Integrating Multi-agent Negotiation to Resolve Constraints in Fulfilling Supply Chain Orders

Fu-ren Lin
Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan 804 R.O.C

You-yu Lin
Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan 804 R.O.C

Abstract

As the order fulfillment process (OFP) in supply chains shifts to outsourcing paradigm, the OFP performance relies on the coordination among supply chain partners to reach executable and effective plans. The coordination of OFP among supply chain partners can be viewed as a distributed constraint satisfaction problem (DCSP). This paper enhances the existing methods to solve the DCSP by adding the multi-agent negotiation mechanism. We evaluated its enhancement on the OFP in the context of the metal industry via experiments and compared it with the centralized coordination mechanism. The results show that the integrated system outperforms the existing distributed constraint satisfaction algorithms in various demand patterns.

Keywords: order fulfillment process, supply chain management, distributed constraint satisfaction problem (DCSP), negotiation, multi-agent systems.

1. Introduction

A supply chain is a network of suppliers, factories, warehouses, distribution centers and retailers where the raw materials are acquired, manufactured to products, which then are delivered to consumers. The increase of customer expectations in low cost and high quality services has added the premium to effective supply chain reengineering. Many efforts have been endeavoring to improve the supply chain performance to achieve high agility without increasing costs (Billington, 1994; Henkoff, 1994). Electronic data interchange (EDI) and distributed databases have been considered as the most important technical advancement that benefits supply chain performance. At present, the multi-agent system (MAS) is believed as one of the new information platforms for managing supply chain activities. By virtue of an agent's autonomy and information sharing, the dynamics of business processes in a supply chain can be better handled through information exchange, negotiation, and solution resolution among agents. Because a physical multi-agent system operates on the Internet, supply chain partners can cooperate in a more open and dynamic environment than the traditional EDI-based environment.

Internet and agent technologies have changed the way connecting suppliers, manufacturers, distributors, and customers. The Internet enables a shift from individual business processes toward a more distributed, collaborative business model. For example, Rosetta Net, an organization responsible for establishing industry message formats, designs Internet communication mechanisms to streamline order fulfillment process through the supply chain.

Besides new technology and standards, by transforming the order fulfillment problem in a supply chain into a distributed constraint satisfaction problem, we anticipate to taking advantage of the existing solutions for distributed constraint satisfaction problems to resolve order fulfillment scheduling conflicts in a supply chain. Moreover, we are aiming to incorporate negotiation techniques in reaching

globally executable order fulfillment plan.

The objective of this paper is to propose and evaluate a multi-agent coordination mechanism in enhancing supply chain's agility. This approach mainly integrates two methods: *distributed constraint satisfaction algorithms* (DCSA) and *peer-wise negotiation*. The order fulfillment process in a supply chain is formalized as a distributed constraint problem, where each agent embedded in a supply chain partner's information systems maintains its local view of its supply chain status using constraint networks. In resolving the distributed constraints existing between agents, we design a peer-wise negotiation mechanism to reach common agreed states through agent communication protocols.

The performance of the proposed methods is evaluated by the resulting order cycle time, order fulfillment rate, and inventory. Comparing with centralized constraint satisfaction mechanisms which tend to reach the best results but are infeasible in real world distributed supply chain, we anticipate to identifying the feasibility and effectiveness of the proposed methods for improving supply chain performance in various circumstances, such as order arrival patterns.

2. Modeling the OFP in Supply Chains as the DCSP

2.1 The OFP in supply chains

A supply chain involves complex coordination and decision-making processes across organizational boundaries. It expands the scope of the organization being managed beyond the enterprise level to include inter-organizational relationship (Malone, Yates and Benjamin, 1987). An order fulfillment process starts with receiving orders from customers and ends with delivery of the finished goods. A manufacturing practice is shifting toward the outsourcing paradigm; activities of a business process may take place across different companies, which hinder the centralized planning and scheduling. It becomes imperative to integrate the order fulfillment process (OFP) into supply chain to improve the OFP. Some research using multi-agent systems to model supply chains take OFP as the basic research process (Strader, Lin, and Shaw, 1998). Figure 1 is an example of supply chain of Taiwan metal industry.

