle Reply and Confirm messages

```
Input: Request$^*_{p_i}$, Reply$^*_{p_i}$, Confirm$^*_{p_i}$, $\partial$, $\mathcal{R}^{p_i}_c$, $\mathcal{C}^{p_i}_n$
Output: $R^{p_j}_+$
1  $\mathcal{A}^{p_i}_s = \emptyset$
2  if Request$^*_{p_i} \neq \emptyset$ then
3      foreach Request$^r = (\chi^r, \pi^r_c, q^r_c, T^r_e) \in$ Request$^*_{p_i}$ do
4          if $\partial \neq \emptyset$ and $p_i \notin \partial$ then
5              continue
6          else if $(\partial \neq \emptyset$ and $p_i \in \partial)$ or
           $(\partial = \emptyset$ and $min_{g_s} BalCost = p_i)$ then
7              $(R^{p_i}_+, b^{p_i}_d, t^{p_i}_d) =$
               $EvalReq(R^{p_i}_-, (\chi^{p_i}, \pi^{p_i}_c, q^{p_i}_c, T^{p_i}_e), \varrho^{p_i}[l], \mathcal{C}^{p_i}_n)$
8              $state^{p_i} = locked$
9              add $\chi^r$ to $\mathcal{A}^{p_i}_s$
10             send Reply$^{p_i} = (p_i, p_r, \xi)$ to $p_r$
11 if Reply$^*_{p_i} \neq \emptyset$ then
12     foreach Reply$^r = (p_r, p_i, \xi) \in$ Reply$^*_{p_i}$ do
13         if $\xi \in \mathcal{R}^{p_i}_c$ then
14             delete $\xi$ from $\mathcal{R}^{p_i}_c$
15         if $\mathcal{C}^{p_i} \in Post(\varrho^{p_i}_r)$ then
16             send Confirm$^{p_i} = (p_i, p_r, \hat{\xi})$ to $p_r$
17 if Confirm$^*_{p_i} \neq \emptyset$ then
18     foreach Confirm$^r = (p_i, p_r, \hat{\xi}) \in$ Confirm$^*_{p_i}$ do
19         if $\hat{\xi} \in \mathcal{A}^{p_i}_s$ then
20             delete $\hat{\xi}$ from $\mathcal{A}^{p_i}_s$
21         if $\mathcal{A}^{p_i}_s = \emptyset$ then
22             $state^{p_i} = normal$
23 return $R^{p_j}_+$
```

which means p_i needs to broadcast its grade p_i, local complete agent model $\mathcal{C}^{p_i}_n$ and capacity $\aleph^{p_i}$ when it is broken down. After **Warning**^{p_i} is broadcast, all the agents are qualified for capacity $\aleph^{p_i}$, denoted by ν, will respond to the **Warning**^{p_i}. All the agents in ν with the same minimal $\mathcal{O}_{p_j}$ become candidates for inheriting the task of p_i. In order to elect an agent to respond to the warning message and revise its motion plan, we need to solve following problem:

Problem 2: With a little symbol abuse, we denote the states in $\mathcal{C}^{p_i}_n$ by $\tilde{c}_{p_i}$, the motion plan of p_j after revision by $R^{p_j}_+$ and the original one by $R^{p_j}_-$, respectively. Assume that all the hard constraints, such as $\neg obstacle$ (never across the obstacles), have been specified in every agent, which implies the inheriting agent can discard original task specification temporarily and transfer to inherited task under the guarantee of safety. Given the local complete agent model $\mathcal{C}^{p_i}_n$ and $\mathcal{C}^{p_j}_n$, decide the grade ϖ of the agent inheriting the task of p_i and construct R^{ϖ}_+ to minimize the total cost.

