Petri Net-Based Cooperation In Multi-Agent Systems

Y.T. Kotb, S.S. Beauchemin, and J.L. Barron
{*ykotb, beau, barron*} *@csd.uwo.ca*
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada
N6A 5B7

***Abstract*— We present a formal framework for robotic cooperation in which we use an extension to Petri nets, known as work-flow 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 can be theoretically established. We describe the means by which cooperation is performed with Petri nets and analyze their structural and behavioral characteristics in order to show the correctness of our framework.**

I. Introduction

Robotic navigation problems often benefit from the advantages provided by multiple, cooperating mobile agents [5], [9], [12]. Such gains include improved performance and simplicity of robot design. In addition, there are common multi-agent tasks that cannot be carried out by a single robot, such as soccer playing and follow-the-leader swarms [3]. 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 [4].

Cooperation among a group of robots is defined as the process of allocating and managing available resources to reach a certain goal. Typically, these resources include time, actions, knowledge, sensor readings, and computations. As such, a cooperative situation must satisfy various constraints on the goal, the tasks, and the robots themselves. They may be summarized as:

- Constraints on the nature and the amount of resources to be assigned to each robot, and the time frame in which the goal must be reached.
- Constraints on the tasks to perform, such as precedence ordering and the amount of time to complete tasks.
- Constraints on task and robot synchronization, if required.

Cooperating mobile agents thus negotiate for resources and perform task planning and scheduling in order to accomplish common goal.

We briefly cover relevant previous research in Section II, followed by a focus on the related challenges posed by Perti nets in Section III. Section IV illustrates the proposed solution and Section V presents simulation results. Sections VI and VII address the problems posed by input noise and scalability, while Section VIII concludes on our framework.

II. Previous Work

Recently, Lima *et al.* [11] introduced various types of Petri nets to model distinct views of the robotic task model, in addition to quantifying task performance and using learning techniques to improve general efficiency of execution. However, soundness properties were not addressed and it remains unclear whether this framework can guarantee successful cooperative goal completion.

Zhang proposed a Petri net framework for task-level planning [15] in which an algorithm that depends only on the goal and the constraints to derive action sequences is proposed.

Gerkey and Matari presented a domain-independent framework for multi-robot task allocation in which it is shown that task-allocation may be thought of as an instance of the optimal assignment problem [7]. Alternatively, Noborio and Edashige proposed an on-line, deadlock-free path-planning algorithm for multiple agents operating in an infinite world [13].

III. Problem Statement

Given two or more mobile agents, a set of predefined actions, and a goal to attain, we must define a formal cooperative behavior description among the robots to reach the defined goal.

In our approach, we use Petri nets since cooperation requires the handling of concurrency [6], [8]. In addition, Petri nets guarantee the correctness of the cooperation protocol as the notion of reachability in a given Petri net is a provability problem in linear logic. We address the reachability problem by using a special type of Petri nets known as work-flow nets which, when correctly designed, guarantee that the goal is reachable.

A Petri net is a directed graph for which the nodes are either transitions (represented as rectangles), or places (represented as circles). A place is connected to one or more transitions, and a transition is connected to one or more places. Nodes sharing identical types are not directly connected. Activities that are performed by the transitions are represented by tokens (solid circles) and they reside in places. An empty place p that is connected to transition T disables this transition from being executed. A transition is said to be enabled if and only if there is no empty places connected to it as inputs. A transition fires after being enabled and the result of such firing is the removal

of tokens from each of its input places and the creation of tokens in each of its output places.

The entire net models one or more processes in a system. Places can be viewed as conditions to be satisfied for the transitions they are connected to, or as the result of executing one or more transitions in the model. Mathematically, a Petri net is a tuple

$$\aleph =< P, T, F, W > \tag{1}$$

where P is the set of places, T the set of transitions, F the set of arcs among transitions and places, expressed as $P \times T \cup T \times P$, and W a vector containing the weights of the arcs in F.

IV. Methodology and Proposed Solution

In order to design a framework 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.

For instance, we may assume without loss of generality that a set of objects are aligned on a straight line, and that robots r_1 and r_2 are to pick them up. Robots r_1 and r_2 have maximum speeds of 2 m/s and and 1 m/s, respectively. If the constraint on the task is to collect these objects at a speed of 0.5 m/s, then both robots belong to the same polymorphic class given that they both can carry the task out. However, if the required speed of the moving vehicle must be 1.5 m/s, then, the robots do not belong to the same polymorphic domain with respect to this new constraint and only robot r_1 is suitable for the task. This example shows that task constraints and definitions affect the level of cooperation mobile agents may attain.

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.

