A negotiation-based Multi-agent System for Supply Chain Management

Ye Chen, Yun Peng, Tim Finin, Yannis Labrou, Scott Cost

Computer Science and Electronic Engineering
University of Maryland Baltimore County
Baltimore, MD 21250

{yechen, ypeng, finin, jklabrou, rcost1}@cs.umbc.edu

Bill Chu′, Jian Yao′, Rongming Sun″, Bob Wilhelm″

Computer Science′, Mechanical Engineering″
University of North Carolina
Charlotte, NC 28223

{billchu, jyao, rsun, rgwilhel}@uncc.edu

ABSTRACT

This paper describes an ongoing effort in developing a Multi-agent System (MAS) for supply chain management. In our framework, functional agents can join in, stay, or leave the system. The Supply Chain Management System (SCMS) functionality is implemented through agent-based negotiation. When an order arrives, a virtual supply chain may emerge from the system through automated or semi-automated negotiation processes between functional agents. We present our framework and describe a number of negotiation performatives, which can be used to construct pair-wise and third party negotiation protocols for functional agent cooperation. We also explain how to formally model the negotiation process by using Colored Petri Nets (CPN) and we provide an example of establishing a virtual chain by solving a distributed constraint satisfaction problem.

Keywords

Negotiation, muliti-agent system, supply chain management system, negotiation perforamtive, Color Petri Net.

1. INTRODUCTION

Computer software and hardware development leads to the appearance of non-human software agencies. A software agent is considered as an entity with goals, capable of actions endowed with domain knowledge and situated in an environment [1]. Multi-agent systems (MAS) are suitable for the domains that involve interactions between different people or organizations with different (possibly conflicting) goals and proprietary information [1].

A supply chain is a network of suppliers, factories, warehouses, distribution centers and retailers, through which raw materials are acquired, transformed, produced and delivered to the customer [2]. A supply chain management system (SCMS) manages the cooperation of these system components. In the computational world, roles of individual entities in a supply chain are implemented as distinct agents. Correspondingly, a SCMS transforms to a MAS, in which functional agents cooperate with each other in order to implement system functionality. Most of the previous research work in this field sets MAS in a closed environment, that is, the system consists of a fixed number of entities/components and they have a common target [2][3]. The coordination of chain components is a hierarchical scheduling problem [4]. However, this setting could not accurately reflect the real situation in which a supply chain sits. First, every company in the supply chain has its own interests and goals even though companies may also have intentions to deal with each other. The existence of self-interest makes it difficult to model the agent cooperation as a pure scheduling problem. Second, in a real business environment there are no obligations for companies to remain with a supply chain for a certain time period. Companies may join or leave the chain according to their own judgement. In other words, functional agents have to cooperate in a relatively dynamic way. To address this problem, we propose a negotiation-based MAS framework for supply chain management. In our framework, there is no preset relationship between functional agents. When an order comes, a virtual supply chain may emerge through negotiation processes. The components of the chain may change according to the external situation even after the order has been accepted.

The paper is organized as follows. The proposed framework for negotiation-based MAS is described in Section 2. Negotiation performatives for agent are given in Section 3. Section 4 explains how to use a CPN to formally model the negotiation process. A real supply chain scenario is given in Section 5. Section 6 presents our conclusions.

2. Framework

In this section we propose a framework for the Multi-agent System (MAS) for supply chain management. This MAS consists of heterogeneous types of agents, which implement some functionality of the supply chain management, called *functional agents*. All of them have some understanding of system ontology and use a certain Agent Communication Language (ACL) to make conversation. The system ontology includes knowledge about the goods that the system is dealing with and interaction rules, e.g. the negotiation protocol used in the system. Broadly, functional agents can be divided into two types. One is used for providing system information, called *Information agent,* and may be interacted with the outside. The other type of agents is related to a certain goods, for example, laptop and implements one special role of the supply chain. Initially, functional agents in the system might be unaware of who are its cooperating parties in the chain. They can get the information about potential negotiation partners through the related information agents. Functional agents could be autonomous agents and semi-autonomous agents. They could be as simple as CGI-like programs and distributed across the Internet or other large-scale networks. The knowledge of each agent related to the goods is modeled as a set of constraints. The interaction between them, the *negotiation process,* is modeled as a process of collaboratively assigning values to a set of variables. For example, agents may negotiate the due dates and quantities of orders. The dates and quantities may be subject to the constraint of a producer's capacity. A manufacturer may be willing to evaluate different combinations of order sizes and due dates in order to find qualified vendors with competitive price offers.