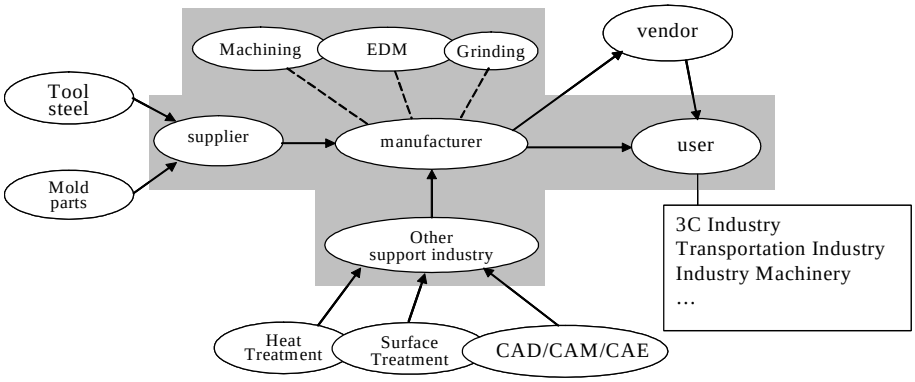


Figure 1. An example of supply chain: Taiwan metal industries

In the past, supply chain is usually coordinated on the EDI-based information infrastructure. EDI (Electronic Data Interchange), an information exchange topology centralized by the VAN (Value Added Network) center, provides the transfer method for electronic transactions. EDI involves the direct routing of information from one computer to another without interpretation or transcription by people, and to achieve this, the information must be structured according to predefined formats and rules,

which a computer can use directly (Holland, Lockett and Blackman, 1992). Because of the lack of standard, organization must agree on the translation software and data format provided by VAN centers.

Several solutions have been proposed to relax the strict constraints of EDI on its limited connectivity and scalability by virtue of the Internet. For example, XML-based information exchanges through the Internet advocated by many industry wide consortiums, *e.g.*, Rosetta Net, Covinst, Transora, etc. The gradual prevalence of Web services technologies further automate the connection of business processes among information systems owned by different companies through standard information formats, such as UDDI, WSDL, SOAP, and BPEL4WS. Besides that, a company that is gifted by autonomous capabilities but still wants to maintain coherent supply chain processes, demands more dynamic coordination facilitation. The agent-based approach greatly benefits Internet-based supply chain coordination.

2.2 Modeling the OFP in a supply chain as the DCSP

A constraint satisfaction problem (CSP) is a problem to find a consistent assignment of values to variables. A typical example of a CSP is a puzzle game called *n*-queens. This type of problems is called a constraint satisfaction problem since the objective is to find a configuration that satisfies the given constraints. Various applications about distributed CSP (DCSP) have been investigated, for example, distributed source allocation problems, distributed scheduling problems, distributed interpretation tasks, and multi-agent truth maintenance tasks (Yokoo, 2001).

CSP is formally defined as m variables $X_1, X_2 \ldots X_m$, taking their values from domains $D_1, D_2 \ldots D_m$ respectively, and a set of constraints on their values. A constraint is defined by a predicate; that is, $P_k (X_{k1} \ldots X_{kj})$, where the constraint is a predicate which is defined on the Cartesian product $D_{k1} \times \ldots \times D_{kj}$. This predicate is true if and only if the instantiations of these variables satisfy this constraint. Solving a CSP is equivalent to finding an assignment of values to all the variables such that all constraints are satisfied.

A distributed CSP is a CSP in which variables and constraints are distributed among multiple automated agents. To solve a DCSP is to assign values to these variables that satisfy inter-agent constraints in order to achieve coherence or consistency among agents. Various application problems in multi-agent systems (MAS) that are concerned with finding a consistent combination of agent actions can be formalized as distributed CSPs (Yokoo, Hirayama, 2000). In modeling business partners of a supply chain as the combination of agents, this study views the supply chain coordination problem as the DCSP.

According to Yokoo and Hirayama's classification (2000), asynchronous weak-commitment search is the most popular distributed constraint satisfaction algorithm. Asynchronous backtracking has a weakness that the agent/variable ordering is statically determined. If the initial value of a higher priority agent is bad, the lower priority agents need to search solution space exhaustively to revise that bad initial value. In order to improve this weakness, the asynchronous weak-commitment search dynamically changes the agent/variable ordering, so that a bad initial value can be revised without performing an exhaustive search. Besides dynamic priority, the asynchronous weak-commitment search is basically identical to the asynchronous backtracking algorithm (Yokoo, 1995).

3. The Integration of Multi-agent Negotiation with the DCSA

3.1 Multi-agent negotiation

In a multi-agent system, negotiation is a key form of interaction that a group of agents make agreement mutually regarding their belief, goal, or plan. In many cases, agents need to negotiate because of limited available resources. The area of negotiation is broad and is suitable for use in different scenarios (Jennings, 2001). Jennings identified three broad and fundamental topics, negotiation protocols, objects, and strategies, for research on negotiation (Beer, et al., 1999).

