This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Workflow Nets for Multiagent Cooperation

Y. T. Kotb, S. S. Beauchemin, and J. L. Barron

***Abstract*—We present a formal framework for robotic cooperation in which we use an extension to Petri nets, known as workflow nets, to establish a protocol among mobile agents based on the task coverage they maintain. Our choice is motivated by the fact that Petri nets handle concurrency and that goal reachability, or soundness, can be theoretically established. In particular, we define a mathematical cooperation operator which turns cooperation problems expressed as workflow nets into algebraic representations. While we do not address the problem of efficiency, we formally demonstrate that this framework guarantees soundness, or goal reachability, using workflow nets.**

***Note to Practitioners*—Cooperative behavior is a key issue in process automation as it enhances the execution time of a process. This contribution proposes a framework which guarantees the completion of a plan given that it is achievable within the set capabilities exhibited by the cooperative agents.**

The proposed algorithm turns a cooperative plan expressed with logic predicates and operators into a workflow net that is guaranteed to complete. If the cooperative plan is static during its execution, then it is achieved at a minimal cost (whether that be time, or any other measures of efficiency). Otherwise, if the cooperative agents possess sufficient perceptual or communication capabilities to assess (partially or fully) the state of the cooperation, then they are free to replan at will. This framework is applicable to any automation scenario that can be expressed in the form of predicate logic.

***Index Terms*—Linear algebra, linear programming, multirobot systems, Petri nets, robotics and automation.**

I. Introduction

ROBOTIC navigation problems often benefit from the advantages provided by multiple, cooperating mobile agents [1]. Such gains include improved performance and simplicity of robot design. In addition, there are common multiagent tasks that cannot be carried out by a single robot, such as soccer playing and follow-the-leader swarms [2]. Conversely, predator-prey and terrain exploration problems are examples of tasks that can be performed by a single agent yet may be more efficiently addressed with multiple robots [3]. Cooperation among a group of robots is defined as the process of merging and managing available capabilities to reach a common goal. Typically, these resources include time, actions, knowledge, sensor readings, and computations.

We briefly survey the relevant literature in Section II, while Section III illustrates the methodology and our proposed framework. Section IV presents a cooperative operator and its properties. Section V elaborates on the rudiments of a cooperation algebra founded on the cooperative operator. Section VI outlines the general cooperation algorithm, while Section VII includes a demonstration of the scalability of the framework. Section VIII offers experimental simulations in support of the framework. Section X offers a conclusion and potential future work.

Manuscript received April 06, 2011; revised June 06, 2011; accepted July 16, 2011. This paper was recommended for publication by Associate Editor Z. Li and Editor M. Zhou upon evaluation of the reviewers' comments.

The authors are with the Department of Computer Science, University of Western Ontario, London, ON N6A-5B7, Canada (e-mail: beau@csd.uwo.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2011.2163510

II. Related Literature

Aalst *et al.* introduced the modeling of workflow systems with an extension to Petri-nets, known as workflow nets [4], [5]. Aalst also introduced constructs for token routing known as AND split and join, and OR split and join [6], [7]. Workflow nets are used in various applied areas of research. For instance, workflow compositions [8], and in areas of synchronization and cooperation[1] [10], [11].

III. The Workflow Framework

In order to design a framework that is capable of supporting cooperation among a set of agents, the tasks to be performed by the system must be taken into consideration. The diversity of task types and constraints yield different designs. An example of this is agent polymorphism, which exists when two or more agents with different capabilities are able to complete the same task. A group of agents is said to be *homogeneous* if the capabilities of the individual agents are identical and *heterogeneous* otherwise. Heterogeneity introduces complexity because task allocation becomes more involved and agents need to model other individuals in the group.

A. Contribution

We aim to define a cooperative framework for robotic agents with the use of workflow nets as defined by Aalst [4], [5]. Our main contributions consist of the application of workflow nets to problems of cooperation among robotic agents. In this context, we define a cooperation operator (Sections IV and V) which is used to compose agent capabilities expressed as workflow nets into a cooperative, composed workflow net. We demonstrate that this cooperation operator preserves the property of soundness, and that the framework is scalable to any number of agents with any number of capabilities. We also propose an algorithm which finds the minimal cost of the agent cooperation. While it can be argued that the difference between synchronization and cooperation may be essentially semantic, we however provide an algorithmic means by which agent capabilities are translated into sound, inductively constructed workflow nets, depicting optimal execution costs, provided certain conditions are met [9], [11]. A detailed description of the algorithm and proofs of our claims are available in a recently published technical report [9].