In the framework, a number of information agents are predefined, which are in charge of providing different system information, for example, who wants to sell or buy laptops. Except for these agents, other system components are not fixed. Functional agents may join the system and leave it according to their own rules or orders from their owners. A functional agent is said to join the system if it advertises its abilities and desired role in a chain to one of the information agents. Similarly, when a function agent wants to leave, it has to notify information agents in which it registered. There are no centralized super-agents or distributed mediators [5] to handle the agent cooperation. All these activities occur through negotiation processes, regardless of whether two sides are involved in bargaining for some goods intentionally or de-committing a contract caused by the outside events. A system is called "dead" when there are either no registered agents in all the information agents or no virtual chains can be formed in a high percentage when customer orders arrive (further discussion on this measurement is in progress). This framework is used to simulate the dynamic situation that a SCMS encounters, which requires the management system organize the chain flexibly.

At first, the buyer agent that carries the order from a customer looks for a cooperation partner. After knowing who are the potential sellers through the help of information agents, the buyer agent will negotiate with the seller agents directly and find the most suitable one for its own interests. Conversely, the seller agent may look for goods manufacturer agents by searching the information agent (another solution is to let the seller agent post the requirement to the information agents and wait for manufacturers to contact it). All these messages will broadcast and gradually propagate through the whole system. Finally, a virtual supply chain may emerge, from material provider agents to retailer agents. The negotiation process resolves the incoming order constraints step by step across the self-interested agents and during which a virtual chain emerges from the MAS. This process could be described by the following figure:

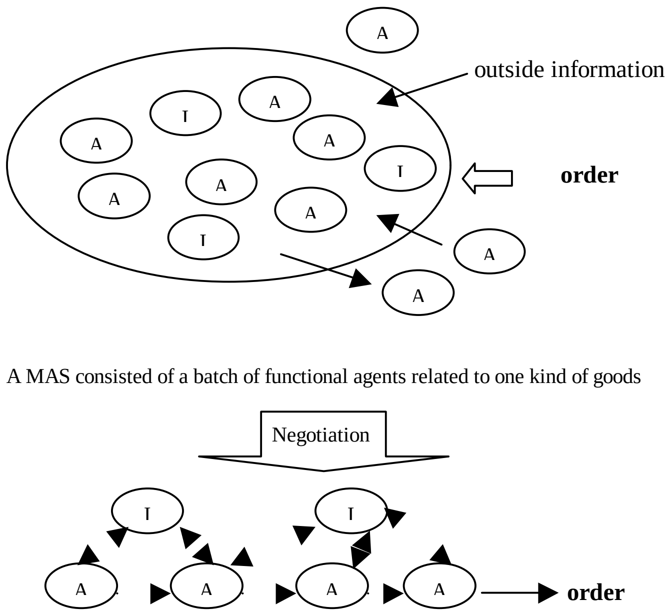
A
outside information
I
A
A
A
I
order
A
A
A
I
A
A
A
A MAS consisted of a batch of functional agents related to one kind of goods
Negotiation
I
I
A
A
A
A
order

Figure 1 A negotiation-based MAS for supply chain management
(A: functional agents that act as a role in supply chain; I: information agents)

In the business environment, the functionality of a SCMS has been defined as "the right products in the right quantities (at the right place) at the right moment at minimal cost" [6]. In our research, the functionality of a SCMS has been transformed to constructing a virtual supply chain in a MAS through the negotiation process among functional agents, in which the constraints of an order has been fully or partial satisfied. The goals of the ongoing research are to build up a MAS base on the framework and support a component-based architecture where different evaluation techniques can be used in conjunction with the negotiation protocols discussed here. One of the evaluation techniques we are experimenting is based on distributed constraint satisfaction [7]. We also attempt to explore the analysis of the life cycle of the system.