The number of parties participating in the negotiation can classify negotiation into bilateral and multilateral negotiations. Distributive negotiation is a decision making process of resolving a conflict involving two or more parties over a single mutually exclusive goal (Lewicki 1997). The game theory describes this situation as a zero-sum game. Integrative negotiation is unlike distributive negotiation, which allows negotiator to exploit trade-offs among different issues, and there would have a strategy to find the win-win solution. The game theory describes the situation as a win-win game.

Based on the cooperation level, a negotiation is in the continuum of cooperation and self-interest. Without cooperation, self-interested negotiation also is called competitive negotiation, which happens between two self-interested agents (Weiss, 1999), and each agent tries to maximize its local utility. In the opposite, in cooperative negotiation, agents try to reach the maximum global utility that considers the whole utility of both sides (Zhang, Poderozhny, and Lesser, 2000).

Originating from the research of distributed artificial intelligence (DAI), a multi-agent system is defined as a loosely coupled network composed of many problem-solver entities that work together to find answers to problem that are beyond their individual capabilities or knowledge (Durfee, Lesser, and Corkill, 1989). A multi-agent system possesses the following characteristics (Jennings, Sycara, Wooldridge, 1998): (1) each agent does not complete capabilities to solve a problem, (2) these is no global control in the system, (3) data is decentralized, and (4) computation is asynchronous.

Interaction is one of the most important features of an agent. In other words, agents recurrently interact to share information and to perform tasks to achieve their goals. Researchers investigating agent interaction identify three key elements to achieve multi-agent interaction: (1) a common agent communication language and protocol, (2) a common format for the content of communication, and (3) a shared ontology.

TAEMS (a framework for Task Analysis, Environment Modeling, and Simulation) proposed by Multi-Agent Systems Laboratory, UMACC, constructs a task environment-oriented modeling framework that can work hand-in-hand with agent-centered approaches. TAEMS improves upon conventional task structures by adding such features as quantitative action characterizations, explicit models of local and remote interactions and mechanisms to represent the wide range of ways a particular task can be achieved.

3.2 FIPA and JADE

The Foundation for Intelligent Physical Agents (FIPA) is a multi-disciplinary group pursuing software standards for heterogeneous and interacting agents and agent-based systems (FIPA, 1997). This organization has made available a series of specifications to direct the development of multi-agent systems. JADE (Java Agent Development Framework) is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents. It includes two main products: a FIPA-compliant agent platform and a

package to develop Java agents. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase.

3.3 The integrated multi-agent negotiation system for solving DCSP

A constraint satisfaction process can be divided into two steps. The first step is called *local constraint satisfaction*, and the second step is called *global constraint satisfaction*. Local constraint satisfaction means that agent can resolve the constraint conflicts locally without affect other agents' actions. For example, an agent (manufacturer) can revise its manufacture schedule to resolve the constraint conflicts. Global constraint satisfaction means that the agent asks other agents to coordinate by modifying their states and actions in order to resolve conflicts. For example, an agent (manufacturer) finds the reason that it cannot accept a customer order because of the shortage of materials. This problem can be solved by asking its supplier to replenish immediately, or asking its customer to extend order due date.

Figure 2 is the multi-agent system architecture for supply chain coordination. Each agent has two databases and one rule base. The belief database stores the agent's local view of environment like supplier list, task structure and the committed order. The negotiation database stores the negotiation history and the constraint network used for DCSP. The local scheduler uses information in belief database to make a new feasible schedule. The coordination engine, the most important component of the agents, follows coordination rules to control the process of global constraint satisfaction.