We use workflow constructs to perform workflow net compositions that are similar to those of Aalst [6], [7]. Our proposed cooperation operator results in performing common place and transition compositions. That our compositions be as such is essential for the incident matrix representation used by our algorithm.

We are interested in cooperating agents sharing their capabilities in the aim of achieving a cooperative plan. In that sense, unlike web service composition approaches [12], [13], we assume that compatibility is assured in the context of agents sharing their capabilities. In web service composition, the issue is one of composed and compatible service at design time, while our approach for cooperating agents must determine the optimal fashion in which to share capabilities for the execution of a cooperative plan at a minimal cost.

Finally, Aalst states that for a composed workflow to be sound, every sub-workflow must end with a token in its output place [7]. Kindler argues that sub-workflows for which it is known that no token will reach their output places should not have undue influence on the soundness of the composed workflow [13]. In our framework, the composed, cooperative workflow is sound as per Aalst's criterion, while the sub-workflows composing it are sound as per Kindler's criterion.

[1]A detailed bibliographical survey of these application areas has been compiled by Kotb *et al.* [9].

1545-5955/$26.00 © 2011 IEEE

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

B. Preliminaries

Workflow nets constitute an extension to Perti nets in that they must contain exactly one place with no incoming transitions and exactly one place with no outgoing transitions. In addition, the notion of soundness with workflow nets implies that the model is both structurally and behaviorally well-formed.

We hypothesize that there is a set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ of primitive agent capabilities that cannot be fragmented into simpler capabilities. Any action α from a robot at a given time is constructed from a list of primitive action types.

If a robot r_i from the set of cooperating robots $R = \{r_1, r_2, \ldots, r_n\}$ has plan d_j from the set of plans $D = \{d_1, d_2, \ldots, d_k\}$, then the robot can perform its plan on its own if and only if the following equation holds:

$$\forall\, \alpha \in d_j : \alpha \in \omega(r_i) \tag{1}$$

where α is an action and

$$\omega(r_i) = \{\mathrm{WF}_{\mathrm{net}\,1}, \mathrm{WF}_{\mathrm{net}\,2}, \ldots, \mathrm{WF}_{\mathrm{net}\,l}\} \tag{2}$$

is the action capability set of robot r_i, where $\mathrm{WF}_{\mathrm{net}\,i}$ are workflow nets, and $d_j = \{\alpha_o(\cup\alpha_k)^*\}$, where $\alpha_o \neq \emptyset$ is a starting action and $(\alpha_k)^*$ is a set of following actions (which might be the empty set $\emptyset$).

Two robots r_i and r_k can cooperate to perform a desired plan d_j if they satisfy the task coverage property as follows:

$$\forall\, \alpha \in d_j : \alpha \in \omega(r_i) \cup \omega(r_k). \tag{3}$$

Robot r_k is a candidate for cooperation with robot r_i if and only if

$$\forall\, \alpha \in \Delta_j : \Delta_j = d_j - \omega(r_i),\ \alpha \in \omega(r_k) \tag{4}$$

where Δ_j is the difference between the capabilities required to achieve plan d_j and the capabilities of robot r_i.

In our proposed framework, we use workflow nets to model robots involved in cooperation and derive benefits from their structural and behavioral characteristics to build a protocol for cooperation. Conditions for an action to be taken are given by the input places to a transition and the results of performing the action are given by the output places from that same transition. Tasks, which can be thought of as sets of actions performed by robots, are represented as tokens in workflow nets.

C. Choice Dependency and Unit Similarity

We address the notions of choice-dependency and unit similarity as they are crucial concepts that affect the design of our framework. For instance, if two or more units among a set of workflow nets are deemed similar, then they can be interchanged in order to accomplish the same task or part thereof. It is thus imperative to identify similar units in order to exploit parallelism and minimize the costs of cooperation.

