Online Multi-Agent Supervisory Control for Warehouse Automation: Heterogeneous Payloads

Moeto Kasahara and Kai Cai

Abstract— **In this paper we consider a problem of controlling multi-agent discrete-event systems to serve tasks dynamically appearing in the environment, where the agents generally have different capacities (or heterogeneous payloads) to serve different numbers of tasks. To solve this problem, we propose an effective online supervisory control approach combined with optimal task assignment which minimizes the total weighted distances from the heterogeneous agents to multiple tasks. The weighted distance takes into account a realistic aspect that traveling the same distance may take different times depending on whether the agents are purely moving straight or involving making turns. We then apply this online control scheme to model and control a warehouse automation system using multiple mobile robots with heterogeneous payloads; the effectiveness of this scheme is demonstrated on a case study.**

I. Introduction

In [1], [2], [3], [4], [5], we have introduced a problem of controlling multi-agent discrete-event systems (DES) to serve multiple tasks, and extended supervisory control theory (SCT) [6], [7] to provide effective solutions. The study of this problem is motivated by logistic automation systems using a team of autonomous robots. A prominent application is Kiva systems in Amazon's warehouses and distribution centers [8].

The tasks considered in [1], [2], [3] are static: the information of the tasks is completely known at the outset, and no newly added tasks are considered. Moreover, which task is assigned to which robot is assumed to be given *a priori*. For this static setup, [1] shows that the standard SCT may be adapted to compute a safe and deadlock free solution for multiple agents to accomplish multiple tasks. To relieve computational burden, [2], [3] further adapt an online SCT based on a limited-lookahead strategy [9]. In this online approach, a supervisor is recomputed at the occurrence of *every* event.

In [4], [5], a more realistic setup is considered in which tasks can appear dynamically; when and where the tasks appear are unknown at the outset. An extended online supervisory control approach is proposed that recomputes a supervisor when (and only when) there are unassigned tasks and available agents. This online scheme is thus distinct from that in [2], [3], [9], and allows the recomputed supervisor to be adaptive to newly appeared tasks. Furthermore, [5] combines the online approach with optimal task assignment [10] to improve efficiency of serving dynamic tasks: the sum of (unweighted) distances from agents to tasks is minimized.

This work was supported in part by JSPS KAKENHI Grant no. 21H04875. The authors are with Department of Electrical and Information Engineering, Osaka City University. Emails: kasahara@c.info.eng.osaka-cu.ac.jp (M. Kasahara), kai.cai@eng.osaka-cu.ac.jp (K. Cai)

In this paper, we build on and further extend [5] in twofolds, both addressing realistic aspects in warehouse automation systems served by multiple robots. First, we differentiate the time that a robot needs to execute a "go straight" action from a "turn left/right" action. In particular, when turning a robot typically needs to first decelerates, then makes the turn, and accelerates again; hence, a turning action is more time-consuming than a going-straight action. As a result, among multiple paths consisting of similar numbers of actions (similar distances), the one with the fewest number of turning actions is likely the most time efficient. This issue is addressed by introducing to optimal task assignment a weight for encouraging moving straight, and thereby minimizing the sum of weighted distances.

Second, we consider the general case where robots may have different payloads, so that they can be assigned with different numbers of tasks. This case is addressed by introducing virtual robots to optimal task assignment to match up the number of robots' payloads and the number of tasks. Moreover, when a robot is assigned to serve multiple tasks, the order of serving these tasks may be optimized to further improve efficiency. This latter optimization is addressed by solving a traveling salesman's problem.

In the sequel, we present the extended online supervisory control approach that effectively address the above two new issues, and illustrate the approach by a warehouse automation case study.

II. Preliminaries

A. Supervisory control basics

In SCT [6], [7], the plant to be controlled is modeled by a finite state *automaton*

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is the finite state set, $q_0 \in Q$ the *initial state*, $Q_m \subseteq Q$ the set of *marker states*, Σ the finite *event set*, and $\delta : Q \times \Sigma \rightarrow Q$ the (partial) *state transition function*. We extend δ such that $\delta : Q \times \Sigma^* \rightarrow Q$, and write $\delta(q,s)!$ to mean that $\delta(q,s)$ is defined. The event set Σ is partitioned into a subset Σ_c of *controllable* events and a subset Σ_u of *uncontrollable* events; only controllable events can be enabled or disabled by an external entity, called *supervisor*, introduced below.