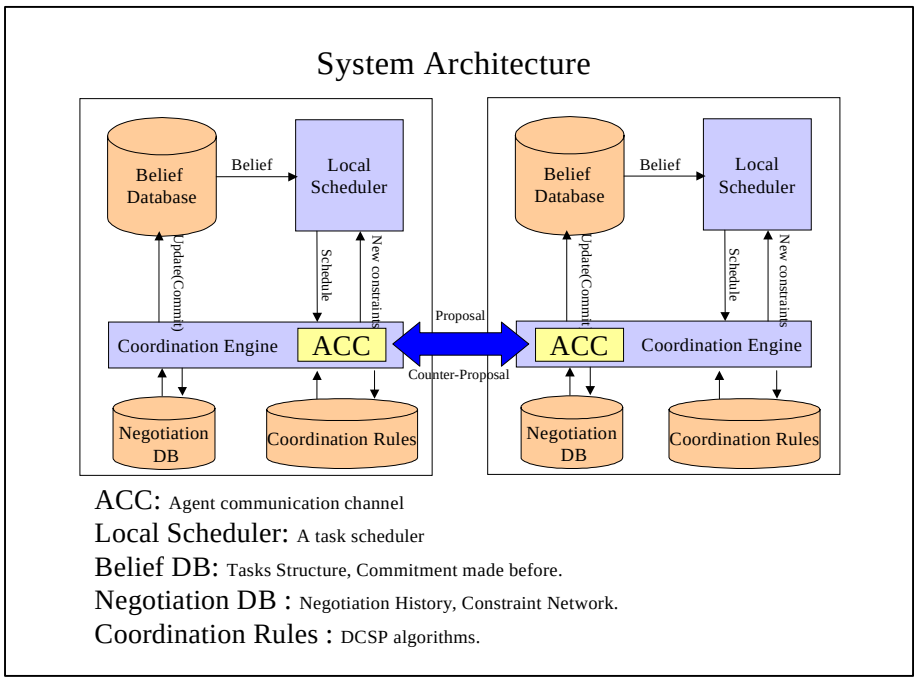


Figure 2. The architecture of multi-agent negotiation in solving supply chain coordination problems

In this research, we adopt the task representation of *TAEMS* to represent agent's task structure and apply all task interrelationships and quality accumulation functions. When an agent receives an order from customer, it will negotiate with customer for finding a feasible schedule that could satisfy customer's demand on cost, quality and cycle time. During the negotiation session, an agent (manufacturer) may not have the

accurate information of its suppliers, and the agent may negotiate with its customer by using the prior supplier information. Such historical information may be obsolete for resolving conflicts in fulfilling the order.

We combine negotiation and the concept of distributed CSP to resolve global constraints in a supply chain. We adopt the asynchronous weak-commitment (AWC) search, a DCSP algorithm, to resolve the global constraint in supply chain. In the AWC search, the priority of each agent is dynamically changed, so that an inferior decision can be revised without performing an exhaustive search.

In AWC, an agent uses two messages to communicate with others. "*ok?*" message is used to deliver the current value and priority value while "*nogood*" message is used to notice a new constraint. A "*nogood*" is a subset of an agent's view, where the agent is unable to find any consistent value with the subset. "*nogood*" is a constraint that reduces the solution space, when there is no solution in possible solution space, the algorithm will terminate.

An agent changes its assignments if its current value assignment is inconsistent with the higher priority agents, the agent generate a new constraint (called a "*nogood"*), and communicates the "*nogood*" to a high priority agent, and the agent's priory increases.

In real world supply chain, a company needs different expertise from different companies to deliver the product to its customers. It may negotiate with potential partners in such issues as price, quality, and cycle time, to reach agreement to acquire components for final product. In such an environment, the AWC algorithm can be extended to include negotiation mechanisms in resolving constraints imposed by partners.

Let's consider a situation that an agent (manufacturer) performs a non-local task, and at first it finds a potential supplier that can perform the non-local task and then make a proposal on the basis of its schedule which is made on the historical information of the non-local task. Because the historical information may be obsolete, *e.g*., a supplier's capacity may have been committed to other tasks, to respond to the agent's request, the supplier searches may find no solution satisfying the agent's demands.

According to the AWC algorithm, when an agent (supplier) cannot find a feasible value to satisfy the higher priority agents' constraints, it should increase its priority and send a "*nogood*" message to the connected higher priority agents to inform them that this value assignment conflicts and cannot find alternative values. After sending the "*nogood*" message to these connected agents, the agent increases its priority to be higher than its neighbor agents. In this example, a supplier will decide its value assignment and sends it to neighbor agents. Because a supplier's priority value is higher than its neighbors now, current value assignments would be a new constraint for the manufacturer. The manufacturer must update its value assignment under the constraint reflected from its supplier and customer.

If an agent owns the ability to negotiate, it may find a better agreement between a manufacturer and a supplier than that entirely compromising supplier's constraint. If no agreement is reached through negotiation, by following AWC algorithm, an agent with a lower priority accepts the offered solution from the high priority agent. Figure 3 illustrates the concept of combining AWC and negotiation.