Choice dependency occurs when two or more units share one or more input places. In such cases, soundness may not be ensured, as one or more of the choice-dependent units may not result in the presence of a token in the output place of the composed, cooperative workflow net.

We ensure that our framework avoids these problems by enforcing that the output place of the cooperative framework is reachable by all units. Additionally, we provide a technique to identify similar units in what follows.[2]

Given a group of robots, their behavioral characteristics must be taken into consideration if the cooperation is to be successful. With that intent in mind, we divide a workflow net into units u_i, where $1 \leq i \leq k$ and k is the number of units composing the net. A unit is a transition comprised of sets of input and output places which model an action, the conditions that must be satisfied prior to its execution, and the results of achieving the action, respectively. We proceed with the mathematical definition of a unit.

Definition 1: A unit is a tuple

$$u_i = (\bullet T_i, T_i, T_i\bullet) \tag{5}$$

where T_i is a transition, $\bullet T_i$ the set of input places to T_i, and $T_i\bullet$ the set of output places to T_i.

The notion of choice dependence among units is relevant as it directly affects levels of cooperation. Units u_1 and u_2 are said to be choice-dependent if and only if their transitions share one or more input places. For instance, if unit u_1 and u_2 are choice-dependent, but unit u_3 is choice-independent, then unit u_1 cannot replace unit u_3 in its actions (and *vice versa*).

Definition 2: A unit u is choice-independent if and only if the following condition holds:

$$\bullet T_i \cap \bullet(T - T_i) = \emptyset \tag{6}$$

where T is the set of transitions in a Petri net, and $T_i \in T$. If the unit is *choice-dependent*, then the set of *choice-dependency* is defined as

$$\{T_j | T_j \in \{T - T_i\} \text{ and } \bullet T_i \cap \bullet T_j \neq \emptyset\} \tag{7}$$

and can be determined by satisfying the following condition:

$$\begin{aligned} &W^+(P_j, T_i) - W^+(P_j, T_k) = 0 \\ &\quad \forall\, {}_{j=1}^{m} P_j \in \bullet T_i \cap \bullet T_k \text{ and } \forall\, {}_{k=1}^{m} T_k \in T \end{aligned} \tag{8}$$

where m is the number of places $P_j \in \bullet T_i$, k the number of transitions $T_k \in T$, and W^+ the input incident matrix of the Petri net. Two units are identical if and only if they satisfy similarities in transition, precondition and postcondition. A transition similarity is defined by the action it belongs to. Two transitions T_1 and T_2 are similar if and only if $T_1 \in \lambda$ implies that $T_2 \in \lambda$, where λ is an action belonging to Λ. Precondition similarities are determined by satisfying

$$\Gamma(T_1) - \Gamma(T_2) = 0 \tag{9}$$

and postconditions similarities, by satisfying

$$\Pi(T_1) - \Pi(T_2) = 0 \tag{10}$$

where $\Gamma(T_i)$ and $\Pi(T_i)$ are column vectors representing the input and output places to and from transition T_i, respectively.

Definition 3: A unit u_1 is similar to unit u_2 (denoted $u_1 \equiv u_2$) if and only if $\exists\, T_1 \in u_1$ and $\exists\, T_2 \in u_2 \mid \Lambda(T_1) = \Lambda(T_2)$ and $\bullet T_1 = \bullet T_2$.

Definition 4: A unit u_1 is identical to unit u_2 (denoted $u_1 = u_2$) if and only if $\exists\, T_1 \in u_1$ and $\exists\, T_2 \in u_2 \mid \Lambda(T_1) = \Lambda(T_2)$ and $\bullet T_1 = \bullet T_2$ and $T_1\bullet = T_2\bullet$.

D. Correctness of Framework

Since we assume that agent capabilities can be expressed with workflow nets ($\mathrm{WF}_{\mathrm{net}}$), then reachability for these is assured [5], [14]. However, we must guarantee that the proposed framework has the property of *soundness* [15], [16], as workflow nets representing initially primitive capabilities are inductively composed to form a cooperation plan, which is also a workflow net. Hence, there is a need to demonstrate that the way by which the plan is constructed preserves soundness.