The closed behavior of $\mathbf{G}$ is the set of all strings that can be generated by $\mathbf{G}$:

$$L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\} \subseteq \Sigma^*.$$

On the other hand, the *marked behavior* of $\mathbf{G}$ is the subset of strings that can reach a marker state:

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G}).$$

$\mathbf{G}$ is *nonblocking* if $L(\mathbf{G}) = \overline{L_m(\mathbf{G})}$ ($\bar{\cdot}$ means *prefix closure*), namely every string in the closed behavior may be completed to a string in the marked behavior.

A language $E \subseteq \Sigma^*$ is said to be *controllable* (with respect to $\mathbf{G}$) if $\overline{E}\ \Sigma_u \cap L(\mathbf{G}) \subseteq \overline{E}$. Let $K \subseteq L_m(\mathbf{G})$ be a specification language imposed on the plant $\mathbf{G}$. Denote by $C(K)$ the family of controllable sublanguages of K, i.e.

$$C(K) := \{K' \subseteq K | \overline{K'}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K'}\}.$$

Then the supremal controllable sublanguage of K exists and is given by $\sup C(K) = \cup\{K' | K' \in C(K)\}$. Let $\mathbf{SUP}$ be a (nonblocking) automaton such that $L_m(\mathbf{SUP}) = \sup C(K)$. We call $\mathbf{SUP}$ the supervisor for plant $\mathbf{G}$ that enforces $\sup C(K)$. The control action of $\mathbf{SUP}$ after an arbitrary string $s \in L(\mathbf{G})$ is to enable an event in the following set

$$\gamma(s) := \{\sigma \in \Sigma_u | s\sigma \in L(\mathbf{G})\} \cup \{\sigma \in \Sigma_c | s\sigma \in L(\mathbf{SUP})\}. \quad (1)$$

B. Optimal task assignment

The task assignment problem is an optimization problem of finding a one-to-one correspondence between an agent and a task so as to minimize the sum of the costs included in the assignment. Consider n agents, n tasks, and $c_{i,j}$ for each pair of $i, j \in \{1, \ldots, n\}$ is the cost when agent i is assigned to serve task j. Then the task assignment problem is formulated as follows.

$$\begin{array}{ll} \text{minimize} & z = \sum_{i=1}^n \sum_{j=1}^n x_{i,j} c_{i,j} \\ \text{subject to} & (\forall j \in \{1, \ldots, n\}) \sum_{i=1}^n x_{i,j} = 1 \ \& \\ & (\forall i \in \{1, \ldots, n\}) \sum_{j=1}^n x_{i,j} = 1 \ \& \\ & (\forall i, j \in \{1, \ldots, n\}) x_{i,j} \in \{0, 1\} \end{array}$$

The (indicator) variable $x_{i,j}$ is equal to 1 when agent i is assigned to serve task j and 0 otherwise.

To find the optimal task assignment is NP-hard, although there are many polynomial time algorithms available to compute approximate solutions. We shall employ the well-known Hungarian algorithm (or Kuhn-Munkres algorithm) whose time complexity is $O(n^3)$ [10].

Remark 1. While the above formulation of the task assignment requires that the number of agents be the same as the number of tasks, the more general case where the numbers are different can be easily addressed. If (without loss of generality) the number of agents is greater than the number of tasks, we simply need to add 'dummy' tasks to match up the numbers and these 'dummy' tasks should have significantly high costs so that they will never be chosen.

III. Online multi-agent supervisory control for heterogeneous payloads

Consider N agents and each agent $i (\in \{1, \ldots, N\})$ has payload $p_i (\geq 1)$. View each agent i as having p_i copies of itself, thus each copy being of payload 1. Think of these copies as virtual agents; then there are $n := \sum_{i=1}^N p_i$ virtual agents each having payload 1. As mentioned in Remark 1, it is without loss of generality to consider that there are also n tasks to be assigned.

Now we present the new online supervisor control approach that addresses heterogeneous agents' payloads.