3. Negotiation performatives and protocols

An agent Communication Language (ACL) is a language with precisely defined syntax, semantics and pragmatics that is the basis of communication between independently designed and developed software agents [8]. Functional agents in a MAS use a common ACL to transfer information, share knowledge and negotiate with each other. Knowledge Query and Manipulation Language (KQML) and the ACL defined by Foundation for Intelligent Physical Agents/ Agent Communication Language

(FIPA ACL) are the most widely used and studied ACLs. Each of them offers a minimal set of performatives to describe agent actions and allows users to extend them if the new defined ones conform to the rules of ACL syntax and semantics. In KQML there are no predefined performatives for agent negotiation actions. In FIPA ACL there are some performatives, such as proposal, CFP and so on, for general agent negotiation processes, but they are not sufficient for our purposes. For example, there are no performatives to handle third party negotiation. In this section we present a negotiation performative set designed for MAS dealing with supply chain management.

3.1 Criteria for performative definition and selection

The criteria we used to define negotiation performatives are the following:

1. **Compatible with existing performatives**. From a practical perspective, to extend either KQML or FIPA ACL performative sets involve a similar process. Since in FIPAACL there is a category containing four negotiation performatives [8], we choose to construct the negotiation performative set based on this subset.

2. **Defining new negotiation performatives based on a negotiation protocol**. There is no clear means to judge the advantages and disadvantages of a particular extension of a standard ACL, just as it is difficult to judge how to add new words and phrases to a language used by human beings. Some research attempts to define a complete negotiation performative set for an ACL by enumerating agent behaviors that may occur in the negotiation process [9]. Agent actions are related to specific scenarios so that it is hard to describe all types of agent behaviors without knowing the details of the negotiation environment or setting. This method, actually, can only produce the incomplete negotiation performative set similar to the other methods. Considering the negotiation process between functional agents, many performatives defined in [9] are unusable. To let functional agents understand so many performatives sounds an overkill.

   A negotiation protocol is used to organize message sequences among agents. Agent negotiation behaviors are depended on the negotiation protocol using in the negotiation process. Since in a MAS for supply chain management, only limited number of negotiation protocols will be adopted, it is appropriate to define negotiation performatives by describing agents possible responses according to a specific negotiation protocol. In this way it is easier to verify the sufficiency of the performative set through modeling the protocol using certain formal tools such as CPN. At the same time, this approach avoids the redundant performative problem, and functional agents need only to learn a relatively small set performatives for negotiation.

In the next two sections we describe the performatives designed for pair-wise and third party negotiation protocols.

3.2 Negotiation performatives for pair-wise negotiation protocol

Performatives for pair-wise negotiation protocol are used when two functional agents negotiate directly. The performatives definitions conform to the FIPA ACL specification. We give their name and corresponding meaning in the following table:

accept-proposal	the action of accepting a previously submitted proposal to perform an action
CFP	the action of calling for proposals to perform a given action
proposal	the action of submitting a proposal to perform a certain action, given certain preconditions
terminate	the action to finish the negotiation process

Table 1 Pair-wise negotiation performatives

Initially, one agent starts negotiation by sending a CFP message to the other agent. After several rounds of conversation in which proposes and counter-propose are exchanged, the negotiation between two agents will end when one side accepts (rejects) the other side's proposal or terminates the negotiation process without any further explanation. It is not necessary that the functional agent responds to each message. A functional agent can simply ignore the incoming messages. It is the sender's responsibility to handle the lost message or in cases of lack of responses.