Consider a cooperative framework among robots

$$\Theta = < \Lambda, R, \Omega(R), D, S, \xi > \tag{11}$$

[2]We inductively define units with the initial set of agent capabilities Λ as a basis.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

where Λ is the set of primitive action types, R the set of cooperating robots

$$\Omega(R) = \{\omega(r_1), \omega(r_2), \ldots, \omega(r_n)\} \tag{12}$$

the set of all robot capabilities, and D the set of plans to be performed by the set of robots. The set of all similarities between robot capabilities is defined as

$$S = \{S_1, S_2, \ldots, S_{n(n-1)}\} \tag{13}$$

where

$$S_k = \mathrm{WF}_{\mathrm{net}\,i} \cap \mathrm{WF}_{\mathrm{net}\,j} \tag{14}$$

$\forall\, \mathrm{WF}_{\mathrm{net}\,i} \in \omega(r_i)$ and $\forall\, \mathrm{WF}_{\mathrm{net}\,j} \in \omega(r_j)$. $\xi = \{\xi_1, \xi_2, \ldots, \xi_z\}$ is the set of workflows that bind two or more different workflows from two or more robots.

A framework is sound if any valid input plan can be carried out successfully, under the hypothesis that the set of robot capabilities satisfies the task coverage requirement. For mathematical convenience, we add a single input place p_i and a single output place p_o.

A cooperation framework Θ is sound if and only if the following conditions are satisfied:

1) $\forall\ \mathrm{WF}_{\mathrm{net}} \in \Omega(R),\ \mathrm{WF}_{\mathrm{net}}$ is sound;
2) $\forall\, \lambda \in \Lambda, \lambda \in \Omega(R)$;
3) $\forall\, d \in D, \exists \mathrm{WF}_{\mathrm{net}_i} \otimes \mathrm{WF}_{\mathrm{net}_j} \mid d$ is executable;
4) $\forall\, \xi_k \in \xi$, ξ_k is a sound workflow net;

where $\mathrm{WF}_{\mathrm{net}_i}$ and $\mathrm{WF}_{\mathrm{net}_j}$ are workflow nets representing capabilities from agents i and j, and $\otimes$ is the cooperation operator, as described in Section IV. Kotb *et al.* provide a proof of this claim [9], which follows the lines of that provided by Aalst [6].

IV. The Cooperative Operator

The cooperative operator embodies the cooperation framework, as it joins (or composes) two cooperative frameworks into one. This joint framework must be sound for it to represent a valid cooperation framework. In light of this, the cooperative operator $\otimes$ must satisfy a number of conditions. For instance, if $\mathrm{WF}_{\mathrm{net}_k} = \mathrm{WF}_{\mathrm{net}_i} \otimes \mathrm{WF}_{\mathrm{net}_j}$, then the following properties must apply:

1) $\mathrm{WF}_{\mathrm{net}_i}$ and $\mathrm{WF}_{\mathrm{net}_j}$ are sound;
2) $\mathrm{WF}_{\mathrm{net}_k}$ is a workflow net with two special places i_k and o_k;
3) $\forall\, \zeta \in \mathrm{WF}_{\mathrm{net}_i} \otimes \mathrm{WF}_{\mathrm{net}_j}$, ζ is sound;
4) $\bullet i_k = \emptyset$, $o_k \bullet = \emptyset$.

The cooperative operator preserves soundness, is associative, noncommutative, and nondistributive. Kotb *et al.* provide the necessary justifications in [9].

V. Cooperation Algebra

In this section we show how a logical description of a plan is converted into our chosen representation of workflow nets using the cooperative operator $\otimes$, as applied to create the incident matrices corresponding to logical operators. These are the *and* ($\wedge$), the *or* ($\vee$), and the *then* ($\rightarrow$)[3].

A. Predicates

In this framework, every predicate is transformed into a unit with a single input and a single output place. For instance, given predicate A, the incidence matrix of its $\mathrm{WF}_{\mathrm{net}}$ is formed as

$$I_A = \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{15}$$

[3]The *not* operator or any higher level operator based on it (such as *xor*) are not used within this framework due to their lack of meaning in the workflow.