In our model, we assume that there is a set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ of all primitive action types that cannot be fragmented into simpler action types, and sets of non-primitive action types $\epsilon \subset \Lambda$. Any action α from a robot at a given time is constructed from a list of non-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 it meets the time constraints (if any), and the following equation holds:

$$\forall \alpha \in d_j : \alpha \in \omega(r_i) \tag{2}$$

where α is an action,

$$\omega(r_i) = \{\mathrm{WF}_{\mathrm{net}_1}, \mathrm{WF}_{\mathrm{net}_2}, \ldots, \mathrm{WF}_{\mathrm{net}_l}\} \tag{3}$$

is the action capability set of robot r_i, where $\mathrm{WF}_{\mathrm{net}_i}$ are extensions to Petri nets known as work-flow 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 $\emptyset$).

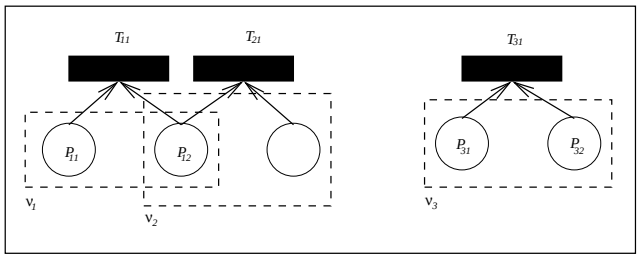

Fig. 1. *Although $T_{11} \equiv T_{31}$, $P_{11} \equiv P_{31}$, and $P_{12} \equiv P_{32}$, ν_1 cannot replace ν_3, since two transitions leave from P_{12}, as per the notion of choice dependency.*

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:

$$\forall \alpha \in d_j : \alpha \in \omega(r_i) \cup \omega(r_k) \tag{4}$$

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

$$\forall \alpha \in \Delta_j : \Delta_j = d_j - \omega(r_i) \text{ such that } \alpha \in \omega(r_k) \tag{5}$$

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 Petri nets to model robots involved in cooperation and derive benefits from their structural and dynamical 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 are thought of as sets of actions performed by robots, are represented as tokens in Petri nets.

Given a group of robots, the structural and dynamical characteristics must be taken into consideration if the cooperation is to be successful. With that intent in mind, we divide a Petri net into units μ_i, where $1 \leq i \leq k$ and k is the number of units composing the Petri 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*

$$\nu_i = (\bullet T_i, T_i, T_i \bullet) \tag{6}$$

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

The notion of choice dependence among units is important because it directly affects the expected level of cooperation. Units ν_1 and ν_2 are said to be choice-dependent if and only if their transitions share one or more input places. For instance,

if unit ν_1 and ν_2 are choice-dependent, but unit ν_3 is choice-independent, then unit ν_1 cannot replace unit ν_3 in its actions, as depicted in Figure 1.

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

$$\bullet T_i \cap \bullet \, (T - T_i) = \emptyset \tag{7}$$

where T is the set of transitions in a Petri net.

If the unit is *choice-dependent*, then the set of *choice-dependency* is defined as:

$$\chi_i = \{T_j \mid T_j \in \{T - T_i\} \text{ and } \bullet T_i \cap \bullet T_j \neq \emptyset\} \tag{8}$$

and can be determined with:

$$\chi(T_i) = \delta(W^-(P_j, T_i) - W^-(P_j, T_k)) \tag{9}$$

$$\forall_{j=1}^{m} \, P_j \in \bullet T_i \cap \bullet T_k \;\text{ and }\; \forall_{k=1}^{m} \, T_k \in T$$

where m is the number of places $P_j \in \bullet T_i$, k is the number of transitions $T_k \in T$, W^- is the input incident matrix of the Petri net, and δ is a Dirac delta function.

Two units are identical if and only if they satisfy similarities in transition, precondition, and post-condition. A transition similarity is defined by the action type 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 type belonging to Λ. Precondition and post-condition similarities are tested by the following equations:

$$S_{pre} = \delta(\Gamma(T_1) - \Gamma(T_2)) \tag{10}$$

and

$$S_{post} = \delta(\gamma(T_1) - \gamma(T_2)) \tag{11}$$

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

Definition 3 *A unit α is similar to unit β if and only if $\exists \, T_1 \in \alpha$ and $\exists \, T_2 \in \beta \mid \Lambda(T_1) = \Lambda(T_2)$ and $\bullet T_1 \equiv \bullet T_2$ and $T_1 \bullet \equiv T_2 \bullet$.*

An example of similar units from two different Petri nets is given in Figure 2. While the concept of units is a step forward in defining cooperative processing, it is not practical, as most units in realistic situations are choice-dependent. Consequently, two or more choice-dependent units may find themselves exchanging actions (or tokens) more often than necessary. Hence, the success of cooperative choice-dependent units is not guaranteed. However, if there is a possibility to view the group of interdependent units as one composition, the process of cooperation becomes feasible and the success of the cooperative process is then guaranteed. Toward this end, we proceed with the definition of compositions of units within Perti nets:

Definition 4 *A composition C is a set of joined units in a topology:*

$$C = \{U, P, F\} \tag{12}$$

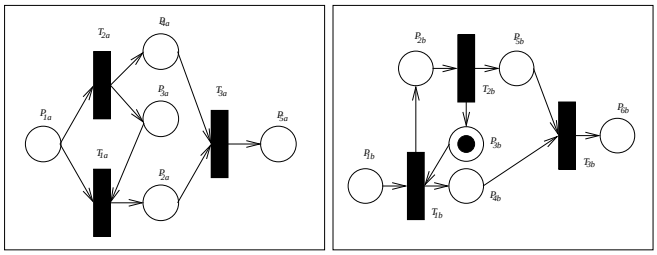

Fig. 2. *Two different Petri nets illustrating the concept of similarity. T_{1a} and T_{1b} are of the same type.*

where U is a set of units, P is a set of places, and $F \subseteq U \times P \cup P \times U$. Figure 4 illustrates similar compositions from two Petri nets.

Definition 5 *A composition $C_1 \subset C_2$ if and only if $\forall \; u_i \in C_1 \;\; \exists \; u_j \in C_2 \mid u_i \equiv u_j$*

A. An Example

Figure 2 shows two simple Petri net units that we would like to test for similarity. Suppose that T_{1a} and T_{1b} are of type ϵ_1, and T_{3a} and T_{3b} are of type ϵ_2. We find the precondition and post-condition similarities as follows:

$$W_1^+ = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad W_1^- = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{13}$$

$$W_2^+ = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad W_2^- = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$\bullet T_{1a} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \bullet T_{1b} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

$$T_{1a} \bullet = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad T_{1b} \bullet = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{16}$$

$$\Gamma(T_{1a}) = 2 \quad \Gamma(T_{1b}) = 2 \quad \gamma(T_{1a}) = 1 \quad \gamma(T_{1b}) = 2 \tag{17}$$

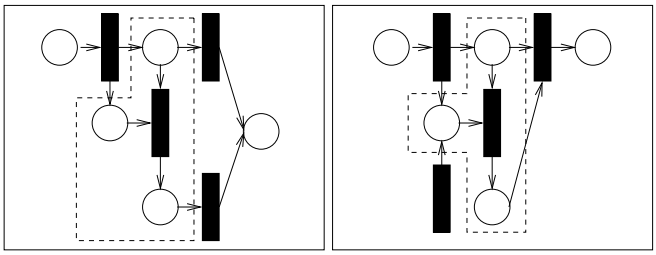

Fig. 3. *Similar units in different Petri nets. In this example, the concept of similarity is applied to subsets of the Petri nets, instead of the entire nets.*

$$S_{pre} = \delta(2-2) = 1 \quad S_{post} = \delta(2-1) = 0 \tag{18}$$

In this example, we find that the two units satisfy the precondition similarity but not the post-condition similarity. Figure 3 illustrates an example of similarity, where the units are comprised within dotted lines. An example of composition is given in Figure 4. A composition is a set of units in a Petri net.

B. Redirecting Activities to Similar Units

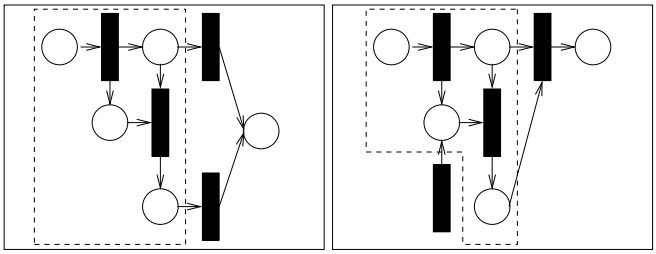

Fig. 4. *Similar compositions in different Petri nets. A composition is is a set of units in a Petri net.*

The merit of this technique is the ability to use similar compositions in Petri nets to perform one or more actions from the task under consideration in such a way that its deadline can met. Consider the example in Figure 5 and assume that there is a token (or action) that is going to miss its deadline in place P_4. If P_2 is empty, then transition T_4 is enabled and executed. As a result, the token under consideration is consumed from place P_4 and regenerated in P_1 and P_3.

When T_1 executes, then the token in P_1 is processed and appears in P_2. Following this sequence, T_5 executes, moves the token to P_5, consumes the token in P_3, and enables T_3, if the required tokens exist. The purpose of transitions T_4 and T_5 is to ensure that the migrating token goes to the desired route and returns, as opposed to being consumed by an undesired transition.

C. Examples of Cooperation

Suppose a work-flow such as that of Figure 6, and two robots r_i and r_j. We assume that every transition in the work-flow is in set $\omega(r_i)$. In the case where $\omega(r_j) = \omega(r_i) - \alpha$, where α is an action to reach a stack of objects, r_i is clearly unable to perform the plan on its own and must ask for the cooperation of r_j to complete its mission, as depicted in Figure 7.