The pair-wise negotiation protocol simulates a conversation between two persons, in which one side sends an "ask" and the other side sends a "reply." The difference is in the pair-wise negotiation protocol we have to limit the response message types after a functional agent receives incoming messages so that the negotiation process does not become irrelevant to the topic, and at the same time simplify the message handling process. Table 2 gives a summary about the possible performatives a functional agent can use when certain performative comes in:

	Performatives followed
accept-proposal	terminate | NONE;
CFP	proposal | terminate | NONE;
proposal	accept-proposal | reject-proposal | terminate | NONE;
reject-proposal	terminate | NONE;
terminate	NONE.

Table 2 Pair-wise negotiation performatives and expected response

3.3 Negotiation performatives for third party negotiation protocol (auction)

Performatives for a third party negotiation protocol are used when functional agents negotiate through the third party (auctioneer). Some performatives defined for the pair-wise negotiation protocol, e.g. accept-proposal, reject-proposal, are still used for this protocol. One new performative, BID, is introuduced. The syntax of BID is as follows:

- Bid: the action for a bidder to send a corresponding response to an auctioneer

```
        Bid
            :sender <word>
            :receiver <word>-----------------auctioneer
            :content <expression>-----------price for a goods
            :language <word> ----------- --- e.g.,
                            Knowledge Interchange Format
        (KIF)
            :ontology <word>----------------system
             :in-reply-to <word>--------------auction number
             :protocol <word>-----------------the default value is
                                              English auction.
```

In FIPA ACL, INFORM is not included in the category of a performatives for negotiation processes. In this work, INFORM is treated as a perfomative aiding the negotiation process. Specifically, INFORM is used to transfer messages between a seller agent and an auctioneer: the seller agent INFORMs the auctioneer of what it wants to sell and what kind of auction protocol it prefers; the auctioneer INFORMs the seller agent of the auction result. The accept-proposal (reject-proposal) is used by the auctioneer to tell the bidder who has won (lost).

Initially, one functional agent (seller) starts the negotiation by sending an INFORM message to the auctioneer. This message includes the goods that it wants to sell and the highest desired price (or the contract that it wants to be bought and the lowest desired price) and the preferred auction protocol. After receiving the message, the auctioneer will broadcast it to potential bidders (assuming the auctioneer knows that information by querying the information agent) and organize an auction according to the requirement the seller submits. After several rounds of conversation, the negotiation process will end with a deal that was reached between seller agent and bidders. It is the auctioneer's responsibility to notify both the seller and bidders of the winner and the losers. The scenario is described in the following figure:

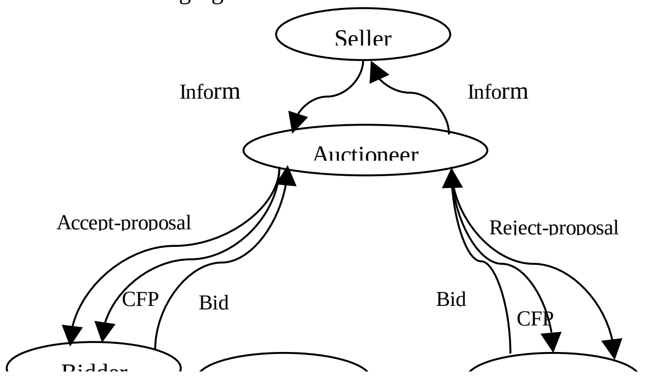

Figure 2 Third party negotiation process

4. Negotiation process modeled by Colored Perit Net (CPN)

In this section we present the negotiation process in a simple MAS associated with the negotiation protocol and negotiation performatives that we have proposed. We use Colored Petri Net (CPN) as the modelling tool. CPN is developed for systems in which communication, synchronisation and resource sharing play an important role. It combines the strengths of ordinary Petri nets with the strengths of a high-level programming language. Petri nets provide the primitives for process interaction, while the programming language provides the primitives for the definition of data types and the manipulations of data values. Usually, a CPN model consists of a set of modules (pages), which each contains a network of *places*, *transition,arcs* and *colored token*. In a CPN-nets place is used to describe possible states of a process. The actions of a process are described by transitions. Arcs are used to connect places and transitions. They are indicated by *ellipses*, *rectangles* and *arrow lines* in the diagram respectively. Tokens contain a dynamically assigned data. Arc expression describes the possible data flow between a place and a transition. A token is called a *colored token* when it is attached a data value. The data value may be of arbitrarily complex type, e.g., a record where the first field is a real, the second a text string, while the third is a list of integer pairs. For a given place all tokens must have token colors that belongs to a specified type. This type is called color. The use of color sets in CP-nets is totally analogous to the use of types in programming languages. Color sets determine the possible values of tokens. Usually a language called CPN ML is used to make CP-net declarations. The statements written in CPN ML can be compiled and tested by some CPN tool, which give users convenience to judge the correctness of the CPN diagram. The modules interact with each other through a set of well-defined interfaces, similar to many modern programming languages. The graphical representation makes it easy to see the basic structure of a complex CPN model, i.e., understanding how the individual processes interact with each other [10]. CPN diagram in figure 4 describes the pair-wise negotiaion process in a simple MAS, which consists of two functional agents bargaining for goods. The messages used are based on the extended FIPA ACL negotiation performative set we propose.

```
Color  AGENT = index a with 1..2;
Color  PR        =  product  AGENT∗ AGENT;
fun     diff (x <> y);
Color  MES      = subset PR by diff declare ms;
Color   PERFORMATIVE  =  accept-proposal | CFP | proposal |  reject-
proposal | terminate;
Color  BARGAIN  = String;
Color  CONTENT =  product PERFORMATIVE ∗ BARGAIN;
Color  E  with e;
fun (s,r,c) = mult′ PR ∗ CONTENT(1`s, AGENT-1`s, c);
```

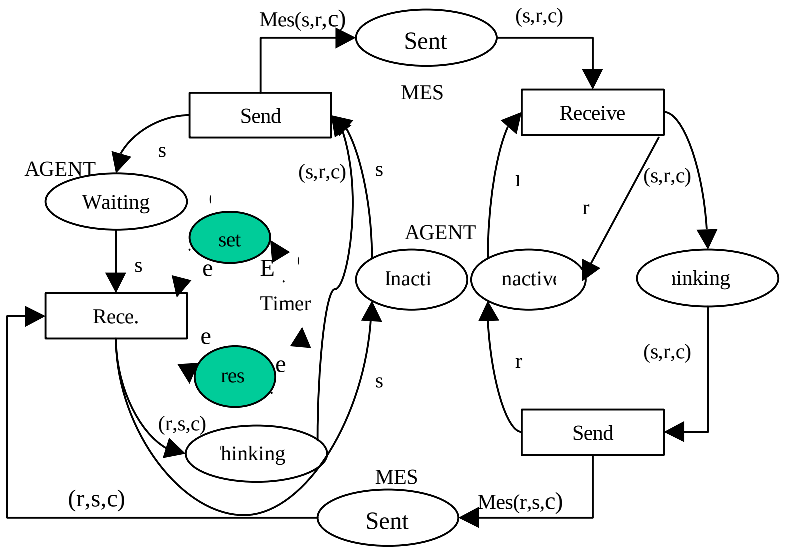

Figure 3 Pair-wise negotiation process for a MAS constituted of two functional agents

From the diagram we can see that there are three different kinds of places: *Inactive*, *Waiting*, and *Thinking*, which reflect the states an agent might sit in during a negotiation process. Both of the agents in this simple MAS have the same architecture. The difference between one functional agent that has three places and the other that only have two comes from the roles they play in the negotiation process. The agent that begin the negotiation, called buyer agent, which is shown on the left side of the diagram, has the responsibility of handling the lost messages or no responses so it has a extra Waiting place. The seller agent (the process on the right) does not need this state. When no messages come in, the seller agent would simply remain in the Inactive place. In the diagram there is one place, *Sent*, and four transitions, *Send XX* or *Receive XX*, that reflect message state and transactions. The place Sent is used to describe the situation that the message is on the way from one agent to the other. The four transitions are used to describe the actions of agent sending or receiving messages. The CPN-ML descriptions above the diagram in figure 4 give a formal way to express the negation process in the system. It can be used for future analysis.

