Robotica (2012) volume 30, pp. 721–730. © Cambridge University Press 2011
doi:10.1017/S0263574711000920

Dynamic task allocation in cooperative robot teams

Athanasios Tsalatsanis†,* Ali Yalcin‡ and Kimon. P. Valavanis§

†*University of South Florida, Tampa, FL, USA*
‡*Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, USA*
§*Department of Electrical and Computer Engineering, University of Denver, Denver, CO, USA*

(Accepted July 7, 2011. First published online: August 17, 2011)

SUMMARY
In this paper, a dynamic task allocation and controller design methodology for cooperative robot teams is presented. Fuzzy-logic-based utility functions are derived to quantify each robot's ability to perform a task. These utility functions are used to allocate tasks in real time through a limited lookahead control methodology partially based on the basic principles of discrete event supervisory control theory. The proposed controller design methodology accommodates flexibility in task assignments, robot coordination, and tolerance to robot failures and repairs. Implementation details of the proposed methodology are demonstrated through a warehouse patrolling case study.

KEYWORDS: Supervisory control; Cooperative robot teams; Task allocation; Limited lookahead policy.

1. Introduction

Many applications in industrial, civilian, and military fields benefit from mobile robot utilization. Application domains vary from warehouse patrolling to service robotics and to space exploration. Mobile robots have been reported to explore, map or inspect friendly or hostile territories,[1–3] or dispense medications in medical facilities.[4] Specifically in industrial applications, such as manufacturing, underground mining, toxic waste cleanup and material storage/handling, where many processes take place in hazardous environments harmful to human health, the choice of robotics-based solutions is justifiable. Furthermore, as the complexity and requirements of an application increases, significant advantages may be drawn from the use of multirobot systems.

A major challenge when working with multirobot systems is that of task allocation and coordination. The overall mission is decomposed into multiple tasks to which one or more robots are assigned. The task allocation problem is further complicated considering the dynamic characteristics of the robot team such as robot failures and repairs that may lead to incomplete tasks. The robot team should be able to complete the mission even if some team members are no longer operational and/or available.

This paper describes a general dynamic task allocation and controller design methodology for cooperative robot teams. The robot team is modeled as a Discrete Event System (DES). Each robot is modularly represented by a finite state automaton model. The mission requirements model is synthesized from individual finite state automata representing task completion requirements.

The proposed control methodology is partially based on the Ramadge and Wonham (RW) supervisory control theory.[5] However, instead of synthesizing a complete supervisor, as the traditional RW theory suggests, a limited lookahead policy is adopted that enables/disables events in the system in real time based on the evaluation of a utility function and robot availability. The utility function uses fuzzy logic to quantify the ability of a robot to perform a task. The robot modules appear or disappear overtime depending on robot failures and repairs. When a failure event occurs, the control methodology reallocates tasks to the operational robots of the team to ensure mission completion.

In cooperative robot teams, several characteristics of the team members, such as endurance, reliability, and efficiency, must be considered in task allocation decisions. We describe these characteristics as fuzzy variables and develop a fuzzy controller to determine the *utility function value* for each task allocation event. These values are then used to determine a preferred task allocation in real time as a part of the proposed controller design methodology.

Our work is motivated first by the fact that in traditional supervisory control theory, the acceptable sequences of event execution determined *a priori* are computationally intractable for realistic size problems. Furthermore, in applications where there is a significant degree of uncertainty associated with resource reliability and the environment, these sequences may not be executable. Instead, we propose a control approach based on a limited lookahead control policy for task allocation in real time.

Second, in supervisory control theory, the criteria used in restricting the state-to-state transition of the system are the marked states, which denote the acceptable states, and the legal language, which denotes acceptable system behavior. These criteria fail to describe a preferred behavior within the acceptable behaviors.

This work combines the modeling strengths of DES and supervisory control theory to model comprehensive intertask dependences and robot interactions applicable to most task allocation problems. Specifically, this works demonstrates flexibility in task assignments, task sequencing, robot cooperation and coordination, and tolerance in robot failures and repairs. In addition, given that there is diversity between the robots' capabilities (heterogeneous robot team), the proposed methodology can be applied to missions where each task presents distinct requirements.

* Corresponding author. E-mail: atsalats@health.usf.edu

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

The rest of this paper is organized as follows: Section 2 discusses the related literature. Section 3 presents the DES models of the robot team and mission requirements models for the task allocation problem. Section 4 describes the utility function concept, the fuzzy controller used to determine the utility function values for task allocation events and the proposed limited lookahead policy. Section 5 introduces failures and repairs in the proposed model. Section 6 discusses scalability issues for the proposed methodology. Section 7 presents simulation results using a case study, and finally, Section 8 concludes this paper.

2. Related Literature

The task allocation problem has been addressed in literature by utility-based approaches and auction-based approaches for both cooperative robot teams and robot swarms. Utility-based approaches have been used for task allocation in many control approaches.[6–14] Each task is assigned to a robot based on various utility estimates, such as *acquiescence* and *impatience*,[6] sensor relevance,[7] net energy,[9] positive and negative feedback assigned to the robot based on historical performance,[10] coalition values,[11] belief on the ability of the robot to perform a task,[13] and costs.[14] Utility has also been used in robot team cooperation to estimate the cost of executing an action[15] and for sensor-based metrics.[16]

Auction-based approaches[15–26] achieve task allocation based on the Artificial Intelligence concept of Contract Net Protocol.[27] Each robot *bids* for an available task and the robot with the higher bid is assigned to that task. In the proposed control methodology, the dynamic task allocation problem is addressed using utility and fuzzy logic. Utility function values are computed based on the ability of each robot to perform a task considering several factors.

Limited lookahead policies for supervisory control have been first studied by Chung *et al.*[28] where a limited lookahead window is used to control the online behavior of the uncontrolled system model (UCSM). The notion of *pending traces* is introduced to describe the legality of a trace in the lookahead window based on a conservative or an optimistic attitude. The notion of pending traces was later raised by Kumar *et al.*[29] by extending the UCSM behavior by arbitrary traces beyond the limited lookahead window. Chung *et al.*[30] present a methodology that recursively computes the future control actions based on previously computed control actions. Later, Chung *et al.*[31] and Hadj-Alouane *et al.*[32] presented an extension to the lookahead policies to cope with the computational complexity problem by making a control decision without exploring the whole lookahead window.