(1) Use the Hungarian algorithm [10] to compute an optimal task assignment such that each agent $i (\in \{1, \ldots, N\})$ obtains a number of tasks equal to its payload.

(1-1) Compute the costs $c_{i,j}$ for all $i, j \in \{1, \ldots, n\}$ (here we consider virtual agents).

(1-2) Compute an optimal task assignment such that each virtual agent $i (\in \{1, \ldots, n\})$ obtains one task.

(1-3) Designate for each real agent the tasks assigned to its virtual copies.

(2) Compute for each agent $i (\in \{1, \ldots, N\})$ the shortest paths for accomplishing the assigned tasks. This is done by solving for each agent i a traveling salesman's problem.

(3) Create the finite state automata $\mathbf{G}_1, \ldots, \mathbf{G}_N$ based on the shortest paths of each agent.

(4) Create a control specification model $\mathbf{SPEC}$ (also a finite state automaton) that imposes a behavioral constraint on the multi-agent system.

(5) Based on the agent models $\mathbf{G}_1, \ldots, \mathbf{G}_N$ and the specification model $\mathbf{SPEC}$, compute by the standard SCT [7] a supervisor $\mathbf{SUP}$. This $\mathbf{SUP}$ ensures safe (i.e. the specification is satisfied) and nonblocking controlled behavior.

(6) Return to (1) whenever there are (newly appeared) unassigned tasks *and* available agents (currently serving no tasks) for assignment.

IV. Case study

We demonstrate how to apply the proposed online multi-agent supervisory control procedure to model and control a warehouse logistic system automated by multiple mobile robots with heterogeneous payloads.

A. Warehouse Environment

Different warehouses have different configurations. For a concrete case study, we consider the grid-type layout as displayed in Fig. 1. Mobile robots are assumed to be initially waiting for tasks at the top area, items to be picked up stored on storage shelves in the black-rectangle areas, and item-delivery destination locations at the bottom.

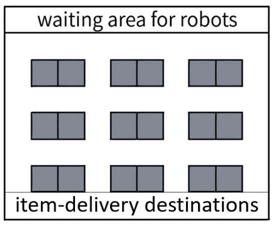

Fig. 1. Warehouse grid enviroment: items to be picked up are stored in black-rectangle areas

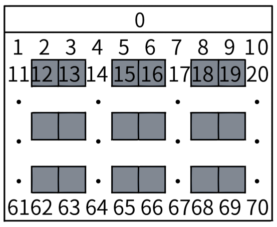

Fig. 2. Warehouse grid assigned with numbers

The occurrence of tasks is uncontrolled: when and where they occur are completely random, and tasks are managed by a queue structure. We assume that the number of tasks never exceeds the capacity of the queue (or otherwise discarded), and that each robot can be assigned with a number of tasks (with the same delivery destination) not exceeding its payload.

All robots initially wait in the top waiting area and only move when they are assigned tasks. Robots are allowed to enter the storage areas only when they are fetching their assigned items. After retrieving the items, the robots move to their designated delivery destination areas at the bottom.

If a robot completes all its assigned tasks, it will be either assigned to serve new tasks (if those have appeared and not yet served) or controlled to return to the top waiting area.

B. Automata Models of Robots

We start by assigning sequential (state) numbers to the warehouse as shown in Fig. 2. Specifically, we assign 0 to the waiting area, and the other areas (or cells) are natural numbers starting from the top left corner. When tasks are assigned to a robot, one of the delivery areas numbered $61,\dots,70$ will be the marker (or goal) state. In the case where no task is assigned to a robot (number of tasks is relatively small as compared to number of robots with different payloads), 0 is the robot's marker state. Each robot shall move in one of the four directions: up, down, left and right. All robots are assumed to be initially located in the waiting area and eventually return to the waiting area after finishing all assigned tasks.

Consider $N(>1)$ robots serving the warehouse. Each of these N robots is modeled by a finite state automaton $\mathbf{G}_i$ ($i \in \{1,\dots,N\}$):

$$\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i}).$$

Here Q_i is a set of states on the paths of robot i (using the numbers assigned to the warehouse as in Fig. 2). Σ_i is a set of four events, given in Table I. All events are assumed to be controllable. δ_i is the state transition function defined according to the paths of robot i; $q_{0,i}$ is the initial state (i.e. the starting point of the robot's paths); and $Q_{m,i}$ is the set of marker states (i.e. the ending points of the robot's paths).