Initially, all predicates within the logical description of a cooperative plan are given such incident matrices. Given an incident matrix I, we adopt the following definitions: $p(I)$ is the number of rows in I equivalent to the number of places, $t(I)$ is the number of columns in I equivalent to the number of transitions, $p_o(I)$ is the row number of the output place in I, and $p_i(I)$ is the row number of the input place in I. These definitions are used in the construction of incident matrices resulting from applying the cooperation operator $\otimes$.

B. The $\wedge$ Operator

The *and* operator $\wedge$ joins the incident matrices of its predicates, yielding a new incident matrix describing the workflow net resulting from applying the operator. In other words, $A \wedge B$ is equivalent to the following:

$$I_A \wedge I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ -1 \end{bmatrix} = I_{A \wedge B} \tag{16}$$

where $I_{A \wedge B}$ is the incident matrix. It is the $\wedge$ operator that gives rise to parallelism in the resulting workflow net. For example, $A \wedge B$ signifies that A and B can be accomplished in parallel, given that enough resources with the required task coverage are available.

With i from 0 to $p(I_A)+p(I_B)+1$ and j from 0 to $t(I_A)+t(I_B)+1$, the incident matrix $I_{A \wedge B}(i,j)$ becomes:

- $I_A(i,j)$ if $i < p(I_A)$ and $j < t(I_A)$;
- $I_B(i - p(I_A), j - t(I_A))$ if $p(I_A) \leq i < p(I_A) + p(I_B)$ and $t(I_A) \leq j < t(I_A) + t(I_B)$;
- 1 if $i = p_i(I_A)$ and $j = t(I_A) + t(I_B)$;
- -1 if $i = p_o(I_A)$ and $j = t(I_A) + t(I_B) + 1$;
- 1 if $i = p_i(I_B)$ and $j = t(I_A) + t(I_B)$;
- -1 if $i = p_i(I_B)$ and $j = t(I_A) + t(I_B) + 1$;
- 1 if $i = p(I_A) + p(I_B)$ and $j = t(I_A) + t(I_B)$;
- -1 if $i = p(I_A) + p(I_B)$ and $j = t(I_A) + t(I_B) + 1$;
- 0 otherwise.

C. The $\vee$ Operator

The *or* operator $\vee$, not unlike the $\wedge$ operator, joins the incident matrices of two predicates to form a new incident matrix describing the resulting workflow net. This operator allows one part or another of the cooperative plan to be executed, depending on the results of prior execution. $A \vee B$ is equivalent to the following:

$$I_A \vee I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \vee \begin{bmatrix} 1 \\ -1 \end{bmatrix} = I_{A \vee B}. \tag{17}$$

With i from 0 to $p(I_A)+p(I_B)+1$ and j from 0 to $t(I_A)+t(I_B)+3$, the incident matrix $I_{A \vee B}(i,j)$ becomes:

- $I_A(i,j)$ if $i < p(I_A)$ and $j < t(I_A)$;
- $I_B(i - p(I_A), j - t(I_A))$ if $p(I_A) \leq i < p(I_A) + p(I_B)$ and $t(I_A) \leq j < t(I_A) + t(I_B)$;
- 1 if $i = p_i(I_A)$ and $j = t(I_A) + t(I_B)$;
- -1 if $i = p_o(I_A)$ and $j = t(I_A) + t(I_B) + 2$;
- 1 if $i = p_i(I_B)$ and $j = t(I_A) + t(I_B) + 1$;
- -1 if $i = p_i(I_B)$ and $j = t(I_A) + t(I_B) + 3$;
- -1 if $i = p(I_A) + p(I_B)$ and $j = t(I_A) + t(I_B)$ or $j = t(I_A)_t(I_B) + 1$;
- 1 if $i = p(I_A) + p(I_B)$ and $j \geq t(I_A) + t(I_B) + 2$;
- 0 otherwise.