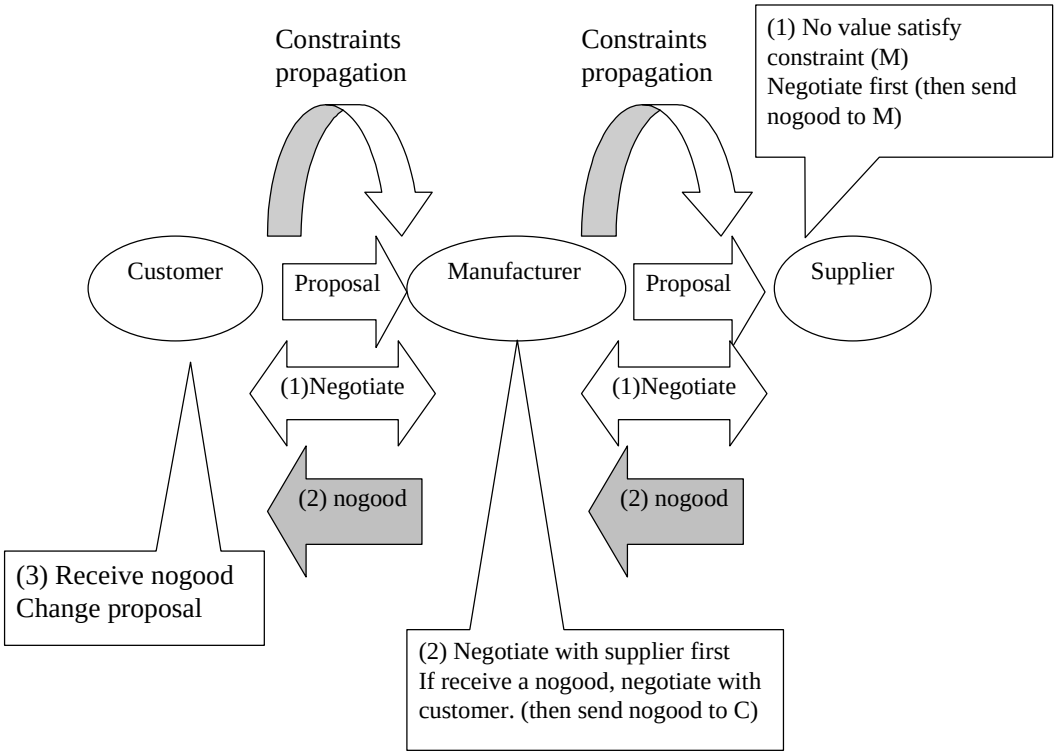


Figure 3. Global constraint satisfaction by integrating AWC and negotiation

4. Experimental Design

In order to evaluate efficiency of the proposed model for supply chain, we strive for designing an experimental environment that is similar to the real world supply chain environment. We use JADE, a multi-agent system design tools, to implement supply chain system embedded with the constraint satisfaction model.

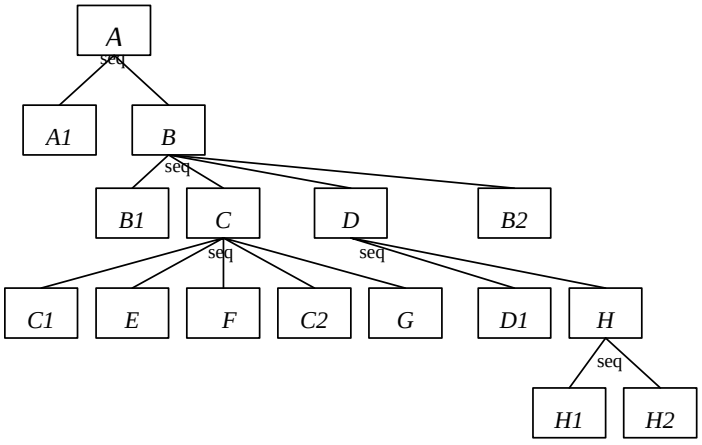
4.1 Experimental settings

We built the multi-agent system using JADE for an experimental OEM supply chain by referencing Taiwan's mental industry. Because it is a highly trusted, inter-dependent partner relationship, the main issue of order fulfillment process is to find a global feasible schedule to satisfy customers' requests. To simplify the problem in order to emphasize the constraint satisfaction problem, we only consider the time issue, and ignore cost and quality issues.

In the experiment system we built, there is no competitor to compete for receiving outsourcing tasks. The issue of outsourcing is not on selecting suppliers, but on generating executable schedules in executing tasks.

(1) Global parameters and variable setting

In experiments, ten companies (*A, B, C,…, J*) compose the experimental supply chain, which makes three types of products (X, Y, and Z). Each product needs *n* companies in the supply chain to finish the product. For example, product X made by the process $A \rightarrow B \rightarrow C$ needs three companies, and product Y made by the process $C \rightarrow A \rightarrow B \rightarrow D$ needs four companies to finish the product. Figure 4 is an example manufacturing process to produce molds.