Another example is given by a task requiring two robots r_i and r_j to synchronously act together to perform the task, as shown in Figure 8. Robot r_i is represented by a WF_{net} with the following actions:

- See an object
- Move toward object
- Grip object
- Move toward home

Robot r_j also has these capabilities, plus the following two actions:

- Reach object stack
- Put object on stack

If the function of both r_i and r_j is to grab objects from a loading zone and put them as stacks in another location, then r_j is able to achieve the required task on its own, whereas robot r_i is only able to get objects from the loading zone to a location near a stack. If the two robots cooperate, then whenever robot r_i grabs an object and transfers it to the store, it hands it to robot r_j if it is available and r_j can put it on the top of a stack. If r_j is not available, then r_i waits until r_j becomes available. This cooperative protocol, depicted in Figure 8, shows that when robot r_i hands the object out to r_j, the token that represents the object is then given as the input node of the work-flow representing r_j, as to respect the requirement that any work-flow has a single input entry point.

D. Proof of Correctness

In order to show that the proposed framework is correct, we need to demonstrate that it yields the desired goals for cooperation. As previously stated, the provability problem from linear logic is a reachability problem in Petri nets. Since we use work-flow nets (WF_{net}), reachability is assured [1], [2]. However, we must guarantee that the proposed framework has the property of soundness [10], [14].

A Petri net $\aleph$ is a WF_{net} if and only if [1], [2]:

- $\aleph$ has an input place i, where $\bullet i = \phi$.
- $\aleph$ has an output place o, where $o\bullet = \phi$.
- If a transition t^* is added to $\aleph$ such that $\bullet t^* = o$ and $t^*\bullet = i$, the Petri net $\aleph^*$ becomes strongly connected.

Note that t^* is a transition which connects the input to the output of the WF_{net}.

A WF_{net} $\aleph$ is sound if and only if:

- $\forall\ M \in\ |M_i\rangle,\ \ M_o \in\ |M\rangle$
- $\forall\ M \in\ |M_i\rangle,\ \ M_k \geq n \Rightarrow M = M_o$
- $\forall\ t \in T,\ \ \exists\ M \in\ |M_i\rangle,\ \ t \in\ |M\rangle$

where M is the marking of the WF_{net}, M_i is the input marking, M_o is the output marking, M_k is the marking at

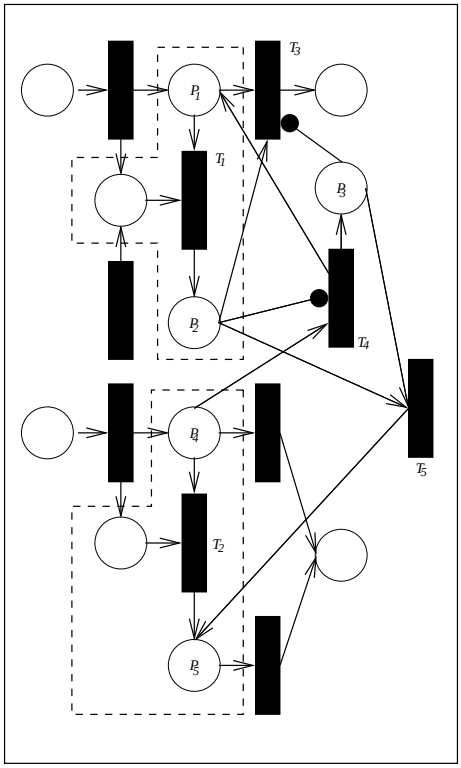

Fig. 5. *Two cooperating Petri nets. Transitions T_1 and T_2 are similar. If there is a token that exists in P_4, then it can be transferred to P_1 to be processed by T_1, instead of T_2. When T_4 fires, it creates a token in P_3 and P_1, which represent the activity under consideration. The token in P_3 disables T_3 and guarantees that the activity flows through the right path. When T_1 is enabled, it eventually fires and the token in P_1 moves to P_2, which temporarily disables T_4 and immediately enables T_5, which in turn fires and produces the processed token by putting it in P_5.*

time k, n is the number of tokens in M_o, and t is a transition.

We now present a cooperative framework among robots:

$$\Theta =< \Lambda, R, \Omega(R), D, S, \xi > \tag{19}$$

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

$$\Omega(R) = \{\omega(r_1), \omega(r_2), \ldots, \omega(r_n)\} \tag{20}$$

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

$$S = \{S_1, S_2, \ldots, S_{n(n-1)}\} \tag{21}$$

where

$$S_k = \text{WF}_{\text{net}_i} \cap \text{WF}_{\text{net}_j} \tag{22}$$

$\forall\ \text{WF}_{\text{net}_i} \in \omega(r_i)$ and $\forall\ \text{WF}_{\text{net}_j} \in \omega(r_j)$.
$\xi = \{\xi_1, \xi_2, \ldots, \xi_z\}$ is the set of work-flows that bind two or more different work-flows from two or more robots.