D. The $\rightarrow$ Operator

The *then* operator $\rightarrow$ creates the sequential sections of cooperative plans between predicates. For instance, the plan $A \rightarrow B$ ensures that A is performed before B. The $\rightarrow$ operator joins the incident matrices of its predicates creating a new incident matrix describing the workflow

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

net resulting from applying the operator. Hence, $A \rightarrow B$ is equivalent to the following:

$$I_A \rightarrow I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} = I_{A \rightarrow B}. \tag{18}$$

With i from 0 to $p(I_A) + p(I_B) - 1$ and j from 0 to $t(I_A) + t(I_B))$, the incident matrix $I_{A \rightarrow B}(i, j)$ becomes:

- $I_A(i, j)$ if $i < p(I_A)$ and $j < t(I_A)$;
- $I_B(i - p(I_A), j - t(I_A))$ if $p(I_A) \leq i < p(I_A) + p(I_B)$ and $t(I_A) \leq j < t(I_A) + t(I_B)$;
- -1 if $i = p_o(I_A)$ and $j = t(I_A) + t(I_B)$;
- 1 if $i = p_i(I_B)$ and $j = t(I_A) + t(I_B)$;
- 0 otherwise.

VI. Algorithm

We defined an algorithm to derive a cooperative workflow net from a plan expressed with the notation used in Section V. It is of note to consider that this algorithm generates a cooperative, composed workflow which contains all the possible cooperative scenarios. Consequently, the algorithm implements a backtracking scheme allowing it to determine the minimal cost cooperative path from the workflow net (the complete description of this algorithm is provided by Kotb *et al.* [9]).

VII. Framework Scalability

Scalability refers to the efficiency with which the system operates when the number of agents increases. Scalability is an important issue and constitutes a measure of the quality of the design of the multiagent system. Our approach guarantees scalability (Kotb *et al.* provide an inductive proof for this claim [9]).

While the framework is scalable, it is not guaranteed to yield the best performance in the case of n heterogeneous robots, where $n > 2$, since the selection of a cooperation robot pair among a set of candidate robots highly affects future plans. However, an optimal solution is obtained in the precise case when plans remain static during their execution since a linear programming technique is applied with respect to the costs of the transitions in the workflow net.

VIII. Experimental Simulations

We built a simulator for the cooperation framework, in which the algorithm described in [9] is implemented. Our goal is to empirically demonstrate that our definition of cooperation is correct and that plans can be adequately established and carried out.

A. Experimental Setup

The input to the simulator consists of a plan in the form of a linear logic expression with operators as described in Section V. Other input parameters consist of a set of agents, each with a set of capabilities, expressed as workflow nets. Each capability corresponds to one action defined in the plan, along with the cost associated with performing that action. Actions that are not part of an agent's set of capabilities have their cost set to infinity. The simulator assigns costs in the following manner: first, a uniformly distributed random variable is used to determine the initial set of capabilities for each agent. When an agent is assigned a capability, the cost for its execution is randomly determined with a normally distributed variable.

Once the agent capabilities are set, the simulation evaluates the task coverage. If the generated agent capabilities are insufficient to provide a complete task coverage, the simulator terminates with ∞ as a cost for execution. Otherwise, the cooperative plan is constructed and executed.

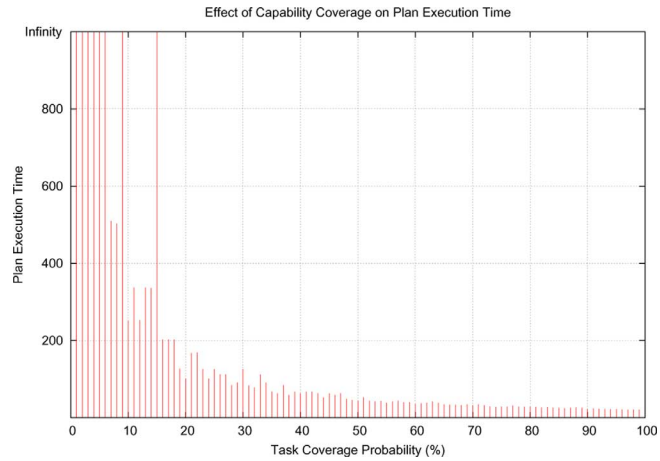

Fig. 1. Plan execution times versus probabilistic agent task coverage for a group of 50 robots.

The execution time is computed simply as the total execution cost in the cooperative workflow net.