In the system, initially, both agents are in *Inactive* places (states). When the buyer side decides to start a negotiation process, it takes an action *Send Messages* (CFP), and its state changes from *Inactive* to *Waiting*. Now the buyer is waiting for a response (*proposal*, *accept-proposal*, *reject-proposal or terminate*). When the buyer side *Receive Messages*, its state changes from *Inactive* to *Thinking*. In this state the seller agent needs to find out how to reply to the incoming message. Of course, it can just ignore the message. Either choice will cause its state direclty chang to *Inactive*. If there is some response back to the buyer side, its state will change from *Waiting* to *Thinking* or *Inactive*. Otherwise, the Timer will enforce this to happen (In the diagram we can see the Timer issue a special non-colored token that controls this change.).If the *Thinking* result is sent out, all the transitons in the diagram would repeat again. Otherwise the buyer side will return to *Inactive* state. The system comes back to the initial state. Further discussion and verification regarding the negotation process in this system are in progress.

5. Example

In this section we present a concrete example of using constraints to represent knowledge of functional agents and how these constraints can be used in evaluating offers. Here we do not present how to solve the entire constraint problem for the entire virtual supply chain emerged, but instead we give a detailed explaination of how to form one part of the chain, that is, from material supplier to manufacturers. We assume that all the companies are represented as functional agents and they use pair-wise negotiation protocol to interact with each other.

Suppose company C has a production schedule listed below:

T	1	2	3	4	5	6	7
D	300	240	340	260	360	0	0
T	15	16	17	18	19	20	21
D	290	340	140	370	390	0	0

where T stands for time period and D stands for quantity due from the suppliers. For easy of discussion, we restrict to only one part, X. Such a schedule is typically the output from company C's manufacturing resource planning system Agent C starts with the constraint that it needs these quantities at different time periods. It may also have a database of qualified vendors, company A and B, represented by agents A, and B respectively. Agent C sends a "call for proposal" (CFP) message to both agents A and B.

Upon the request of CFP, agents A and B replies with proposals in the form of constraints:

The supplier A's proposal may be:
(if ((supply A X) $\wedge$(2≤supply_lead_time ≤ 4)) ((quantity ≤ 600) $\wedge$(unit_price X 3.05)))

(if ((supply A X) ∧(5≤supply_lead_time ≤ 10)) ((quantity ≤ 1500)∧ (unit_price X 3.00)))
(if ((supply A X) ∧(supply_lead_time ≥ 11)) (unit_price X 2.95))

The first constraint means that if delivery is expected within 2 to 4 weeks, then A can only supply 600 pieces of X at the unit cost of $3.05.

Similarly, supplier B's constraints may be:
(if ((supply B X) ∧ (2≤supply_lead_time ≤ 3)) ((quantity ≤ 500) ∧(unit_price X 2.95)))
(if ((supply B X) ∧ (4≤supply_lead_time ≤ 8)) ((quantity ≤ 1600) ∧(unit_price X 2.90)))
(if ((supply B X) ∧(supply_lead_time ≥ 9)) (unit_price X 2.85))