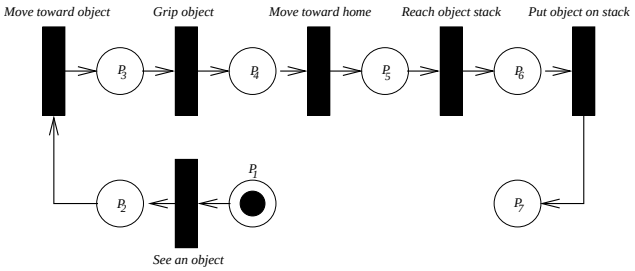

Fig. 6. *A simple WF$_{net}$ model for a robot activity which consists of 6 transitions, each representing a predefined action. Usually, models for robot activities are complex; this example is meant to show how actions are mapped into transitions.*

We present a theorem of soundness for our framework. 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 as shown in Figure 8.

Theorem 4.1: A cooperation platform Θ is sound if and only if the following conditions are satisfied:

- $\forall\ \text{WF}_{\text{net}} \in \Omega(R)$, WF_{net} is sound
- $\forall\ \epsilon \in \Lambda, \epsilon \in \Omega(R)$
- $\forall\ d \in D,\ \exists\ \text{WF}_{\text{net}} \times \text{WF}_{\text{net}} \mid d$ is executable
- $\forall\ \xi_i \in \xi,\ \xi_i$ is a sound work-flow net

Proof:

- Since Ω is sound, then $p_o \in |p_i\rangle$ and $\forall\ d \in D$, d will eventually reach p_o, regardless of the WF it goes through. Hence, we have $\forall\ \text{WF}_{\text{net}} \in \Omega(R)$, WF_{net} is a sound work-flow net
- Since $p_o \in |p_i\rangle$, $D = \alpha_o \cup (\alpha_i^*)$, and that d will eventually reach p_o, we have $\forall \epsilon \in \alpha, \epsilon \in \omega(R)$ and, consequently, $\forall\ d \in D,\ \exists\ \text{WF}_{\text{net}} \times \text{WF}_{\text{net}} \mid d$ is executable
- The definition of soundness $\forall\ M \in |\,p_i\rangle$, $p_o \in |M\rangle$ implies $\forall\ \xi_i \in \xi,\ \xi_i$ is a sound work-flow

Therefore, for sound Θ, the four conditions are satisfied. ■

We now show the converse, namely that if the four conditions are satisfied, the framework is sound.

Proof:

- Since $\forall\ \text{WF}_{\text{net}} \in \Theta(R)$, WF_{net} is a sound work-flow net, we have $\forall\ \xi_i \in \xi$, ξ_i is a sound work-flow net.
- We have $p_o \in |p_i\rangle$ and, since $\forall\ \epsilon \in \alpha, \epsilon \in \Theta(R)$, we obtain $\forall\ d \in D,\ \exists\ \text{WF}_{\text{net}} \times \text{WF}_{\text{net}} \mid d$ is executable.
- With $d \in D$ and $d \in p_i$, d will eventually reach p_o.

We conclude that Θ is sound. ■

V. Simulation Results

Figures 9 and 10 show the dynamic programming algorithm which determines the plan adopted by the two robots to achieve a predefined objective. These Tables solve the cooperation problem shown in Figure 7, where the objective is a plan made of six consecutive actions. The Tables represent robot transitions T_i and desired actions a_j.

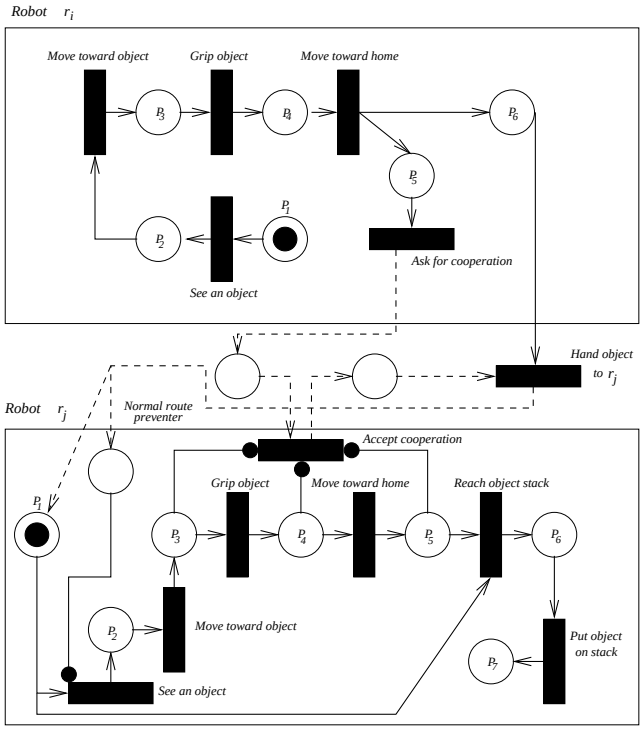

Fig. 7. *A cooperative solution for two robots r_i and r_j to collect objects and put them in a stack in a home zone. Robot r_i sees an object, moves toward it, grips it and returns home. Robot r_j has the same capabilities besides the ability reach the object stack and put the object on the top of the stack. Hence, robot r_i can not complete its mission unless it cooperates with r_j. If robot r_i already has an object and is waiting in the home zone, it asks r_j for cooperation which accepts only if it does not have another object between its grippers and is in the home zone. If this is the case, then robot r_j accepts cooperation and the task is fed to the beginning of r_j's work-flow, and redirected to the transition (reach object stack), so that robot r_j completes the task. Physically, r_i gives the object to robot r_j.*