B. Experiments

The first experiment was designed to demonstrate the way by which the framework exploits parallelism in plan execution. The plan to execute is expressed as

$$((((A \vee E) \rightarrow (B \vee C)) \rightarrow D) \wedge (G \rightarrow F)) \tag{19}$$

where each predicate represents a unique robot capability. For this plan, seven different capabilities corresponding to the predicates A, B, C, D, E, F, and G are required for its execution. We used 50 agents in 100 simulations, where we controlled the probability of an agent to possess each capability. For instance, for a task coverage probability of 50%, each one of the 50 agents had a 50% chance of possessing each of the seven capabilities. This experiment was performed for task coverage probabilities from 1% to 100%. As expected, with a 100% task coverage probability, each agent possesses all the seven capabilities, resulting in full parallelism of execution, while respecting the flow constraints of the plan. The time units are expressed in terms of transition costs in the workflow nets. Note that these could express other measures. Fig. 1 shows plan execution times for this set of simulations. As expected, times for low task coverage probabilities are high, and sometimes infinite in cases when the probabilistic attribution of agent capabilities is insufficient to complete the plan. It is also observed that, as the probabilistic task coverage increases for each agent, the execution time decreases in what seems to be a negative exponential function. This is, in part, due to the logical structure of the plan, expressed with (19), which allows for parallelism. Conversely, execution times for a plan such as

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \tag{20}$$

would turn out as constant, or infinity when the sum of probabilistic task coverages of agents is insufficient.

Our second set of experiments explores the effects of varying numbers of agents on plan execution times. As expected, execution times are shortened by increasing numbers of agents. Fig. 2(a) depicts this situation where a growing number of agents significantly reduces execution times. Fig. 2(b), showing the cases in which probabilistic task coverages are insufficient to complete the plan, and Fig. 2(c), showing

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

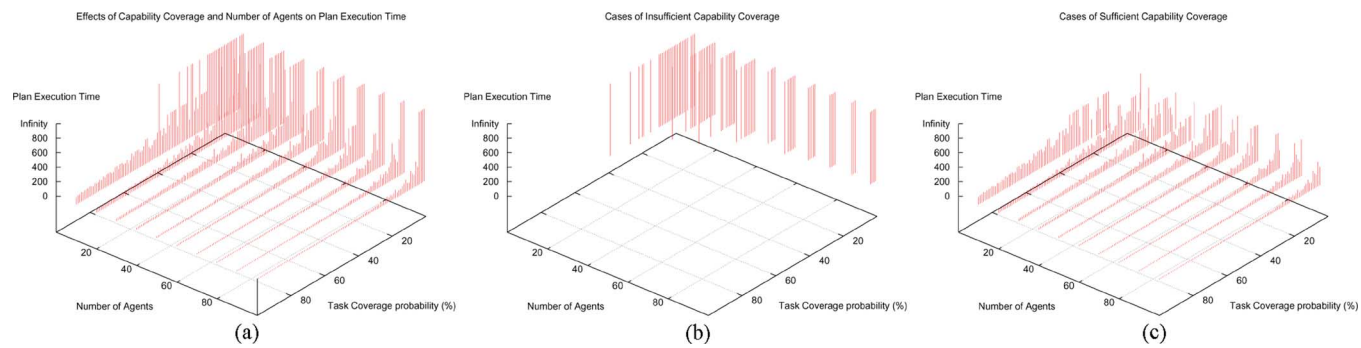

Fig. 2. (a) Plan execution times versus number of agents and probabilistic task coverage. (b) Insufficient task coverage cases. (c) Sufficient task coverage cases.

the converse, demonstrate that in the particular case of this plan, extended agent capabilities reduce execution times more drastically than the number of robots. In this particular case, the framework favors agents with better task coverage than number of robots for plan execution time reduction.

IX. Limitations

Our proposed framework for cooperation has a number of limitations, which we proceed to describe.

1) The input plan must be expressed with linear logic. Cooperative situations that cannot be expressed as such, cannot be dealt with.
2) The framework implicitly assumes that agent capabilities are sufficient to provide task coverage. In cases where agent capabilities do not provide an adequate task coverage, the cooperative plan will not terminate.
3) The framework poses the hypothesis that every agent action will be successful. In practice, when this is not the case, no mechanism is provided to remedy the situation and the cooperative plan may fail.
4) Any change in agent status cannot be considered while the cooperative plan is being determined. For instance, such changes may be modifications to agent capabilities. When capabilities change, the algorithm (from Section VI) must be executed again.