Further enhancements in limited lookahead policies for supervisory control have been proposed. In work by Heymann *et al.*,[33] a lookahead policy is presented for systems with partial observability. Also, in work by Kumar and Garg,[34] system's uncertainty is considered by assigning probabilities to event occurrences and in work presented by Lin[35] by modeling all possible variations of the system. Winacott and Rudie[36] designed a lookahead methodology that incorporates probabilistic event occurrences and desirability functions for strings of events. Millan and O'Young[37] present a hybrid model that utilizes limited lookahead windows to build nonblocking supervisors. Finally, Auer *et al.*[38] extend the limited lookahead control policy to enable concurrency scenarios. To our knowledge, there are no limited lookahead policies in the literature designed to control cooperative robot teams.

As noted Grigorov and Rudie,[39] only few approaches, as described by Yi-Liang *et al.*[40] and Gordon and Kiriakidis,[41] concentrate in time varying systems where system modules appear or disappear in time. In these approaches, resource modules disappear only after the completion of assigned tasks. In this work, we relax this assumption by considering failures during task execution. In coordinated robot teams, the concept of robot failures and repairs is important since a robot failure while executing a task will lead to an incomplete mission unless the control model reassigns the task elsewhere. The lookahead policy presented in this paper considers robot failures and repairs to ensure mission completion.

Supervisory-control-based approaches on DES have been used by a number of researches to control mobile robot teams. However, although a limited amount of work considers robot failures, not much effort is found in the area of control decisions concerning robots rejoining the robot team after repairs. Specifically, the automata-based approaches[42–44] consider situations where some robots go offline but do not take into account situations where robots come back online. Similarly, the Petri Net controller by Jamie *et al.*[45] disregards robot repairs. Finally, the control architecture presented by Kimura *et al.*[46] handles only robot failures. This work is a continuation to the work presented by Tsalatsanis *et al.*,[14,47] which models robot failures and repairs as uncontrollable events in the finite automata (FA) models representing each robot.

The contributions of this work can be summarized as follows:

(1) Modular design that allows for consideration of robot failures and repairs in the system's behavior. It should be noted that most work in the literature of task allocation addresses only cases of robot failures.
(2) Modeling of comprehensive intertask dependencies and robot interactions.
(3) Development of a fuzzy-logic-based utility function to quantify the ability of a robot to perform a set of tasks. The fuzzy logic controller takes into account several practical aspects of robot operation to suggest which robot is appropriate for which task.
(4) Task allocation that relies not only on the concept of acceptable behavior rather the concept of preferred system behavior.

The weakness of this work is the increasing state size due to FA representation. The proposed methodology is applicable to medium-size robot teams (two to seven robots) in realistic size missions (two to nine tasks).

3. Robot Team Models

In RW supervisory control theory, the UCSM and the mission requirements are separately modeled using FA models. The FA model $G_j = (\Sigma_j, Q_j, q_{j0}, \delta_j, Q_{jm})$ represents the

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

Table I. List of states, events, and their description.

State/event	Type of event	Description
I_j	-	Robot j in idle state
B_jk	-	Robot j in busy state with Task k
F_jk	-	Robot j in failed state while executing Task k
start_jk	Controllable	Initiation of Task k by Robot j
complete_jk	Uncontrollable	Completion of Task k by Robot j
failure_jk	Uncontrollable	Failure of Robot j while executing Task k
repair_jk	Uncontrollable	Repair of Robot j
drop_jk	Controllable	Repair of Robot j and reinitialization of Task k

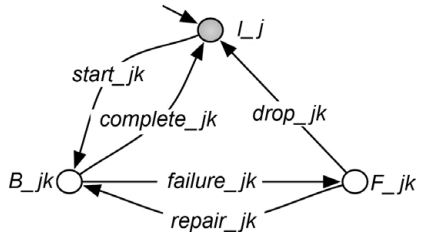


Fig. 1. Transition graph for Robot j.

uncontrollable behavior of Robot j. Σ_j is the set of events, Q_j is the set of states, and $\delta_j : \Sigma_j \times Q_j \rightarrow Q_j$ is the transition function. q_{j0} and Q_{jm} are the initial and final states, respectively. The set of events Σ_j consists of the controllable and the uncontrollable events $\Sigma_j = \Sigma_{jc} \cup \Sigma_{ju}$ and $\Sigma_{jc} \cap \Sigma_{ju} = \emptyset$.

Figure 1 depicts the transition graph for the automaton describing Robot j. An arrow marks the initial state and the marked state is shown as a dark circle. Regarding a Task k, the states and the events of Robot j are defined as shown in Table I.

The FA model design incorporates two different cases of robot failures and repairs. A robot failure may be considered as (i) *temporary failure* or (ii) *failure with task reinitialization*. In the first case, the failed robot will continue executing its task as soon as is repaired while in the second case the task of the failed robot needs to be reinitialized.

The UCSM that represents the uncontrolled behavior of the robot team is composed of the synchronous product[48] of the individual robot modules as follows:

$$G = (\Sigma, Q, q_0, \delta, Q_m) = G_1||G_2||G_3. \qquad (1)$$

The *specification's model* is the finite automaton that models the mission requirements that is synthesized using individual task completion requirement models. Four alternative task completion requirements are modeled:

Alternative 1: Task k can be performed only by Robot j.
Alternative 2: Flexibility in task assignments: Task k can be performed by Robot j or by Robot $i, i \neq j$.
Alternative 3: Task sequencing and robot coordination: Task k must be performed first by Robot j and subsequently by Robot $i, i \neq j$.
Alternative 4: Robot cooperation: Task k must be performed by Robots i and j simultaneously.

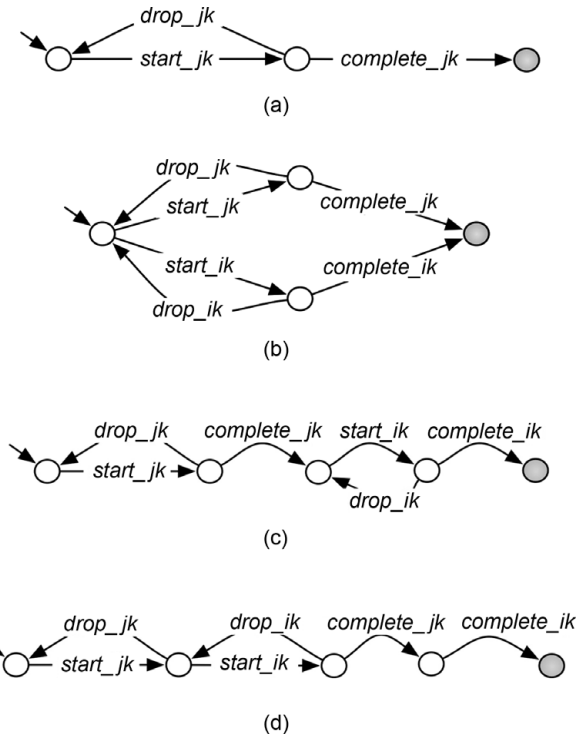


Fig. 2. Transition graphs for the task completion requirements (a) Alternative 1; (b) Alternative 2; (c) Alternative 3; (d) Alternative 4.