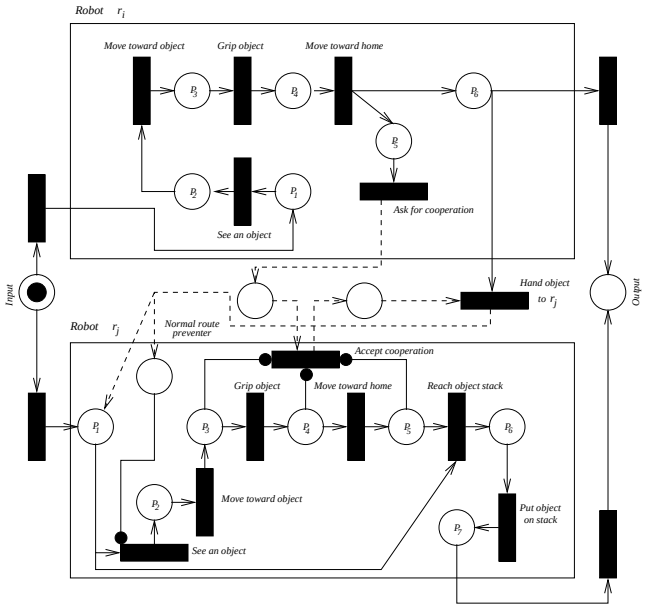

Fig. 8. *The cooperative framework of Figure 7. This diagram displays the input and output places, which are added for mathematical convenience.*

The capabilities of robot r_a are represented by the transitions *See an object*, *Move toward object*, *Grip object*, and *Move toward home*, which are labeled in the Tables from Figures 9, and 10 as T_1, T_2, T_3, and T_4 respectively. The capabilities of robot r_b are labeled by the transitions *See an object*, *Move toward object*, *Grip object*, *Move toward home*, *Reach object stack*, and *Put object on stack*, which are labeled as T_5, T_6, T_7, T_8, T_9, and T_{10} respectively. The dynamic programming algorithm has two phases, forward and backward. The forward phase fills the matrix with either 0 or 1, according to the following equation:

$$f(a_j, T_i) = \begin{cases} 1 & \text{if} \quad T_i \in \lambda \text{ and } a_j \in \lambda \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

For instance, if a transition T_i has the same type of action as a_j, then the matrix element (a_j, T_i) is set to 1, and zero otherwise.

In the backtracking phase, the algorithm constructs plans by gathering the elements that are set to 1 in the forward phase. The algorithm continues until the transition that performs action a_1 is in the plan. After the algorithm halts, a typical plan can be formed from a_n to a_1. It is worth noting that the set of actions the robots take in this typical plan are from a_1 to a_n.

The algorithm shown in Figure 9 gives a plan. In order to get every possible plan, the algorithm has to verify each and every possibility, as shown in figure 10. The dotted rectangles in Figures 9 and 10 separate the transitions which belong to robot r_a from those of robot r_b. If the plan resides within the set of transitions of a single robot, then this plan does not support cooperation.

The simulation results are shown in Figure 11. The x-axis represents the number of objects to be collected, and the y-axis represents the time required to collect that number of objects. We assume an equal amount of time for each transition.

Line $a)$ in the graph represents the case of a single robot collecting a number of objects. It starts with 6 time units for collecting one object and reaches 6000 time units when it is collecting 1000 objects. Line $b)$ represents the cooperation case, and starts at 6 time units when the number of collected objects is 1 and ends with approximately 3550 time units. Line $c)$ in the graph represents the fully parallel case in which each robot is equipped with all the capabilities it needs to perform the task alone. The simulation results show that the cooperation performance tends to get closer to that of the fully parallel case. It is important to notice that getting the cooperation performance closer or equal to the fully parallel case depends on the basic capabilities of the cooperating robots.