A: PC OEM/ODM manufacturer
- *A1*: component design
- *B*: Mold manufacturer
- *B1*: mold parts design
- *B2*: mold assembly & mold test

C: Parts machining
- *C1*: Wire cutting
- *C2*: Milling

D: Surface treatment
- *D1*: Polishing

E: Heat treatment

F: EDM

G: Finishing

H: Coating
- *H1*: Acid wash
- *H2*: Cr coating

Figure 4. An example of a manufacturing process to produce molds

One company is responsible for receiving orders from customers outside the supply chain. When a company accepts orders, it will start the order fulfillment process to integrate the supply chain to fulfill the order. The system time step is set to day to denote the time unit to produce products and the order deadline. We make additional two conditions to define supply chain settings. First, each agent adopts the FCFS scheduling policy that an agent schedules tasks as early as possible in order to reduce the cycle time. Second, in order to create variations to simulate real world supply chain partners, we suppose that each agent participates in other supply chains, so that the state of an agent is changed both by the focal supply chain and other supply chains. A partner may accept orders from another supply chains, which may consume its limited capability. We set a variable called random_order_outside to 0.1 to denote that an agent has probability 0.1% to accept an order from other supply chain partners each day. An agents accepting order outside will randomly reduce three days production capabilities in 100 day.

(2) Local parameters and variable setting

An agent's capability is fixed per day. A company has its local task tree to produce products. A local task tree, a part of global task tree, records a company's manufacturing variables to perform tasks. We predefine the importance rank value for each company. In this setting, the company participating in more products manufacturing process has higher importance rank because it is hard to acquire resources from them. Table 1 lists the values set for the global environment.

Table 1. Global parameters for the experimental setting

Parameters	Value
Main issue	time
System time unit	day
Probability of receive order from another supply chain per day	10% per day to receive order occupying its 3 day capability (Random in 100 day)
Number of companies	10 (*A*, *B*, *C*…. *J*)
Number of products	3 (X, Y, Z)
Average manufacturing time of Product X	25
Average manufacture time of Product Y	20
Average manufacturing time of Product Z	15
Number of Product X manufacturing processes	6 (*B*, *C*, *E*, *G*, *H*, *J*)
Number of Product Y manufacturing processes	6 (*A*, *E*, *F*, *G*, *I*, *J*)
Number of Product Z manufacturing processes	5 (*A*, *C*, *D*, *E*, *H*)
Schedule rules	FCFS

(3) Random order generation setting

An order generator randomly generates an order for one kind of products with the order arrival date and the expected shipping date, and use three parameters to adjust and generate orders. The first parameter is the order inter-arrival time to control the frequency of order arrival. The second is order quantity to randomly generate the order quantity. In order to differentiate the fulfillment difficulty of each order, a formula to generate the expected shipping date is defined as *expected shipping date = order arrival date +*(*average manufacturing date of a product * order quantity*)* *variation allowance.* Variation allowance is a random variable ranging between 1 and 2. When the allowance value moves toward 1, it means that it is getting urgent to fulfill orders.

4.2 Experimentation

We use the order fulfillment process to evaluate the constraint satisfaction model. This experiment tests the efficiency of the model in four different demand order arrival patterns. Two parameters, order quantity and order inter-arrival time, are used to generate demand orders. We design four experiments by changing different setting of order quantity and order inter-arrival time. Table 2 lists the settings of the four experiments.

Table 2. Experimental setting of the experimentation

	Demand pattern	Order quantity	Order inter-arrival time
Experiment A	High order inter-arrival time variation and normal order quantity variation	Rand (1~3)	Rand (1~17)
Experiment B	Normal order inter-arrival time variation and normal order quantity variation	Rand (1~3)	Rand (1~13)
Experiment C	High order inter-arrival time variation and high order quantity variation	Rand (1~5)	Rand (1~17)
Experiment D	Normal order inter-arrival time variation and high order quantity variation	Rand (1~5)	Rand (1~13)

* rand(*a*~*b*) denotes the random number generated by uniform distribution between *a* and *b*.

In each experiment, according to the experiment settings, the order generator generates 50 orders. Before the experiment starts, we randomly feed five orders into supply chain. After the warm up time, we start to feed these 50 orders into the supply chain according to order_arrival_time. Supply chain performance is analyzed from ten-time experiments. A contrast experiment is used as a benchmark to test the negotiation model's performance. We assume that ten companies *A, B, C,…, J* belong to an enterprise, and an agent for the enterprise can build a global task tree and schedule the production schedule for these companies by pooling complete capacity information from each company. To generate executable global schedules is viewed as a centralized constraint satisfaction problem. The global agent's experimental results serve as the benchmark to compare with the distributed cooperative model.