Through the task completion requirements, it is possible to model intertask dependences and robot interactions. Figure 2 depicts the transition graphs for these four alternative task completion requirements. Figure 2(a) describes the requirement where Task k can be performed only by Robot j. If Robot j fails during the task, the task may be reinitialized as shown by the *drop_jk* event. Figure 2(b) describes *flexibility in task assignment* where Task k can be performed by Robot j or Robot i. Figure 2(c) describes *task sequencing and robot coordination* where Task j must be completed first by Robot j and then by Robot i. Consider the scenario where two robots, e.g., Robots 1 and 2, must perform one task, e.g., Task 1. To demonstrate, assume that Task 1 is to inspect an area for fire and chemical spills and that Robot 1 carries a fire detection sensor and Robot 2 carries a chemical sensor. To avoid damaging any of the robots, it would be logical to first inspect for fire using Robot 1 and then for chemical spills with Robot 2. Finally, Fig. 2(d) describes a robot cooperation procedure where two robots have to perform a task simultaneously.

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

Formally, task completion requirements are modeled as FA of the form

$$R_n = (\Sigma_n, X_n, \xi_n, x_{n0}, X_{nm}), \tag{2}$$

where n is the number of the individual task completion requirements.

The specification's model, S, is a finite automaton synthesized by the synchronous product of all mission requirements. The specifications model is synthesized as follows:

$$S = (\Sigma, X, \xi, x_0, X_m) = R_1||...||R_n. \tag{3}$$

The supervisory control model for the team of robots consists of the coupled system model S/G and the control pattern Ψ. The coupled model is defined as the product of the UCSM and the specifications model, which includes all the events that are allowed by both models

$$S/G = (\Sigma_S \cap \Sigma_G, X \times Q, \gamma = \xi \times \delta, (q_{0S}, q_{0G}), X_m \times Q_m). \tag{4}$$

The control pattern Ψ is a function $\Psi : \Sigma \times X \rightarrow (0, 1)$ based on the supremal-controllable language of the coupled model that enables (1) or disables (0) the controllable events in the UCSM so that desirable system behavior is guaranteed. The synthesis of the control pattern, and consequently, the solution to the supervisory control problem is a computationally prohibitive procedure for larger systems. Furthermore, while a task allocation and desired control pattern determined *a priori* may be executable with reliable resources in controlled environments, such a sequence of events is very unlikely to be executed to completion in applications associated with cooperative robot teams due to unreliable resources and uncontrollable, and possibly hostile, environments that robot teams typically operate in.

In this paper, instead of following the traditional supervisory control approach and synthesizing the complete supervisor for the system, a limited lookahead control policy is adopted. Limited lookahead control approaches are suitable for highly dynamic systems since only a portion of the system corresponding to the system's behavior in the near future is considered for evaluation. A limited lookahead window of finite depth is used to direct the behavior of the system. Every time an event is executed in the UCSM, the lookahead window is reconstructed and all possible sequences of events in the lookahead window are evaluated. The event leading to the highest evaluated string is enabled while the rest of the controllable events are disabled. The evaluation criteria based on the utility concept are described in the next section.

4. Definition of Utility Function

In heterogeneous cooperative robot teams, each robot possesses unique characteristics including but not limited to sensory capabilities, cost, efficiency, and endurance. Each robot presents a different level of ability to perform a certain task. A function can capture the robot's ability to perform a task in terms of utility. In addition, the same utility function may be used to capture qualitative concepts such as the system designer's choice/knowledge to assign certain tasks to specific robots. For example, consider the case where the system's designer knows that between two robots that can perform the same task, one of the robots will perform better than the other. Obviously, the designer will wish to assign the particular task to the robot that performs better. These aspects complicate the task allocation problem.

An evaluation method that maximizes the overall performance of the robot team is required. In supervisory control theory, traditional system evaluation criteria are the marked states, which denote the acceptable states, and the legal language, which denotes acceptable system behavior. However, these criteria fail to describe undesirable yet acceptable behavior.

The proposed control methodology employs a *utility function* to evaluate strings in the lookahead window. We define a utility function $u : \Sigma \rightarrow [0, 1]$, which associates an event $\sigma \in \Sigma$ with a utility value between 0 and 1 and we define the utility of a string s as

$$U(s) = \sum_{\sigma \in s} u(\sigma). \tag{5}$$

The attributes mentioned that could be used to compute the robot's ability to perform a task, such as endurance, efficiency, and designer's choice, represent vague concepts hard to describe mathematically. However, these concepts can be described in terms of fuzzy logic as fuzzy variables with linguistic membership functions. A Mamdani type fuzzy logic controller[49] is proposed that receives as inputs the membership of each fuzzy variable and computes the ability of a robot to perform a task.

Depending on the mission's functional requirements and the operational characteristics of each robot in the team, the design of the fuzzy controller can include multiple attributes. Such attributes are defined by the system's designer and are limited only by the level of specificity the designer wishes to apply in the task allocation. Examples of these attributes are the robot's endurance and efficiency, the designer's preference in task allocation, the cost of the robot, the number and the type of sensors a robot carries, the distance a robot has to travel to perform a task, and the cost of assigning a robot to a task.

To demonstrate the operation of the fuzzy controller, three fuzzy variables are considered in this study: (i) *robot's endurance*, (ii) *designer's choice*, and (iii) *robot's efficiency*. The first fuzzy variable has three membership functions {*short*, *fair*, *long*} and denotes how long a robot can remain functional. The second fuzzy variable with three membership functions {*low*, *medium*, *high*} denotes the system designer's choice to assign certain tasks to specific robots. Finally, the third fuzzy variable with three membership functions {*low*, *medium*, *high*} denotes the robot's efficiency level. The output of the fuzzy logic controller is also a fuzzy variable with membership functions {*low*, *medium*, *high*} denoting a robots ability to perform a task.