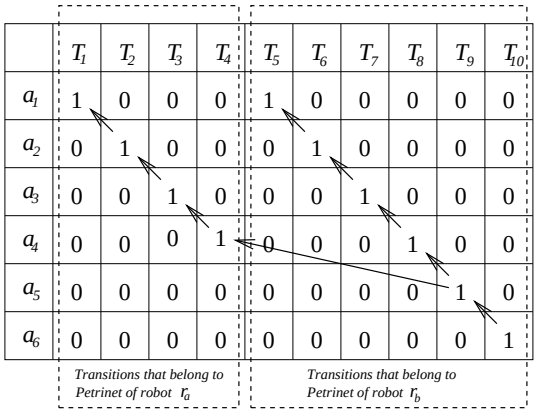

Fig. 9. *A dynamic programming model for finding a plan. This model corresponds to the situation depicted in Figure 7.*

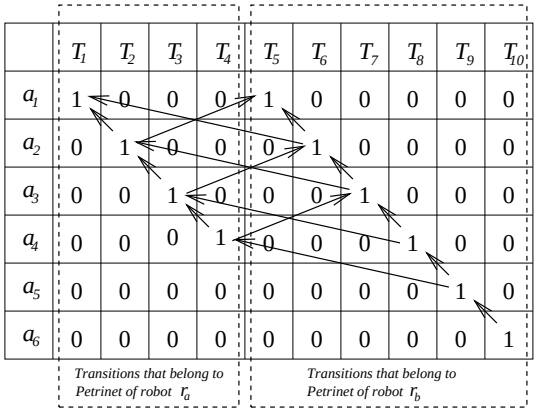

Fig. 10. *A dynamic programming model for finding all plans. This model corresponds to the situation depicted in Figure 7.*

VI. Input Noise

Any action that is required for the success of the cooperation plan must belong to the union of all capability sets of the cooperating agents. If the platform possesses action types that support the handling of noise, then the platform will be able to perform correctly when noise is present. As an example, consider the case shown in Figure 8 and assume that the loading zone and the stack exist in two different rooms separated by a door. Suppose that at some point during the experiment, the door closes for any reason (noise). If one of the agents has the capability of performing the appropriate action (open door), then the platform can handle the situation by forming a new sub-plan to handle the noise as a form of exception handling. However, if none of the cooperating agents is able to perform the appropriate action, then the exception will cause the plan to fail.

VII. Scalability

Scalability refers to the operation of the platform when the number of agents increases. Our approach guarantees scalability and, to prove our claim, we use induction and a

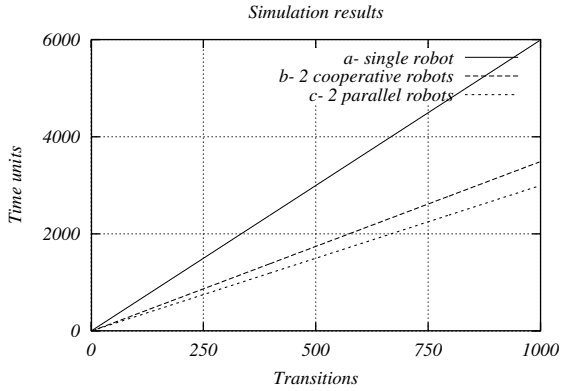

Fig. 11. *Performance analysis of the algorithm. Time units are assumed as constant and identical for all graph transitions.*

union operator $\otimes$ which is applied between two cooperative platforms and produces a third platform as a result. The two cooperative platforms must be sound in order to yield a valid cooperative platform. The operator is also applied on two agents as a base case since a single agent can be formulated as a cooperative platform:

$$\Theta =< \Lambda, R, \Omega(R), D, S, \xi > \tag{24}$$

where S and ξ are both $\emptyset$, since the cooperative platform contains a single agent.

Assume $\Theta = R_i \otimes R_j$, where Θ is the union of R_i and R_j, the work-flows drawn inside the designated boxes from Figure 8. We prove the scalability of the platform by induction using three base cases as follows:

Proof:

- Base case $N = 2$ with agents r_i, r_{i+1}, which corresponds to the case shown in Figure 8.
- Base case $N = 3$ with agents r_i, r_{i+1}, r_{i+2}:

$$r_i = \Theta_i =< \Lambda, r_i, \Omega(r_i), D, \emptyset, \emptyset > \tag{25}$$

$$r_{i+1} = \Theta_{i+1} =< \Lambda, r_{i+1}, \Omega(r_{i+1}), D, \emptyset, \emptyset > \tag{26}$$

$$r_{i+2} = \Theta_{i+2} =< \Lambda, r_{i+2}, \Omega(r_{i+2}), D, \emptyset, \emptyset > \tag{27}$$

Thus

$$\Theta_i \otimes \Theta_{i+1} = \Theta_{c1} \tag{28}$$

and

$$\Theta_{c1} \otimes \Theta_{i+2} = \Theta_{c2} \tag{29}$$

are sound cooperative platforms.