$\mathcal{C}^{p_i}_n$ and $\mathcal{C}^{p_j}_n$ can be combined to obtain $union\ agent\ model$. The union agent model $\widetilde{\mathcal{U}}_{ij}$ is defined as following:

$$\widetilde{\mathcal{U}}_{ij} = (\widetilde{Q}^{ij}, \tilde{\delta}^{ij}, \widetilde{Q}^{ij}_0, \widetilde{Q}^{ij}_F, \widetilde{W}^{ij}, \tilde{\tau}^{ij})$$

where $\widetilde{Q}^{ij} = Q^{p_i,*} \cup Q^{p_j,*}$; $\tilde{\delta}^{ij} = \delta^{p_i} \cup \delta^{p_j} \cup (\tilde{c}_{p_i}, \tilde{c}_{p_j})$, $\tilde{c}_{p_i}$ is bordered to $\tilde{c}_{p_i}$; $\widetilde{Q}^{ij}_0 = Q^{p_i,*}_0 \cup Q^{p_j,*}_0$; $\widetilde{Q}^{ij}_F = Q^{p_i,*}_F \cup Q^{p_j,*}_F$; $\widetilde{W}^{ij} = W^{p_i,*} \cup W^{p_j,*}$ and $\tilde{\tau}^{ij} = \tau^{p_i} \cup \tau^{p_j}$.

Assume that part of the local FTS $\mathcal{F}^{p_i}$ and $\mathcal{F}^{p_j}$ of p_i and p_j are bordered and then the construction procedure of union agent model $\widetilde{\mathcal{U}}_{ij}$ is as following:

(1)Add all the nodes and edges in $\mathcal{C}^{p_i}_n$ and $\mathcal{C}^{p_j}_n$ into $\widetilde{\mathcal{U}}_{ij}$.

(2)Find all the position pair $\tilde{s} = (\tilde{\eta}_{p_i}, \tilde{\eta}_{p_j})$, where $\tilde{\eta}_{p_i}$ and $\tilde{\eta}_{p_j}$ appertain to $\mathcal{F}^{p_i}$ and $\mathcal{F}^{p_j}$, respectively and they are bordered, which means agent can transition from $\tilde{\eta}_{p_i}$ to $\tilde{\eta}_{p_j}$ by one step and vice versa. We call these position pair bordered position pair and denote the set of $\tilde{\eta}_{p_i}$ and $\tilde{\eta}_{p_j}$ by $\hat{\eta}_{p_i}$ and $\hat{\eta}_{p_j}$, respectively.

(3)For all states $\tilde{c}_{p_i}$ and $\tilde{c}_{p_j}$ in $\mathcal{C}^{p_i}_n$ and $\mathcal{C}^{p_j}_n$, if $\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}$ and $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}$ is a bordered position pair and there are no violations of task specifications when transferring from $\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}$ to $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}$, then we add an edge $(\tilde{c}_{p_i}, \tilde{c}_{p_j}, weight)$ into $\widetilde{\mathcal{U}}_{ij}$, where $weight$ denotes the distance between $\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}$ and $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}$. If the transition violates the coupled task specifications and $(\tilde{c}_{p_i}, \tilde{c}_{p_j})$ is a couple-edge according to *Definition 2*, then add $(\tilde{c}_{p_i}, \tilde{c}_{p_j}, weight)$ as a couple-edge into $\widetilde{\mathcal{U}}_{ij}$.

After the union agent model is constructed, we decide ϖ and synthesize R^{ϖ}_+. Denote the current state of p_j is $\tilde{c}^*_{p_j}$, p_j needs to interrupt its own task so that the states of Büchi Automaton can no longer restrict the motion of p_j and each $\tilde{c}_{p_j}$ satisfying $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}} = \tilde{c}^*_{p_j}|_{\mathcal{F}^{p_j}}$ can be the departure point of R^{ϖ}_+. Therefore, once p_j is assigned to inherit the task of p_i, all the $\tilde{c}_{p_j}|_{\mathcal{B}^{p_j}}$ are deposited and $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}$ are composed with the $Q^{p_i,*}_0|_{\mathcal{B}^{p_i}}$.

Remark 3: Actually, the effect of above method is the same as the union of $\mathcal{F}^{p_j}$ and $\mathcal{C}^{p_i}_n$. We deposit the $\tilde{c}_{p_j}|_{\mathcal{B}^{p_j}}$ instead of abandoning them in order to use the same model to plan p_j to go back to its own task after p_i is repaired.