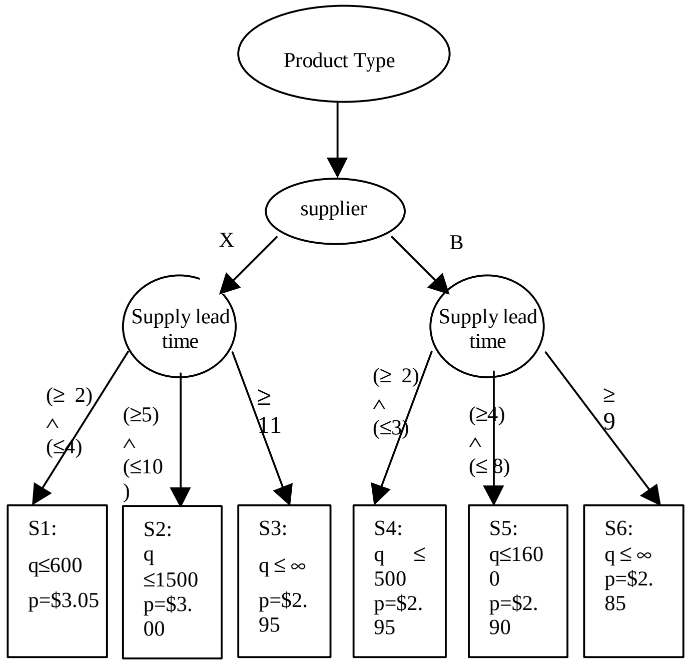

Figure 4 Search tree for constraint solving

Based on all the constraints, agent C will construct a search tree as follows. If a satisfactory solution can be found, agent C will accept the one or both proposals from suppliers A or B. If no solution can be found, agent C has the following options: Change its master schedule. That is to say, agent C will try to relax its own constraints, or Request supply A and B to relax one of their constraints. This alternative would produce counter proposal messages to agents A or B. After several rounds of negotiation, Agent C might setup a cooperation relationship with Agent A or Agent B.

6. Conclusion

In this paper, we have presented a framework of negotiation-based MAS for supply chain management. Negotiation performatives for pair-wise and third party protocol have been designed. A CPN diagram explaining negotiation protocol made up of the proposed performative has been given. The design and implementation of the negotiation-based MAS for supply chain management is in progress. The definition of negotiation performatives needs to be refined and new ones will be added in if necessary. The negotiation protocol expressed in CPN diagram will be given a more formal verification. We model the evaluation process as a distributed constraint satisfaction problem. Many challenges remain. Key research issues include: convergence behavior of a network of negotiating agents, strategies, and orders for deciding when to relax constraints.

7. REFERENCES

[1] P. Stone and M. Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," *Under review for journal publication*, February, 1997.

[2] M. Barbuceanu, and M. S. Fox, "The Information Agent: An Infrastructure Agent Supporting Collaborative Enterprise Architectures," in *Proceedings of Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Morgantown, West Virginia, IEEE Computer Society Press, 1994.

[3] N. Sadeh, "MASCOT: An Agent Architecture for Multi-Level Mixed Initiative Supply Chain Coordination," *Internal Report*, Intelligent Coordination and Logistics Laboratory, Carnegie Mellon University, 1996.

[4] D. Kjenstad, "Coordination Supply Chain Scheduling," *Ph. D. Dissertation*, Department of Production and Quality Engineering, Norwegian University of Science and Technology, 1998.

[5] R. Kalakota, J. Stallaert, and A. B. Whinston, "Implementing Real time Supply Chain Optimiaztion System," in *Proceedings of the Conference on Supply Chain Management*, HongKong, 1995.

[6] NEVEM-workgroup, *Performance Indicators in Logistics*, IFS Publications, Springer-Verlag, 1989.

[7] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms", in *IEEE Transaction on knowledge and data Engineering*, Vol 10, No. 2, 1998, pp. 673-685.

[8] FIPA '97 Specification Part 2, "Agent Communication Language," http://drogo.cselt.stet.it/ufv/leonardo/fipa/spec/fipa7112.zip

[9] B. Fordhan, K. Kalpakis, and Y. Yesha, "Extending KQML for Inter-Agent Negotiation," *Master Thesis*, Computer Science Department, University of Maryland Baltimore County, May, 25,1995.

[10] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts,* Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing 1997.