5.3 Four evaluation criteria

Experimental results are measured in four criteria: *order fulfillment rate*, *cycle time*, *WIP (Work-In-Process) inventory cost*, and *final product inventory cost*. Order fulfillment rate denotes the ratio that arrival orders can be finished by the due dates. Cycle time represents the duration from the order received to the finished final product delivered. WIP inventory cost denotes the cost to store work-in-process materials, which may be caused by unconcatenated tasks executed by different companies. For example, it is not allowed for an agent to adjust the schedule after its suppliers accept its proposal by returning "*ok*" message. If *A* wants *B* to perform a task from 10^{th} to 40^{th} day, *B* says "ok, I can do it for you from day 15^{th} to 40^{th}." There will be a WIP inventory cost from day 10^{th} to 14^{th}. The final product inventory cost is calculated by counting the days after the final product is finished before the customer's due date.

5. Evaluation and Discussions

5.1 Evaluation results of the distributed coordination model

We list the difference of OFP performance between centralized and distributed coordination methods (*i.e.*, performance outcomes of centralized minus distributed methods). We normalize the degree of performance gap between centralized and distributed CSPs as shown in Table 3. The normalization formulas are shown as follows.

Gap of order fulfillment rate =

$$\frac{CCSP\ order\ \ fulfillment\ rate - DCSP\ order\ \ fulfillment\ rate}{CCSP\ \ order\ \ fulfillment\ \ rate}$$

$$\text{Gap of cycle time rate} = \frac{CCSP\ cycle\ time - DCSP\ cycle\ time}{CCSP\ cycle\ time}$$

$$\text{Gap of WIP inventory cost} = \frac{CCSP\ WIP\ inventory\ \cos t - DCSP\ WIP\ inventory\ \cos t}{CCSP\ WIP\ inventory\ \cos t}$$

$$\text{Gap of inventory cost} = \frac{DCSP\ inventory\ \cos t - CCSP\ inventory\ \cos t}{DCSP\ inventory\ \cos t}$$

The performance of the distributed coordination method approaches to that of the centralized method in experiments A and B. It indicates that the distributed coordination method is more suitable for the OFP in a supply chain with less demand pattern variation. As the demand pattern variation increases, the performance gap increases.

There is a relation between cycle time, WIP inventory cost and inventory cost. The cycle time is the sum of real manufacturing time and WIP inventory transfer time. The duration from order arrival day to shipping date is the sum of cycle time and the waiting time for final product inventory sitting in the warehouse. Therefore, these three criteria are correlated.

In WIP inventory cost, the performance gap in experiments A and B (product quantity 1~3) is less than that in experiments C and D (product quantity 1~5). This performance gap also indicates that the distributed coordination method exhibits less performance gap from benchmarks in relatively stable demand patterns.

Inventory cost is calculated by counting days after the final product is finished and before the customer's expected shipping date. The results show that the centralized method results in more inventory cost than the distributed method because the former has generated a more compact schedule that keeps the cycle time shorter than that from the distributed method. Therefore, those early finished products sit in the warehouse to wait for customers to pick up by the expected shipping dates. It also shows that the final inventory cost resulting from these two methods is much closer in experiments A, B, and C. This indicates these two methods have similar outcomes in final product inventory in the stable demand patterns.

Table 3. The comparison of normalized performance gap

	Order fulfillment rate	Average cycle time	Average WIP inventory cost	Average Inventory cost
Experiment A	0.0494	0.0596	0.8229	0.1338
Experiment B	0.0322	0.0655	1.3901	0.187
Experiment C	0.0574	0.0527	0.3464	0.2139
Experiment D	0.1236	0.0881	0.4167	0.8059

5.2 The explanation of experimental results

Since the centralized schedule as the benchmark indicates the optimal schedule for the order fulfillment process, we anticipate that the proposed distributed constraint satisfaction methods can approximate in certain degree to the optimal result. The gap between these two methods draws us to identify the usability of the distributed coordination methods under different demand environments. The following reasons

address the possible explanation of these outcomes.

The first reason is that, in distributed coordination model, when an agent negotiates with others for time to schedule a feasible partial solution, the agent cannot make a local optimal solution though the simplified negotiation process. Because lack of precise knowledge about the product and negotiation strategy, a contractor may propose a large time slot for contractees to choose its suitable time. The redundant time slot of the time range in the proposal causes the WIP inventory and makes capability more fragmented. The fragmented time slots make backtracking happen more frequently and make the distributed approach harder to find an optimal solution.

The second reason is the schedule ability of local scheduler. With the simplified scheduling process, an agent may not adjust the schedule to find a better local solution after it receives all responses from other agents and once constraints are satisfied.