It is obvious that the suffix $R^{\varpi,s}_+$ is invariant due to the inheriting of task specification φ_i, so we can transform the *Problem 2* to the synthesizing of prefix $R^{\varpi,p}_+$ from any $\tilde{c}_{p_j}$ satisfying $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}} = \tilde{c}^*_{p_j}|_{\mathcal{F}^{p_j}}$ to $\tilde{c}_{p_i}$ where $\tilde{c}_{p_i}|_{\mathcal{B}^{p_i}} \in Q^{p_i,*}_F|_{\mathcal{B}^{p_i}}$. We apply $DijksTargets$ method [add ref] to synthesize $R^{\varpi,p}_+$, where all the $\tilde{c}_{p_j}$ satisfying $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}} = \tilde{c}^*_{p_j}|_{\mathcal{F}^{p_j}}$ are set as source nodes and all $\tilde{c}_{p_i}$ satisfying $\tilde{c}_{p_i}|_{\mathcal{B}^{p_i}} \in Q^{p_i,*}_F|_{\mathcal{B}^{p_i}}$ are set as target nodes and we can obtain the shortest path between two nodes from source nodes and target nodes, ultimately, the shortest path in all the shortest paths is selected as $R^{\varpi,p}_+$. Then $R^{\varpi,p}_+$ is synthesized:

$$R^{\varpi}_+ = R^{\varpi,p}_+ \oplus R^{\varpi,s}_-$$

Then all the candidates of inheriting the task need to elect the agent with minimal cost to revise its motion plan, so all the agents calculate $Cost(R^{\varpi}_+)$ and transmit it among all the candidates until one of them is elected to inherit the task.

When the broken down agent is fixed properly, we can drive the inheriting agent to return to its own task by reversing above method. The entire procedure of constructing the union agent model and each candidate finds out R^{ϖ}_+ is exhibited by Algorithm 4 and *Problem 2* is solved.

Algorithm 4: the construction of $\widetilde{\mathcal{U}}_{ij}$ and the synthesis of R_+^{ϖ}

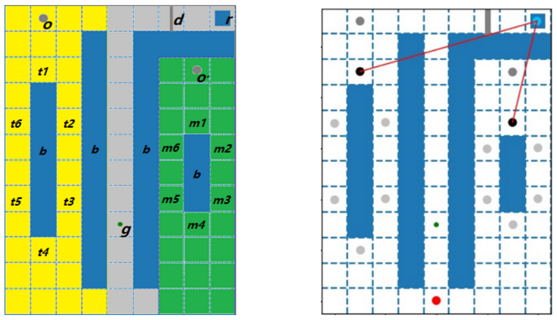
```
    Input: $\mathcal{C}_n^{p_i}$, $\mathcal{C}_n^{p_j}$ and $\tilde{c}_{p_j}^*$
    Output: $\widetilde{\mathcal{U}}_{ij}$ and $R_+^{\varpi}$
1   $\widetilde{\mathcal{U}}_{ij} = \mathcal{C}_n^{p_i} \cup \mathcal{C}_n^{p_j}$
2   $\tilde{\eta}_{p_i} = \emptyset, \Omega_{p_j} = \emptyset$
3   foreach $\tilde{c}_{p_j} \in \mathcal{C}_n^{p_j}$ do
4       if $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}} = \tilde{c}_{p_j}^*|_{\mathcal{F}^{p_j}}$ then
5           add $\tilde{c}_{p_j}$ to $\Omega_{p_j}$
6       foreach $\tilde{c}_{p_i} \in \mathcal{C}_n^{p_i}$ do
7           if $\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}$ is bordered to $\tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}$ then
8               add $\tilde{c}_{p_i}$ to $\tilde{\eta}_{p_i}$
9               if $CheckTranB(\tilde{c}_{p_i}|_{\mathcal{B}^{p_i}}, L^{p_i}(\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}), \tilde{c}_{p_j}|_{\mathcal{B}^{p_j}}, \tilde{c}_{p_j}|_{\mathcal{F}^{p_j}}) \geqslant 0$ then
10                  $\tilde{c}_{p_j} \in \tilde{\delta}^{ij}(\tilde{c}_{p_i})$
11                  $\widetilde{W}^{ij}(\tilde{c}_{p_i}, \tilde{c}_{p_j}) = dist(\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}, \tilde{c}_{p_j}|_{\mathcal{F}^{p_j}})$
12                  $\tilde{\tau}^{ij}(\tilde{c}_{p_i}, \tilde{c}_{p_j}) = None$
13              else
14                  if $L^{p_i}(\tilde{c}_{p_j}|_{\mathcal{F}^{p_i}}) \in sym(\varphi_c^{p_j})$ then
15                      $\tilde{c}_{p_j} \in \tilde{\delta}^{ij}(\tilde{c}_{p_i})$
16                      $\widetilde{W}^{ij}(\tilde{c}_{p_i}, \tilde{c}_{p_j}) = dist(\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}}, \tilde{c}_{p_j}|_{\mathcal{F}^{p_j}})$
17                      $\tilde{\tau}^{ij}(\tilde{c}_{p_i}, \tilde{c}_{p_j}) = L^{p_i}(\tilde{c}_{p_i}|_{\mathcal{F}^{p_i}})$
18  $R_+^{\varpi,p} = bidirection\_dijkstra(\Omega_{p_j}, \tilde{\eta}_{p_i})$
19  $R_+^{\varpi} = R_+^{\varpi,p} \oplus R_-^{\varpi,s}$
20  return $\widetilde{\mathcal{U}}_{ij}$ and $R_+^{\varpi}$
```