X. Conclusion and Future Work

We proposed a workflow net-based cooperative framework for multirobot systems. The framework provides an algorithm to verify similarities among robot capabilities in order to determine the possibility of cooperation with respect to a desired task. Similarities are examined from what we have defined as compositions. The dynamic behavior of the framework is also studied by investigating the reachability criteria and ensuring that the framework is sound, provided that the design obeys the specified constraints, while the scalability issue is dealt with in full. To conclude, cooperation is achievable by the proposed framework provided that the task coverage criteria are met by the robots and the design follows the soundness constraints specified in [9].

References

[1] J. Haddad and S. Haddad, "Self-stabilizing scheduling algorithm for cooperating robots," in *Proc. ACS/IEEE Int. Conf. Comput. Syst. Appl.*, 2003, pp. 128–135.
[2] T. Arai, E. Pagello, and L. Parker, "Editorial: Advances in multi-robot systems," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 655–661, Oct. 2002.
[3] T. Balch and L. Parker, *Robot Teams: From Diversity to Polymorphism*. Natick, MA: A. K. Peters, Ltd., 2002.
[4] W. van der Aalst, V. Hee, and G. Houben, "Modelling and analysing workflow using a Petri-net based approach," in *Proc. 2nd Workflow on Comput. Support Cooperative Work, Petri Nets and Related Formalisms*, Jun. 1994, pp. 31–50.
[5] W. van der Aals, "Verification of workflow nets," in *Proc. 18th Int. Conf. Appl. Theory of Petri Nets*, 1997, pp. 407–426.
[6] W. van der Aals, "The application of Petri nets to work-flow management," *J. Circuits Syst. Comput.*, vol. 8, no. 1, pp. 21–66, 1998.
[7] W. van der Aals, "Interorganizational workflows: An approach based on message sequence charts and Petri nets," *Syst. Sci.*, vol. 34, no. 3, pp. 335–367, 1999.
[8] W. Tan, Y. Fan, M. Zhou, and Z. Tian, "Data-driven service composition in enterprise SOA solutions: A Petri net approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 686–694, Jul. 2010.
[9] Y. Kotb, S. Beauchemin, and J. Barron, Work-flow nets for multi-agent cooperation: Theory and simulation Dept. Comput. Sci., Univ. Western Ontario, London, ON, Canada, Tech. Rep. TR-742, 2011.
[10] A. Borkowski, M. Gnatowski, and J. Malec, "Mobile robot cooperation in simple environments," in *Proc. 2nd Int. Workshop on Robot Motion and Control*, Oct. 2001, pp. 109–114.
[11] K. Barkaoui and R. B. Ayed, "Uniform verification of workflow soundness," *Trans. Inst. Meas. Control*, vol. 33, no. 1, pp. 133–148, 2011.
[12] W. Tian, Y. Fan, and M. Zhou, "A Petri net-based method for compatibility analysis and composition of web services in business process execution language," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 94–106, Jan. 2009.
[13] E. Kindler, A. Martens, and W. Reisig, "Inter-operability of workflow applications: Local criteria for global soundness," *Lecture Notes in Computer Science, Business Process Management*, vol. 1806, pp. 235–253, 2000.
[14] W. van der Aalst, M. Weske, and G. Wirtz, "Advanced topics in work-flow management: Issues, requirements and solutions," *J. Integr. Design and Process Science*, vol. 7, no. 3, pp. 49–77, 2003.
[15] Y. Kotb and E. Baderdin, "Synchronization among activities in a work-flow using extended work-flow Petri nets," in *Proc. 7th IEEE Int. Conf. E-Commerce Technol.*, 2005, pp. 548–551.
[16] M. Purvis, M. Purvis, and S. Lemalu, "An adaptive distributed work-flow system framework," in *Proc. 7th Asia-Pacific Software Eng. Conf.*, Los Alamitos, CA, 2000, pp. 311–318.