The paths needed to obtain the automaton model $\mathbf{G}_i$ are computed as follows. There are two cases.

(Case 1) When robot i is assigned with several tasks, compute the shortest paths from its current location to each of the task locations exactly once and finally to the destination. This is done by solving a traveling salesman's problem. In this case, $q_{0,i}$ is the state number of the robot's current location, and $Q_{m,i}$ is the singleton subset of state numbers $61,\dots,70$, which is the destination area of the items' delivery.

(Case 2) When a robot finishes all of its assigned tasks and is not assigned with any new task, compute the shortest paths from the robot's current position to the top waiting area. In this case, $q_{0,i}$ is the state number of the robot's current location, and $Q_{m,i}$ is the singleton set $\{0\}$.

TABLE I
EVENT NUMBERS OF EACH ROBOT $i \in \{1,...,N\}$

go up	$i\times 10+1$
go right	$i\times 10+3$
go down	$i\times 10+5$
go left	$i\times 10+7$

C. Example of Three Robots

Consider six tasks $T_1,\dots,T_6$ to be assigned and three robots R_1,R_2,R_3 with payloads 3, 2, and 1 respectively. The initial arrangement of the robots and tasks is shown in Fig. 3. All robots are initially in state 0, and the tasks are located at 12, 16, 33, 36, 52, and 59.

The first step of the proposed online supervisory control is to optimally assign the six tasks to the three robots according to their respective payloads. For this, we consider six virtual robots (three for R_1, two for R_2, one for R_3) and generate the 6×6 cost matrix C as follows:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} \\ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} & c_{4,6} \\ c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & c_{5,5} & c_{5,6} \\ c_{6,1} & c_{6,2} & c_{6,3} & c_{6,4} & c_{6,5} & c_{6,6} \end{bmatrix}$$
$$= \begin{bmatrix} 2 & 8 & 11 & 12 & 12 & 17 \\ 2 & 8 & 11 & 12 & 12 & 17 \\ 2 & 8 & 11 & 12 & 12 & 17 \\ 9 & 5 & 10 & 7 & 13 & 10 \\ 9 & 5 & 10 & 7 & 13 & 10 \\ 8 & 2 & 11 & 10 & 14 & 13 \end{bmatrix}$$

The entries of C are the weighted distances between the robots and the items, where a go-straight action has weight 1 and a turn action has weight 2. Note that depending on the orientation (or direction of movement) of each robot, any of its four events can be a moving-straight action or a turning action; thus the weight considered here is not directly associated with the events. Having matrix C, we apply the Hungarian algorithm to derive

$$C^* = \begin{bmatrix} 0 & 6 & 0 & 4 & 0 & 6 \\ 0 & 6 & 0 & 4 & 0 & 6 \\ 0 & 6 & 0 & 4 & 0 & 6 \\ 8 & 4 & 0 & 0 & 2 & 0 \\ 8 & 4 & 0 & 0 & 2 & 0 \\ 6 & 0 & 0 & 2 & 2 & 2 \end{bmatrix}$$

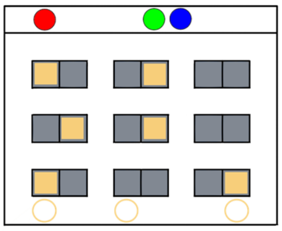

Fig. 3. Initial arrangement of the three robots (colored discs: red R_1, blue R_2, green R_3) and the six tasks (yellow squares: items; yellow circles: destinations)