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

Considering a Task k and a Robot j, the ability of Robot j to perform Task k, denoted by $ability_{jk}$, is computed based on a set of rules such as:

- ◇ *If the robot's endurance is* ***long****, the designer's choice is* ***high*** *and the robot's efficiency is* ***high****, then the* $ability_{jk}$ *of Robot j to perform Task k is* ***high****.*
- ◇ *If the robot's endurance is* ***fair****, the designer's choice is* ***medium*** *and the robot's efficiency is* ***medium****, then the* $ability_{jk}$ *of Robot j to perform Task k is* ***medium****.*
- ◇ *If the robot's endurance is* ***short****, the designer's choice is* ***low*** *and the robot's efficiency is* ***low****, then the* $ability_{jk}$ *of Robot j to perform the task k is* ***low****.*

The ability of a robot to perform a task is closely related to task allocation and consequently to task initiation events $\{start_jk\}$. Thus, the utility function value of the events $\{start_jk\}$ is equal to the ability of Robot j to perform Task k. In other words,

$$u(\sigma) = ability_{jk}, \text{ where } \sigma \in \{start_jk\}. \tag{6}$$

The events $\{drop_jk\}$ corresponding to task reinitialization due to robot failure represent cancelation of task assignments. The utility function values of the $\{drop_jk\}$ events are

$$u(\sigma) = -ability_{jk}, \text{ where } \sigma \in \{drop_jk\}. \tag{7}$$

The higher the utility function value for an event, the more desired this event is. Since uncontrollable events $\{complete_jk, failure_jk, repair_jk\}$ cannot be enabled/disabled, their utility function values are

$$u(\sigma) = 0, \quad \sigma \in \{complete_jk, failure_jk, repair_jk\}. \tag{8}$$

Even though the uncontrollable events $failure_jk$ and $repair_jk$ have utility function values equal to zero, occurrences of such events influence indirectly the string utility. The impacts of such events to the controller design are discussed in the next section.

Each time an event σ occurs in the UCSM and the limited lookahead window is reconstructed, the utility function values for all the strings $s \in L(S/G)$ (the language of the coupled model) in the lookahead window are computed. The string with the highest utility function value corresponds to the most desirable system behavior. The maximization procedure is implemented as a dynamic programming problem[50] with a forward sweep and a backtracking pass.

A string that includes many task initiation events is evaluated higher than a string with fewer task initiation events. However, the highest evaluated string may not always correspond to the most desirable task allocation. Consider the case where three robots (Robots 1, 2, and 3) are assigned to perform three tasks (Tasks 1, 2, and 3). If $u(start_11) = u(start_22) = u(start_34) = 1$ and $u(start_31) = 2$ indicating that assigning Task 1 to Robot 3 is the desired action since $u(start_31) > u(start_11)$.

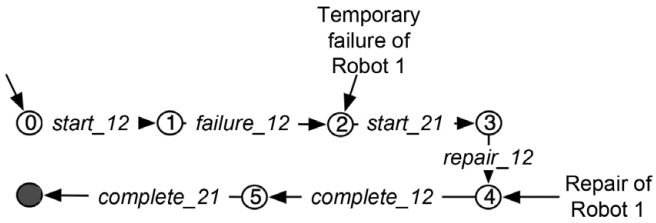


Fig. 3. Events executed in the UCSM after a temporary failure.

Suppose that the event $start_11$ (initiation of Task 1 from Robot 1) is a part of the string $start_11start_22start_34$ and the event $start_31$ (initiation of Task 1 from Robot 3) is a part of the string $start_31complete_31complete_jk$ in a limited lookahead window with depth 3, where only three future events are considered. The utility function value for the string $start_11start_22start_34$ is higher than the utility of the string $start_31complete_31complete_jk$ (3 and 2, respectively). Thus, the control methodology will disable the event $start_31$, which is a more desirable task allocation. To eliminate this bias arising from the limited depth of the lookahead window, a normalized utility function

$$U_N(s) = \frac{1}{N} \sum_{\sigma \in s} u(\sigma), \tag{9}$$

where N is the number of task initiation events in the string s is used. The normalized utility values of the two strings would be 1 and 2, respectively, leading to the preferred choice of task allocation. It should be noted that the normalization of the utility function is also a design consideration and is customizable depending on the characteristics of the mission.

5. Robot Failures and Limited Lookahead Control Policy

Frequently, during a mission, a robot may go offline due to a sensor failure or communication loss resulting in an incomplete task. The control methodology should be able to compensate for robot failures by reallocating tasks to the operational robots of the team. Two kinds of failures are considered in this work: temporary failures and failures with task reinitialization.

Temporary failures, such as communication loss, are failures that can be repaired in a short period of time and the robot may continue executing its task after the repair. Figure 3 demonstrates a temporary failure in a mission with two robots (Robots 1 and 2) and two tasks (Tasks 1 and 2), where each robot can perform both tasks. Robot 1 is failed in State 2 and repaired in State 4 to continue executing its task assignment.

Failures with task reinitialization are failures that require reinitialization of the task assigned to the failed robot. For example, consider a sensor failure that it is not immediately recognized. However, during task execution, the sensor failure is realized and the task needs to be reinitialized. The task, assigned to the failed robot, is reevaluated for allocation. Figure 4 demonstrates a temporary failure with

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

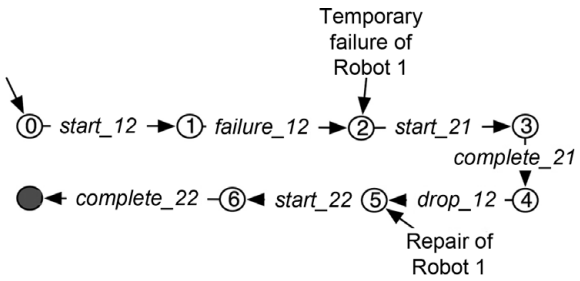


Fig. 4. Events executed in the UCSM after a failure with task reinitialization.

task reinitialization. Robot 1 has failed in State 2 while executing Task 2 and the failure is considered as failure with task reinitialization. In State 4, Task 2 is dropped and assigned to Robot 2.

As mentioned in the previous section, the uncontrollable events *failure_jk* and *repair_jk* have utility function values equal to zero. However, occurrences of these events influence the utilities of the controllable events allowing for an intelligent task allocation design. Consider the case where two robot candidates are to perform a task. Based on the evaluation methodology proposed in this work, the robot with the highest utility will be assigned to the task. If the robot fails while executing the task two things may happen: (1) task reinitialization and (2) the robot will be repaired and complete the task. Based on the utility assignment method discussed so far, the task will be reassigned to the same robot after its repair, since this is the robot with the highest utility. In a real-world application, such decision will not be ideal since the same failure might be repeated. For example, even though a robot can perform the specific task better than any other robot of the team, assume that its communications capabilities are not as strong as the capabilities of other robots. If the robot fails while performing a task due to a communication error, then repeating the task will potentially result in the same failure.