VII. Overall Structure

Given the task specification φ^{p_i} and FTS $\mathcal{F}^{p_i}$ of each agent, we decouple the compact coupled task specification and construct new complete agent model $\mathcal{C}_n^{p_i}$ through applying Algorithm. The initial motion path ϱ^{p_i} of each agent p_i can be synthesized and we plan in speed matching horizon to reduce the complexity of task collaboration. To guarantee the completeness of collaboration task, an online collaboration scheme is adapted to revise the initial motion path of each agent, where the collaboration task is completed through **Request**^{p_i}, **Reply**^{p_i} and **Confirm**^{p_i} messages. Corresponding to the properties of compact coupled task, we propose an agent scheduling system to prevent long-time demission and dispatch agents to strive to complete task when part of agents failure. The agent scheduling system is triggered by specific conditions, such as the ultimate time T_u is reached or **Warning**^{p_i} messages are transmitted through communication network, while the online collaboration system is running.

VIII. Case Study

Consider the logistics system in a depot, there are three groups of agents assigned three different task specifications. For simplicity, denote the agents by $g_k = p_{i,k}, i = 1, 2, ..., \forall k = 1, 2, 3$. The algorithms proposed are implemented and simulated on a laptop (i5-7300HQ, 2.50GHz Duo CPU and 8GB RAM).



(a) environment (b) initial positions

Fig. 1. (a) shows the simulation environment, where the relevant regions of different groups are in different colors and some pivotal regions are labeled. (b) shows the initial positions of agents.

A. Workspace Description

The environment of depot is shown in Fig.1(a) and the initial positions of agents are as shown in Fig.1(b). The blue strips are the conveyor belts in the depot which occupy the space and agents cannot across them, and the regions labeled as ti and mi $(i = 1, 2, ..., 6)$ are checkpoints around the conveyor belts where the agents working on belts need to visit frequently to ensure the belts work properly. Two groups of agents which are silver in Fig.1(b) are responsible for the work on the conveyor belts and need to take care of the switch which is labeled by o and o'. The regions associated with these two groups are painted yellow and green. A group of agents which are red in Fig.1(b) are assigned to collect goods in g, move across the door d and place the goods at the cargo collection point which is labeled r. The regions associated with them are painted gray.