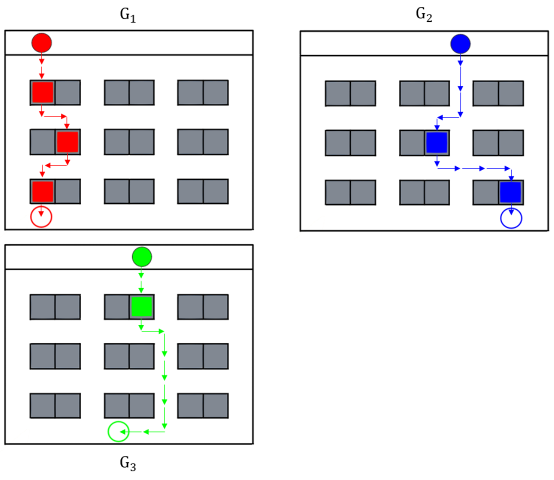

Fig. 4. Shortest paths for the three robots to serve all assigned tasks

Observe that in C^* one can select 0 from each row and each column without duplication: namely $(1,1),(2,3),(3,5),(4,4),(5,6),(6,2)$. These 0-entries yield an optimal task assignment: T_1, T_3, T_5 are assigned to the three virtual copies of R_1 (hence to R_1); T_4, T_6 to the two virtual copies of R_2 (hence to R_2); and T_2 to R_3.

The second step is to compute the shortest paths for each of the three robots from its initial location to all the item locations and finally to the destination. These shortest paths are computing by solving a traveling salesman's problem for each robot,[1] and the resulting paths are displayed in Fig. 4.

In Step 3, we create the automata models for the three robots based on the above computed shortest paths. Let the automaton of R_1 be $\mathbf{G}_1$, the automaton of R_2 be $\mathbf{G}_2$, and the automaton of R_3 be $\mathbf{G}_3$, respectively.

As the control specification, in Step 4 we impose mutual exclusion on each cell of the grid so that the robots do not collide with one another (i.e. safety). One exception is the waiting area state 0: we assume that this area is large enough

[1]We note that the traveling salesman's problem considered here is a variation of the standard one, in that we require that the destination area be the last location to visit. This variation may be readily solved using standard algorithms by appropriately choosing the weight between the initial waiting area and the destination.

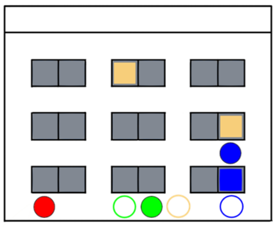

Fig. 5. Robot R_1 (red disc) finishes its assigned three tasks and two new tasks appear (yellow squares); while the other two robots are still en route to finish their assigned tasks

such that multiple robots can be at state 0 at the same time.

In Step 5, we employ the standard SCT to compute a supervisor that satisfies the safety control specification. The resulting supervisor is guaranteed to be not only safe but also nonblocking (the latter ensures that no deadlock ever occurs and all tasks are eventually accomplished).

The computed supervisor executes its control decisions online according to (1). When robot R_1 finishes delivering its assigned three items to the destination, as shown in Fig. 5, it becomes available again to serve new tasks. Suppose that there have appeared two new task at cells 15 and 39; then the online procedure will return to Step 1 and compute a new supervisor to respond to these newly appeared tasks (in this case assigning them both to robot R_1, as the other two robots are still en route to accomplish their assigned tasks).

REFERENCES

[1] Y. Tatsumoto, M. Shiraishi, and K. Cai, "Application of supervisory control theory with warehouse automation case study," *Trans. ISCIE*, vol. 62, no. 6, pp. 203–208, 2018.

[2] M. Shiraishi, Y. Tatsumoto, K. Cai, and Z. Lin, "Online supervisory control of multi-agent discrete-event systems with warehouse automation case study," in *Proceedings of the SICE Annual Conference*, 2018, pp. 1059–1062.

[3] Y. Tatsumoto, M. Shiraishi, K. Cai, and Z. Lin, "Application of online supervisory control of discrete-event systems to multi-robot warehouse automation," *Control Engineering Practice*, vol. 81, pp. 97–104, 2018.

[4] K. Cai, "Warehouse automation by logistic robotic networks – a cyber-physical control approach," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, pp. 693–704, 2020.

[5] M. Kasahara and K. Cai, "Online supervisory control with optimal task assignment for efficient and adaptive warehouse automation," in *Proc. the 63rd Japan Joint Automatic Control Conf.*, 2020, pp. 90–93.

[6] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[7] W. M. Wonham and K. Cai, "Supervisory Control of Discrete-Event Systems," *Springer*, 2019.

[8] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–19, 2008.

[9] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1921–1935, 1992.

[10] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.