The third reason is that the nature of DCSP just focuses on finding a feasible solution. An agent for solving DCSP just checks if the partial solution is consistent, but it does not check if this solution can be further improved to obtain better results. The DCSP method satisfies the global solution in a distributed way, but in each backtracking domain, the partial solution is not optimal and the combined solution is not optimal either. Not only local optimal solutions cannot achieve, but also, bad partial solutions may aggregate to even worse global solutions. This phenomenon will constantly happen to make each agent's capability more fragmented, where fragmented capability leads DCSP harder to find an optimal partial solution. It is a vicious circle.

Because of commerce secret and enormous cost of gaining all information, centralized model is impracticable in the real supply chain. The distributed coordination model is still a practicable solution that exhibits stable performance for supply chain.

6. Conclusions

This study proposes an agent-based cooperative model for supply chains to commit orders by satisfying constraints. Due to the limitation of the real world environment, the centralized schedule model to handle constraint satisfaction is impractical, it is important to excise the distributed constraint satisfaction model to meet the outsourcing paradigm of supply chain management.

We propose a distributed constraint satisfaction model integrating the negotiation mechanism to conduct the order fulfillment process in a supply chain. In order to evaluate the proposed distributed coordination method, we design and conduct experiments to compare the OFP performance with the centralized method in four criteria under different degrees of order inter-arrival time and ordering quantity. The results have shown that that the OFP performance of the proposed distributed coordination method approaches well to that of the centralized method which may renders more optimal performance.

But in a pity, we found that the performance gap between these two methods exhibits significant difference using *t*-test. As the demand pattern variation increases, the performance gap also increases. It indicates that the distributed method may be more applicable to less variant demand environment. The results lead us to identify the possible cues that explain the failure of the distributed method to reach better results. We refer the performance gap to the failure of local optimum caused by the primitive negotiation strategy, and the nature of constraint satisfaction problem, which is not aiming to seek the optimal schedule.

Acknowledgment

The authors are grateful to the following sponsors in this research: the Ministry of Education (Program for Promoting Academic Excellent of Universities under the grant number A-91-H-FA08-1-4), National Science Council (NSC92-2416-H-110-001), and Metal Industry Research and Development Center.

References

Beer, M., d'Inverno, M., Luck, M., Jennings, N. R., Preist C. and Schroeder, M., "Negotiation in Multi-Agent Systems," *Knowledge Engineering Review, Vol.* 14, No.3, pp. 285-289. 1999.

Billington, C., "Strategic Supply Chain Management", *OR/MS Today*, pp. 20-27, April 1994.

Durfee, E.H., Lesser, V.R. and Corkill, D.D. "Trends in Cooperative Distributed Problem Solving," *IEEE Transactions on Knowledge and Data Engineering*, March, KDE-1(1), pp. 63-83. 1989.

FIPA "FIPA 97 Specification," *Foundation for Intelligent Physical Agents*, Version 1.2, October 10, 1997.

Henkoff, R. "Delivering the goods," *Fortune*, 140(11), November 28, 1994, E34-47.

Holland, C., Lockett, G. and Blackman, I. "Planning for Electronic Data Interchange," *Strategic Management Journal*, Vol. 13, No. 2, pp.153-166, 1992.

Jennings, N.R., "An agent-based approach for building complex software systems" *Communications of the ACM*, 44 (4) 35-41, 2001.

Jennings, N.R., Sycara, K. and Wooldridge, M. "A Roadmap of Agent Research and Development," *Autonomous Agents and Multi-Agent Systems Journal, Kluwer Academic Publishers, Boston*, Vol. 1, No. 1, pp. 275-306, 1998.

Malone, T. W., Yates, J. and Benjamin, R. I., "Electronic Markets and Electronic Hierarchies", Communications of the ACM Vol.30, No.6, pp.484-497, 1987.

Strader, T. J., Lin, F.-r., and Shaw, M.J., "Simulation of Order Fulfillment in Divergent Assembly Supply Chains", *Journal of Artificial Societies and Social Simulation*, Vol.1, No.2, March 1998.

Weiss, G., "Distributed Rational Decision Making," *Multi-agent System: Chapter 5, MIT press*. 1999.

Yokoo, M., Hirayama K., "Algorithms for Distributed Constraint Satisfaction: A Review", *Autonomous Agents and Multi-Agent Systems Journal*, Vol. 3, No.2, pp.198-212, 2000.

Yokoo, M., "Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems". In *Proceeding of the First International Conference on Principles and Practice of Constraint Programming* pp. 88-102. *Lecture Notes in Computer Science*. 976. 1995.

Zhang, X., Poderozhny, R., Lesser, V., "Cooperative, MultiStep Negotiation Over a Multi-Dimensional Utility Function," In *Proceeding of the IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000)*, pp. 136-142, July 2000.