The switches in gray control the door d and when the switch is open, it will change to yellow. Only if o and o' are open simultaneously, the door is open and agents can move across it. If an agent sends request message, it will change to blue, meanwhile, if there is an agent reply the request message, it will change to black and there will be a red line connect them. In Fig.1(b), the agent in r has sent a request message because it intends to move across the door d, and we can see there are two agents turn to black to reply the request message and there are two red lines connect each of them to the request agent.

B. Task Specification

Three different tasks are assigned to three groups of agents. The task of agents working on conveyor belts is specified as $(\Box\Diamond t1 \wedge \Box\Diamond t2 \wedge \Box\Diamond t3 \wedge \Box\Diamond t4 \wedge \Box\Diamond t5 \wedge \Box\Diamond t6 \wedge \Box(\neg b))$ and $(\Box\Diamond m1 \wedge \Box\Diamond m2 \wedge \Box\Diamond m3 \wedge \Box\Diamond m4 \wedge \Box\Diamond m5 \wedge \Box\Diamond m6 \wedge \Box(\neg b))$ which means that they need to always eventually visit all the checkpoints and never move across the conveyor belts. The task of agents which need to collect goods and place them in cargo collection point is specified as $(\Box\Diamond g \wedge \Box(g \Longrightarrow \Diamond r) \wedge \Box(\neg b) \wedge \Box(\neg d \cup (o \wedge o')))$, which means the good collection agents need to always eventually visit g and when

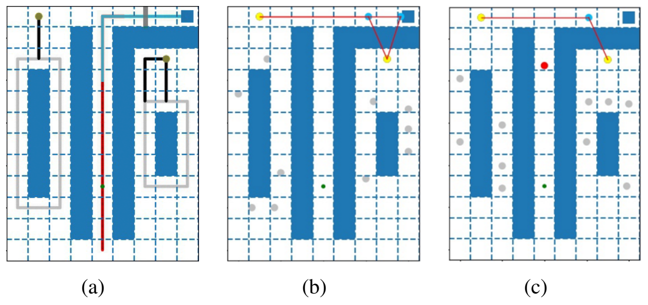


Fig. 2. (a) shows the motion paths of agents, (b) shows the message transmission when two agents are passing through the door and moving to the good collection point and (c) shows the message transmission when two agents are going back to collect goods after they placed the goods.

they collect goods in g, they need go to r to place them. The good collection agents can never move across the belts and only when the switches are both open can they move across the door d. We can see $\Box(\neg r \cup (o \wedge o'))$ is the compact coupled task φ_c because it indicates that only when two switches are both open can the good collection agents move across the door.

C. Simulation Result

The motion path of each agent is shown in Fig.2(a), where different colors indicates different states of agents. The agents which are responding to deliver goods are denoted by g_1, they will change from red to blue when they send request message. Meanwhile, two agents are elected from g_2 and g_3 to reply the request message and change from blue to black. We record the position of agents in each time step and synthesize this motion path of agents.

We select two scenes to analyze the online collaboration scheme. In Fig.2(b), two agents, denoted by p_0 and p_1, in g_1 are passing through the door d to place the goods in good collection point r. Meanwhile, two agents, denoted by p_2 and p_3, in g_2 and g_3 are elected to collaborate. p_0 and p_1 are turned to blue, and p_2, p_3 are turned to black. There are red solid lines connecting p_0 and p_2, p_3, p_1 and p_2, p_3, which indicates they are transmitting message.

In Fig.2(c), p_0 and p_1 have placed the goods in r and they are going back to g to collect goods. p_0 has passed through d so it stop the sending of request message, but p_1 is still passing through d so it continues to send the request message and p_2, p_3 continue to reply to it, so there are red solid lines connecting p_0 and p_2, p_3.

In order to represent the collaboration of agents more intuitively, we draw the Gantt chart as Fig.3. Each horizontal line in Gantt chart demonstrates the change of agent states over time. Same as previous figures, different colors indicates different states of agents. If a section of the horizontal line indicates the agent is sending request messages or replying for the request messages of other agents, which is represented by blue and black, then the indices of agents that is sending request messages or the agent is replying for are identified at the beginning of this section.