In addition, even in the case of temporary failure without task reinitialization, it might be desired to reassign the task to another robot. Consider that a mission is decomposed into multiple tasks and that Task k is the only task left to complete the mission. If Robot j fails while performing Task k, it might be preferable to drop the task and assign it to a different robot instead of waiting for Robot j to be repaired. Such observations lead to an intelligent controller design for task allocation.

To accommodate these observations in the control design, uncontrollable events influence the utility values of the controllable events related to the failed robots as follows: If a failure occurs in the UCSM, the utility of the failed robot for the specific task is reduced to the minimum of the utilities other robots in the team have for the same task. The evaluation process is repeated. This will allow the controller design to decide in reallocating the task to a different robot.

The block diagram of the control methodology algorithm consists of five main modules as shown in Fig. 5:

(1) *System Initialization*: The UCSM and the specifications are generated as described in Section 3. Based on the fuzzy logic controller, each event is assigned a utility function value using Eqs. (7)–(9).
(2) *Failure Detection*: When a robot failure is detected, the information to be used in the calculation of the new limited lookahead window is forwarded to the next module. This information includes the set of events to be masked, the type of failure, and the expected time to repair.
(3) *Lookahead window formation*: Using as root state the initial state of the UCSM and the specifications model, a lookahead window is formed that includes all the transitions starting from the initial state up to a certain predefined depth in the coupled model. The transitions in the lookahead window form a tree of strings that can be executed in the UCSM.
(4) *String Evaluation*: Each string in the lookahead window is evaluated using the normalized utility function shown in Eq. (9).
(5) *Control Decision*: The event exiting the root state leading the string with the highest utility function value is enabled. All the other controllable events exiting the root state are disabled. In the case that two or more strings present the same utility function value all events leading these strings are enabled. At the next state, the new system state becomes the root state for the lookahead window and the procedure is repeated until the system reaches a marked state.

6. Scalability

The computational complexity of the traditional supervisory control problem is PSPACE hard. However, the proposed methodology does not require generation of a complete supervisor. The evaluation process is based on the events in the system's coupled model. The generation of the coupled model has linear computational complexity.

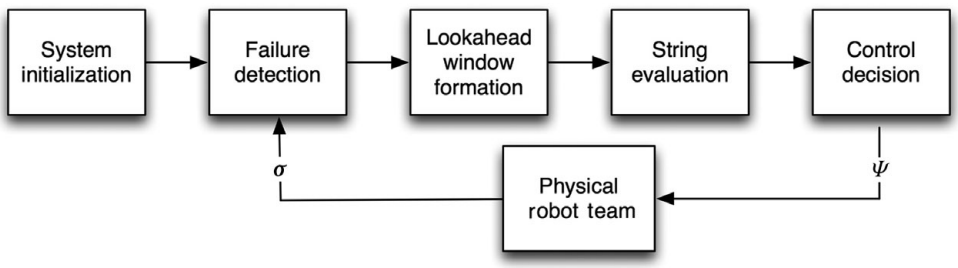


Fig. 5. Block diagram of the control algorithm.

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

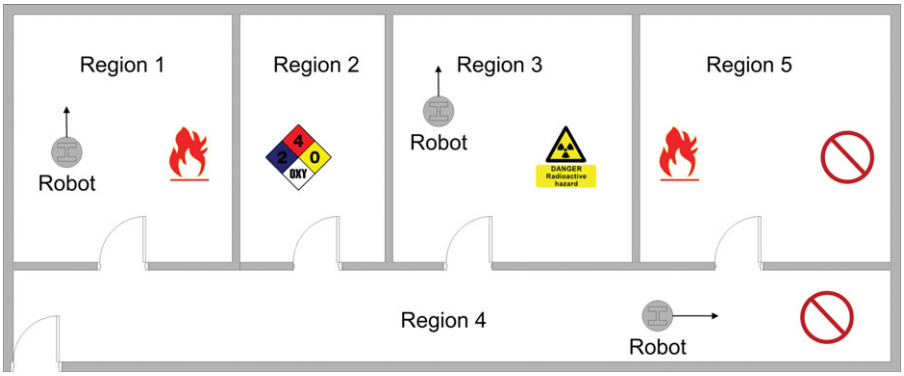


Fig. 6. (Colour online) Warehouse partition for the patrolling scenario.

Table II. Robot sensors.

Robot	1	2	3
Sensors	Fire detector Chemical detector Geiger Counter	Chemical detector Geiger Counter Vision System	Fire detector Vision system

In addition, even though the evaluation problem is a general search problem with an NP-hard computational complexity, using the lookahead control policy presented in this paper, not all the states of the system need to be evaluated.

The complexity of the fuzzy logic design does not depend on the number of robots in the team or the number of the tasks. It depends only on the number of membership functions and the number of fuzzy variables the designer wishes to allow. Thus, the fuzzy logic design is scalable to any number of robots or tasks.

The bottleneck of the proposed methodology is the generation of the coupled model. Even though the computational complexity is linear, the state space increases exponentially. For this reason, the proposed methodology is applicable to medium-size robot teams. In our experiments, we were able to formulate problems with up to seven robots and nine tasks. The number of robots and tasks is comparable and in most cases greater than similar work in task allocation for robot teams reported in literature. An alternative controller design, based on control pattern generation on the fly, as in ref. [51][51] would disregard the coupled model generation and allow the proposed methodology to be applicable to bigger size robot teams. In future research, such methodologies will be considered.

7. Case Study

This section discusses a case study based on computer simulation results. The controller design methodology for dynamic task allocation is applied to a warehouse patrolling application scenario. As depicted in Fig. 6, a team of mobile robots is assigned to patrol a warehouse containing hazardous and security sensitive materials. Three robots with different sensory capabilities are considered. Table II summarizes the sensor suite for each robot.

Based on the *partition-based* patrolling strategy described in ref. [52], the warehouse is partitioned into five regions as follows: Region 1 contains flammable materials, Region 2 contains chemical materials, Region 3 contains radioactive materials, Region 4 contains security sensitive materials, and Region 5 contains security sensitive and flammable materials. This configuration allows us to demonstrate features such as flexibility in task assignment (Region 1 can be assigned to Robot 1 or Robot 3), and cooperation between the robots (Region 5 must be assigned either to Robot 3 or first to Robot 1 and subsequently to Robot 2). The team's mission is to inspect all five warehouse regions.