- Base case $N = 4$ with agents $r_i, r_{i+1}, r_{i+2}, r_{i+3}$:

$$r_i = \Theta_i =< \Lambda, r_i, \Omega(r_i), D, \emptyset, \emptyset > \tag{30}$$

$$r_{i+1} = \Theta_{i+1} =< \Lambda, r_{i+1}, \Omega(r_{i+1}), D, \emptyset, \emptyset > \tag{31}$$

$$r_{i+2} = \Theta_{i+2} =< \Lambda, r_{i+2}, \Omega(r_{i+2}), D, \emptyset, \emptyset > \quad (32)$$

$$r_{i+3} = \Theta_{i+3} =< \Lambda, r_{i+3}, \Omega(r_{i+3}), D, \emptyset, \emptyset > \quad (33)$$

Thus

$$\Theta_i \otimes \Theta_{i+1} = \Theta_{c1} \quad (34)$$

and

$$\Theta_{i+2} \otimes \Theta_{i+3} = \Theta_{c2} \quad (35)$$

and

$$\Theta_{c1} \otimes \Theta_{c2} = \Theta_c \quad (36)$$

are sound cooperative platforms.

- Hypothesis: We assume that the following applies:
  1) There are $k-1$ agents
  $$\{r_i, r_{i+1}, ..., r_{i+k-2}, r_{i+k-1}\}$$
  2) The first $k-2$ agents constitute a sound platform Θ_{k-2}.
  3) $\Theta_{k-2} \otimes r_{i+k-1}$ constitutes a sound Θ_{k-1}.
- Induction: It is required to show that an agent r_{i+k} can join the cooperation plan given that the proposed theorem applies.
  1) From the hypothesis, Θ_{k-2} is a sound cooperation platform.
  2) From the base case, $r_{i+k-1} \otimes r_{i+k} = \Theta_c$ is a sound cooperation platform given that the theorem applies.
  3) From the definition of the union operator, $\Theta_c \otimes \Theta_{k-2}$ yields Θ_k which is sound given that the proposed theorem applies.

Therefore, The proposed platform is scalable for any number k of agents. ■

While the platform is scalable, it is not guaranteed to yield the best performance in the case of $N > 2$ heterogeneous agents, owing to the fact that the selection of a cooperation pair among a set of candidate agents highly affects the planning process. The problem of achieving full cooperation among a set of N agents while taking performance into consideration is part of our future work.

VIII. Conclusion

We proposed a Petri net-based cooperative framework for multi-agent systems. The framework provides an algorithm to verify similarities among agent 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 criterion and ensuring that the framework is sound, provided that the design obeys the specified constraints. A theorem and a proof of soundness is proposed. To conclude, cooperation is achievable by the proposed framework provided that the task coverage criterion is met by the agents, and the design follows the soundness constraints specified in the theorem.

References

[1] W.M.P. Van Der Aalst. The application of petri nets to work-flow management. *The Journal of Circuits Systems and Computers*, 8(1):21–66, 1998.

[2] W.M.P. Van Der Aalst, M. Weske, and G. Wirtz. Advanced topics in work-flow management: Issues, requirements and solutions. *Journal of Integrated Design and Process Science*, 7(3):49–77, 2003.

[3] T. Arai, E. Pagello, and L.E. Parker. Editorial: Advances in multi-robot systems. *IEEE Trans. On Robotics and Automation*, 18(5):655–661, October 2002.

[4] T. Balch and L.E. Parker. *Robot Teams: From Diversity to Polymorphism*. A.K. Peters, Ltd., 2002.

[5] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(5):7–27, 1997.

[6] I. Chebbi, S. Tata, and S. Dustdar. The view-based approach to dynamic inter-organizational work-flow cooperation. Technical Report TUV-1841-2004-23, Vienna University of Technology, Salzburg, Austria, 2004.

[7] B.P. Gerkey and M.J. Matari. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *IEEE Int. Conf. on Robotics and Automation*, pages 3862–3868, September 2003.

[8] D. Grigori, F. Charoy, and C. Godart. Coo-flow: A process technology to support cooperative processes. *Int. Journal of Software Engineering and Knowledge Engineering*, 14(3), 2003.

[9] J.E. Haddad and S. Haddad. Self-stabilizing scheduling algorithm for cooperating robots. In *Proc. ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2003.

[10] Y. Kotb and E. Baderdin. Synchronization among activities in a work-flow using extended work-flow petri nets. In *Proc. of the 7th IEEE Int. Conf. on E-Commerce Technology*, pages 548–551, 2005.

[11] P. Lima, H. Gracio, V. Veiga, and A. Karlsson. Petri nets for modeling and coordination of robotic tasks. In *Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 190–195, San Diego, USA, October 1998.

[12] J. Liu and J. Wu. *Multi-Agent Robotic Systems*. The CRC international Series on Computational Intelligence, 2001.

[13] H. Noborio and M. Edashige. A cooperative path-planning for multiple automata by dynamic/static conversion. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1955–1962, July 1993.

[14] M.K. Purvis, M.A. Purvis, and S. Lemalu. An adaptive distributed work-flow system framework. In *7th Asia-Pacific Software Engineering Conf.*, pages 311–318, Los Alamitos, CA, 2000. IEEE Computer Society Press.

[15] W. Zhang. Representation of assembly and automatic robot planning by petri net. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(2):418–422, 1989.