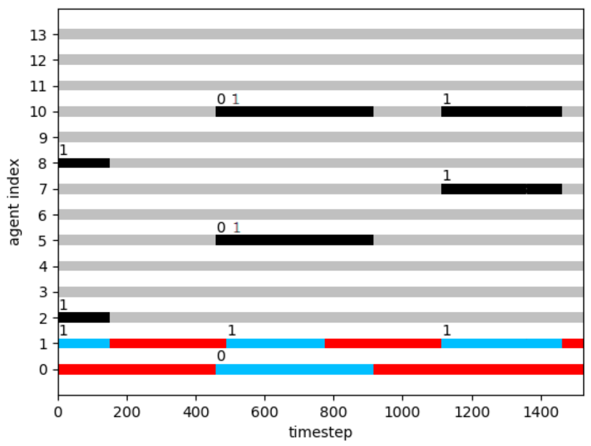


Fig. 3. The gantt chart of agents. Different colors indicate different states of agents. The number at the beginning of blue section indicates the agent sending request messages and the number p_s at the beginning of black section in horizontal line of agent p_t indicates the p_t is replying for p_s.

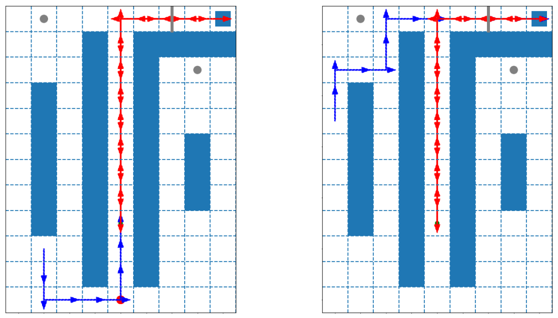


Fig. 4. (a) and (b) show the motion paths of agents which start from different positions to inherit the good collection task.

D. Agents Scheduling When Failure Occurs

In this section we assume one good collection agent would breakdown at random time step. The agents working on belts more than the good collection agents and the good collection task is more important than checking the belts, so when one good collection agent breaks down the agent working on belts need to inherit the good collection task. Fig.4 shows two motion paths of agents which start from different positions to inherit the good collection task. In Fig.4, the invariant suffix is denoted by red arrows and the prefix is denoted by blue arrows.

E. Computational Complexity

In the first simulation, the team has 14 agents and the simulation environment consists of 108 states and 97 edges, the NBA associated with g_1 consists of 8 states and 30 transitions, and the NBA associated with the NBA associated with g_2 and g_3 consists of 7 states and 34 transitions. As a result, the centralized method products all the FTS and NBA of agents in team will have $108^{14} \times 8^6 \times 7^6 (9 \times 10^{38})$ states. In our decentralized method, each agent only has partial environment which consists of 38, 17, 30 states as shown in Fig.1(a). Our method relies on local product of FTS and NBA of each agent and synthesize the initial motion path locally so the complete agent model of g_1, g_2 and g_3 consists of 136, 266 and 210 states, respectively and it costs an average of 0.02s to synthesize the initial motion path for each agent.

We adjust the number of agents and the motion plan is synthesized within the same time, which indicates the computational complexity of our decentralized method only associated with the states of local FTS and NBA of each agent, unlike previous centralized method, whose computational complexity increases exponentially accompanied by the increasing of agent number. The number of states of union agent model depends on the two local complete agent model. In the second simulation, the number of the state number of union agent model is 402 and the maximum number is 476 and it costs an average of 0.035s to synthesize the motion path for the agent to inherit the task of broken agent

IX. Summary And Future Work

Through the simulation results we can see the proposed method can handle the decentralized collaboration of multiagent system under compact coupled task specifications. Proposed decentralized method depends on local FTS and NBA of agent so the computational complexity doesn't increase accompanied by the increasing of agent number, which reduces the computational complexity significantly compared to centralized method. In the future, we will continue to find method to further reduce the computational complexity and combine proposed method with existing control algorithms to improve the overall convergence rate of Cyber-physical system.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

PLACE
PHOTO
HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.