The overall mission is divided into five tasks where task $k = \{1, 2, 3, 4, 5\}$ corresponds to the patrolling/inspection of the warehouse Region k.

Based on the robot sensory capabilities described in Tables II and III, possible robot-task allocations for Task k and Robot j are defined as

$$k = \begin{cases} \{1, 2, 3, 5\}, & \text{if } j = 1, \\ \{2, 3, 4, 5\}, & \text{if } j = 2, \\ \{1, 4, 5\}\ , & \text{if } j = 3. \end{cases} \tag{10}$$

Each robot is modeled as a finite automaton as described in Fig. 1. The UCSM is described by Eq. 1. Task completion requirements are also modeled as FA, Fig. 2, and the overall mission requirements are described by Eq. 3. Tasks 1–4 are modeled based on Alternative 2 and Task 5 is modeled based on Alternative 3.

The ability of each robot to perform a task is computed by the fuzzy logic controller described in Section 4. The membership functions for the fuzzy variables endurance, efficiency, and designer's choice are shown in Table IV.

Table III. Region/robot allocation.

Region 1	Region 2	Region 3	Region 4	Region 5
Flammable material Robot 1/3	Chemical material Robot 1/2	Radioactive material Robot 1/2	Security Sensitive material Robot 2/3	Flammable & Sec. Sensitive material Robot 3/1 & 2

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

Table IV. Fuzzy logic membership functions for the patrolling scenario.

Rob.	Fuzzy variable	Task 1	Task 2	Task 3	Task 4	Task 5
1	Endurance	Long	Fair	Long	–	Long
	Efficiency	Low	Med.	Med.		Med.
	Des. Choice	Low	High	Low		Low
2	Endurance	–	Fair	Fair	Fair	Short
	Efficiency		High	Med.	Med.	Med.
	Des. Choice		Low	High	High	Low
3	Endurance	Short	–	–	Fair	Fair
	Efficiency	Med.			Med.	Med.
	Des. Choice	High			Low	High

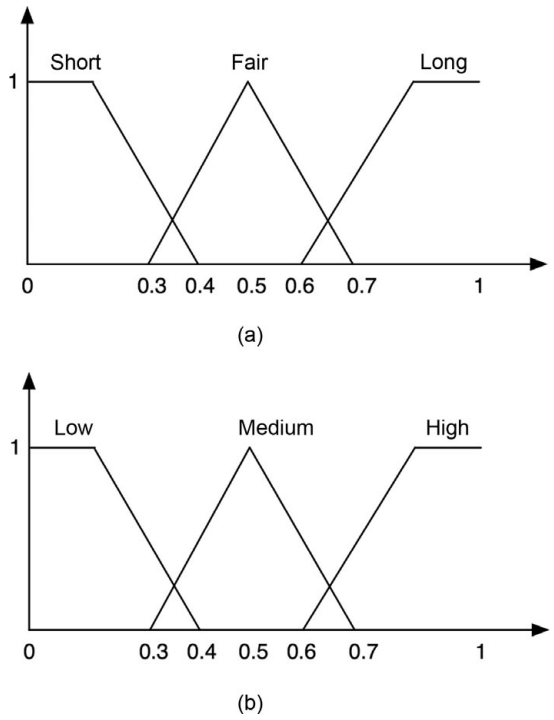


Fig. 7. Membership functions of the fuzzy variables: (a) robot's endurance and designer's choice, (b) robot's efficiency and robot's ability.

Figure 7 depicts the fuzzy variables and their membership functions we have adopted for the patrolling application.

There are 27 such rules in the fuzzy controller, which cover all combinations among the membership functions of the input variables. The corresponding event utility function values based on Eq. (7) are

$$\begin{aligned} &u(start_11) = 0.16,\ u(start_12) = 0.81,\ u(start_13) = 0.33,\\ &u(start_15) = 0.16,\ u(start_22) = 0.33,\ u(start_23) = 0.83,\\ &u(start_24) = 0.81,\ u(start_25) = 0.18,\\ &u(start_31) = 0.81,\ u(start_34) = 0.17,\\ &u(start_35) = 0.83. \end{aligned} \tag{11}$$

The maximum profit due to control actions is equal to 4.09 and is achieved when the following task assignments are made in any order

$$\{start_31,\ start_12,\ start_23,\ start_24,\ start_35\}. \tag{12}$$

Table V. Summary of simulation results.

Depth		Average	% compared with maximum lookahead depth
2	Total utility	3.38	
	Avg. state space	3.13	1%
	Total state space	39.00	1%
3	Total utility	4.03	
	Avg. state space	13.20	4%
	Total state space	139.85	5%
6	Total utility	3.93	
	Avg. state space	132.37	37%
	Total state space	1168.45	43%
14	Total utility	3.87	
	Avg. state space	356.87	100%
	Total state space	2723.00	100%

Table VI. Summary of simulation results, experiment 2.

Depth		Average	% compared with maximum lookahead depth
3	Total utility	3.96	
	Avg. state space	28.29	Less than 1%
	Total state space	339.00	2%
9	Total utility	4.07	
	Avg. state Space	1462.00	40%
	Total state space	11667.00	59%
14	Total utility	4.10	
	Avg. state space	3299.00	92%
	Total state space	19382.00	99%
18	Total utility	4.12	
	Avg. state space	3586.00	100%
	Total state space	19488.00	100%

Robot failures are generated using a uniformly distributed random variable $p_j \in [0, 1]$ denoting the probability of Robot j to fail.

Three performance measures of interest are reported. *Total utility* refers to the sum of the utility of the events executed in the experimental run. This is an indicator of the quality of task allocation decisions where the higher values indicate better task allocation decisions. *Average state space* refers to the average number of states explored each time a lookahead window is created during an experimental run. This metric refers to the computational requirement for real-time decision making. Finally, *total state space* metric refers to the total number of new states explored during the entire experimental run.

Table V summarizes the results of the 20 experimental runs. In all cases including the maximum lookahead depth of 14, the average total utility was less than the maximum total utility. Note that the percentages of computational requirements for the LLD of 2, 3, and 6 were less since the maximum lookahead depth in this scenario was higher.

In order to demonstrate the impact of the LLD on task allocation decisions based on the total utility of the executed events, we compared the null hypothesis

$$H_o : \mu_2 = \mu_3 = \mu_6 = \mu_{14} \tag{13}$$

against the alternative hypothesis that the means are different using a single-factor analysis of variance (ANOVA) with four

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

levels of LLD and 20 repetitions. The results indicate that there is a significant difference between the means and LLD is a factor that impacts the total utility level. Subsequently, we conducted similar experiment but this time using three levels of LLD 3, 6, and 14. The results of this analysis show no significant difference between the means.

In summary, these preliminary experiments indicate that a limited lookahead control policy provides a computationally efficient approach to the problem of task allocation. However, the depth of the limited lookahead window must be carefully chosen based on the characteristics of the mission as well as the desired level of optimality as the quality of the task allocations is dependent on this parameter. Furthermore, these preliminary results indicate that LLD of less than $^1/_4$ of the maximum lookahead depth provides compounded reductions in the computational requirements where on average less than 10% of the computational requirements are sufficient to make control decisions. However, it must be pointed out that this statement must be further verified with larger experimental designs.

Additional simulations with three robots and six tasks have been performed. Table VI summarizes the results. The maximum utility in this case is 5.09.

8. Conclusions

In this paper, we describe a novel control methodology for task allocation in cooperative robot teams. The FA formalism is used to model the robot team and the mission requirements as DESs. In developing the system model, we considered flexibility in task assignment, robot coordination for task completion, and robot failures and repairs. These characteristics are commonly encountered in mission planning and execution of cooperative robot teams. We also describe a utility function for task allocation that uses fuzzy logic to describe various robot capabilities, which are difficult to quantify. Subsequently, a limited lookahead control policy coupled with a fuzzy controller is developed for task allocation in real time.

The use of limited lookahead policies presents significant advantages in terms of computational complexity. The computational complexity of the traditional supervisory control problem is polynomial[5] if the UCSM and the specification's model describe *perfectly* the behavior of the robot team and the mission requirements. However, if there is *imperfect* information about the system then the complexity becomes PSPACE hard.[53] The limited lookahead policy proposed in this paper is based on the coupled model of the system where the computational complexity of the coupled model generation is linear. The computational results show that only a fraction of the coupled model needs to be explored in the limited lookahead window to make task allocation decisions. The preliminary results show that task allocation decisions based on the limited lookahead window control policy produce comparable results when compared with exploring the complete coupled model. Further completely randomized experimentation is required to generalize these findings. The limited lookahead depth is a critical parameter affecting the quality of task allocation as well as computational complexity. The results in Section 7 indicate that larger limited lookahead depths lead to higher number of states visited by the control algorithm. This is an expected result, however, the associated increase in the utility of task allocation is not as clear cut. In this work, as in most of the referenced literature, the depth of the lookahead window is arbitrarily chosen. In ref. [28], the depth window is computed based on the number of uncontrollable events in the system. A future research direction involves determining the characteristic associated with cooperative robot teams and their missions, which may be used to develop a methodology to calculate a dynamic limited lookahead depth in real time. Such an approach will result in a controller that is adaptable to the changing needs of missions and cooperative robot teams.

References

1. V. Kumar and F. Sahin, "Cognitive Maps in Swarm Robots for the Mine Detection Application," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA (2003) pp. 3364–3369.
2. Y. Zhang, M. Schervish, E. U. Acar and H. A. Choset, "Probabilistic Methods for Robotic Landmine Search," *Proceedings of the International Conference on Intelligent Robots and Systems*, Maui, HI, USA (2001) pp. 1525–1532.
3. J. S. Jennings, G. Whelan and W. F. Evans, "Cooperative Search and Rescue with a Team of Mobile Robots," *Proceedings of the International Conference on Advanced Robotics*, Monterey, CA (1997) pp. 193–200.
4. B. Krishnamurthy and J. Evans, "HelpMate: A Robotic Courier for Hospital Use," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Chicago, IL, USA (1992) pp. 1630–1634.
5. P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optim.* **18**(5), 452–462 (1987).
6. L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Autom.* **14**(2), 220–240 (1998).
7. R. Zlot, A. Stentz, M. B. Dias and S. A. T. S. Thayer, "Multi-Robot Exploration Controlled by a Market Economy," *Proceedings of the International Conference on Robotics and Automation*, Washington, DC, USA (2002) pp. 3016–3023.
8. A. V. Timofeev, F. A. Kolushev and A. A. Bogdanov, "Hybrid Algorithms of Multi-Agent Control of Mobile Robots," *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, USA (1999) pp. 4115–4118.
9. W. Liu, A. F. T. Winfield, J. Sa, J. Chen and L. Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adapt. Behav.* **15**(3), 289–305 (2007).
10. T. H. Labella, M. Dorigo and J. L. Deneubourg, "Self-organised task allocation in a group of robots," *Distrib. Auton. Robot. Syst.* **6**, 389–398 (2007).
11. J. Chen and D. Sun, "Resource constrained multirobot task allocation based on leader-follower coalition methodology," *Int. J. Robot. Res.* (2011) (doi:10.1177/0278364910396552).
12. K. Lerman, C. Jones, A. Galstyan and M. J. Matari, "Analysis of dynamic task allocation in multi-robot systems," *Int. J. Robot. Res.* **25**(3), 225–241 (2006).
13. Y. Han, D. Li, J. Chen, X. Yang and Y. Hu, "Task Allocation Algorithm Based on Robot Ability and Relevance with Group Collaboration in a Robot Team," *Proceedings of the International Conference on Intelligent Networks and Intelligent Systems* (2009) pp. 273–277.
14. A. Tsalatsanis, A. Yalcin and K. P. Valavanis, "Optimized Task Allocation in Cooperative Robot Teams," *Proceedings of the 17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece (2009) pp. 270–275.
15. S. C. Botelho and R. Alami, "M+: A Scheme for Multi-Robot Cooperation Through Negotiated Task Allocation

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press

and Achievement", *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, USA (1999) pp. 1234–1239.
16. B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.* **18**(5), 758–768 (2002).
17. M. G. Lagoudakis, M. Berhault, S. Koenig, P. A. Keskinocak and A. J. Kleywegt, "Simple Auctions with Performance Guarantees for Multi-Robot Task Allocation," *Proceedings of the International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA (2004) pp. 698–705.
18. R. D. Morton, G. A. Bekey and C. M. Clark, "Altruistic Task Allocation Despite Unbalanced Relationships within Multi-Robot Communities," *Proceedings of the International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA (2009) pp. 5849–5854.
19. J. Muralidharan, A. Gunasekaran, S. Nachiappan and A. Nivin Kumar, "Efficient mechanism development for multirobot coordination," *International Journal of Industrial and Systems Engineering*, (2008) Vol. 3, No. 2, pp. 149–161.
20. I. Mezei, V. Malbasa and I. Stojmenovic, "Auction aggregation protocols for wireless robot-robot coordination," *Proceedings of the International Conference on Ad-hoc Mob. Wirel. Netw.* Murcia, Spain (2009) pp. 180–193.
21. M. Elango, S. Nachiappan and M. K. Tiwari, "Balancing task allocation in multi-robot systems using k means clustering and auction based mechanisms," *Expert Syst. Appl.* **38**(6), 6486–6491 (2010).
22. T. Song, X. Yan, A. Liang, K. Chen and H. Guan, "A Distributed Bidirectional Auction Algorithm for Multirobot Coordination," *Proceedings of the International Conference on Research Challenges in Computer Science*, Shanghai, China (2009) pp. 145–148.
23. S. Koenig, P. Keskinocak and C. Tovey, "Progress on Agent Coordination with Cooperative Auctions," *Proceedings of the AAAI Conference on Artificial Intelligence*, Atlanta, GA, USA (2010).
24. H. L. Choi, L. Brunet and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot. Autom.* **25**(4), 912–926 (2009).
25. G. Thomas and A. B. Williams, "Sequential Auctions for Heterogeneous Task Allocation in Multiagent Routing Domains," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, USA (2009) pp. 1995–2000.
26. A. Viguria and A. M. Howard, "An integrated approach for achieving multirobot task formations," *IEEE/ASME Trans. Mechatronics* **14**(2), 176–186 (2009).
27. R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artif. Intell.* **20**(1), 63–109 (1983).
28. S. L. Chung, S. Lafortune and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Trans. Autom. Control* **37**(12), 1921–1935 (1992).
29. R. Kumar, H. M. Cheung and S. I. Marcus, "Extension based limited lookahead supervision of discrete event systems," *Automatica* **34**(11), 1327–1344 (1998).
30. S.-L. Chung, S. Lafortune and F. Lin, "Recursive computation of limited lookahead supervisory controls for discrete event systems," *Discrete Event Dyn. Syst.* **3**(1), 71–100 (1993).
31. S.-L. Chung, S. Lafortune and F. Lin, "Supervisory control using variable lookahead policies," *Discrete Event Dyn. Syst.* **4**(3), 237–268 (1994).
32. N. B. Hadj-Alouane, S. Lafortune and L. Feng, "Variable lookahead supervisory control with state information," *IEEE Trans. Autom. Control* **39**(12), 2398–2410 (1994).
33. M. Heymann and F. Lin, "On-line control of partially observed discrete event systems," *Discrete Event Dyn. Syst.* **4**(3), 221–236 (1994).
34. R. Kumar and V. K. Garg, "Control of stochastic discrete event systems modeled by probabilistic languages," *IEEE Trans. Autom. Control* **46**(4), 593–606 (2001).
35. F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Trans. Autom. Control* **38**(12), 1848–1852 (1993).
36. C. Winacott and K. Rudie, "Limited Lookahead Supervisory Control of Probabilistic Discrete-Event Systems," *Proceedings of the IEEE Conference on Communication, Control, and Computing*, Monticello, IL, USA (2009) pp. 660–667.
37. J. Millan and S. O'Young, "On-Line Supervisory Control of Hybrid Systems Using Embedded Simulations," *Proceedings of the International Workshop on Discrete Event Systems*, Berlin, Germany (2010).
38. A. Auer, J. Dingel and K. Rudie, "Concurrency control generation for dynamic threads using discrete-event systems," *47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA (2009).
39. L. Grigorov and K. Rudie, "Near-optimal online control of dynamic discrete-event systems," *Discrete Event Dyn. Syst.* **16**(4), 419–449 (2006).
40. C. Yi-Liang, S. Laortune and L. Feng, "How to Reuse Supervisors when Discrete Event System Models Evolve," *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, USA (1997) pp. 2964–2969.
41. D. Gordon and K. Kiriakidis, "Adaptive Supervisory Control of Interconnected Discrete Event Systems," *Proceedings of the IEEE International Conference on Control Applications*, Anchorage, AK, USA (2000) pp. 935–940.
42. D. Gordon-Spears and K. Kiriakidis, "Reconfigurable robot teams: Modeling and supervisory control," *IEEE Trans. Control Syst. Technol.* **12**(5), 763–769 (2004).
43. K. Kiriakidis and D. Gordon, "Supervision of Multiple-Robot Systems," *Proceedings of the American Control Conference*, Arlington, VA, USA (2001) pp. 2117–2120.
44. W. Xi, P. Lee, A. Ray and S. A. P. S. Phopa, "A Behavior-Based Collaborative Multi-Agent System," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA (2003) pp. 4242–4248.
45. K. Jamie, R. K. Pretty and R. G. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Trans. Syst. Man Cybern.* **33**(5), 615–619 (2003).
46. S. Kimura, M. Takahashi, T. Okuyama, S. A. Tsuchiya and Y. A. Suzuki, "A fault-tolerant control algorithm having a decentralized autonomous architecture for space hyper-redundant manipulators", *IEEE Trans. Syst. Man Cybern.* **28**(4), 521–527 (1998).
47. A. Tsalatsanis, A. Yalcin and K. P. Valavanis, "Automata-Based Supervisory Controller for a Mobile Robot Team," *Proceedings of the IEEE Third Latin American Robotics Symposium*, Chile (2006) pp. 53–59.
48. C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems* (Kluwer Academic Publishers, Norwell, Massachesetts, USA, 1999).
49. E. H. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *Int. J. Man-Mach. Stud.* **8**(6), 245–254 (1976).
50. R. Bellman, "On the theory of dynamic programming" *Proc. Natl. Acad. Sci.* **38**(8), 716–719 (1952).
51. A. Yalcin, A. Khemuka and P. Deshpande, "Modelling inter-task dependencies and control of workflow managements systems based on supervisory control theory," *Int. J. Prod. Res.* **43**(20), 4359–4379 (2005).
52. Y. Chevaleyre, "Theoretical Analysis of the Multi-Agent Patrolling Problem," *Proceedings of the International Conference on Intelligent Agent Technology*, Beijing, China (2004) pp. 302–308.
53. V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica* **36**(9), 1249–1274 (2000).

https://doi.org/10.1017/S0263574711000920 Published online by Cambridge University Press