2007

Real-Time Supply Chain Control via Multi-Agent Adjustable Autonomy

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Lucas Agussurja

Ramesh Thangarajoo

DOI: https://doi.org/10.1016/j.cor.2007.01.027

Real-Time Supply Chain Control via Multi-Agent Adjustable Autonomy

Hoong Chuin LAU [a,*], Lucas AGUSSURJA [b], Ramesh THANGARAJOO [b]

[a] *School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902*

[b] *The Logistics Institute Asia Pacific, National University of Singapore, 11 Law Link, Singapore 119260*

Abstract

Real time supply chain management in a rapidly changing environment requires reactive and dynamic collaboration among participating entities. In this work, we model supply chain as a multi agent system where agents are subject to an adjustable autonomy. The autonomy of an agent refers to its capability and capacity to make decisions within a multi-agent system. Adjustable autonomy means changing the autonomy of the agents during runtime as a response to changes in the environment. In the context of a supply chain, different entities at different points of time will have different autonomy levels. We propose a centralized fuzzy framework for translating the changes in the environment to the changes in autonomy levels of the agents. A coalition-formation algorithm based on the autonomy levels is then applied, allowing the agents to collaborate to cope with the changes while achieving global consistency. Our proposed framework is applied to two supply chain control problems with drastic changes in the environment: one in controlling a military hazardous material storage facility under peace-to-war transition, and the other in supply management during a crisis (such as bird-flu or terrorist attacks). Experimental results show that by adjusting autonomy in response to environmental changes, the behavior of the supply chain system can be controlled accordingly.

Key words: Crisis management, Fuzzy controller, Military warehouse, Multi-agent system, Real-time control

* Corresponding author. Tel.: +65-68280229; fax: +65-68280919

Email addresses: hclau@smu.edu.sg (Hoong Chuin LAU), tlila@nus.edu.sg (Lucas AGUSSURJA), tlitr@nus.edu.sg (Ramesh THANGARAJOO).

1 Introduction

Real-time management of supply chains in a rapidly changing environment entails effective, efficient and reactive collaboration among participating entities. This is a challenging task because the parties involved in the supply chain have their own resources and objectives. A centralized authority controlling participating entities is likely to be costly and ineffective.

In this work, we model supply chains as multi-agent systems. A multi-agent approach allows reactive, real-time collaborative decision making. Decisions can be made asynchronously and can be performed by the most suitable partners, thereby allowing the overall supply chain system behavior to react dynamically according to the changing environment. The similarities of supply chain and multi-agent systems are given in table 1.

An agent's autonomy refers to its capability and capacity to make autonomous and independent decisions within the multi-agent system. Adjustable autonomy means changing the level of autonomy of the agents during the running time of the system. The changes can be caused by: (1) the dynamic environment (e.g. an accident occurs, and the agent loses its resources, thus reducing its autonomy level), or (2) human interference (e.g. in a critical system like a missile control system, at some point of time, the autonomy of the agent needs to be reduced to zero, and decision making is taken over completely by humans). In reactive, real-time multi-agent system that interacts with the dynamic environment, adjustable autonomy is unavoidable, either between agents or between the agents and humans.

In this work, we are interested in the relative autonomy levels among agents where the changes are caused by the changing environment. Each agent has its own objective which also changes dynamically, and each agent has the full capability to change its own objective. In our model, we assume that the autonomy levels can be decided by a central authority, the agents are fully cooperative, and information flow is reliable. We propose an efficient approach based on a fuzzy selection process to translate the changes sensed in the environment to the autonomy levels of the agents. Furthermore, we propose a coalition-formation based algorithm that allows the agents to negotiate based on their autonomy levels to achieve a global consistency.

We apply this approach to two supply chain control problems: one in controlling a military hazardous material storage facility under peace-to-war transition, and the other in supply management during a crisis (such as bird-flu or terrorist attacks). Experimental results show that by adjusting autonomy in response to environmental changes, the behavior of the supply chain system can be controlled accordingly.

2 Related works

The following is a list of proposed agent frameworks for implementing supply chain management:

- **Integrated Supply Chain Management**[6]. In this work, each agent performs one or more supply chain functions and coordinates its decisions with other relevant agents. There are

Table 1
Similarity between Supply Chain and Multi Agent Systems

Nature of Supply Chain	**Nature of Multi Agent**
A supply chain consists of multiple parties working on multi stage tasks	A multi agent system consists of different types of agents with different roles and functions
Each entity in a supply chain has its own objectives, capabilities, performs certain tasks, and follows certain rules	Agents have their own objectives, resources tasks, and decision rules specified by the user
There is a need to coordinate material, information, and financial flows between and among all the participating entities	Agents coordinate with each other through communication and interaction, in a network
Information is distributed. Each entity has incomplete information. Information need to be shared across organizational, functional and system boundaries	Agents can communicate with human, with other information systems, and with other agents. They can share information and knowledge through message exchange between agents
There is no single authority. Knowledge is distributed among members in supply chain. Decision making in supply chain is through multiparty negotiation and coordination	Agents are autonomous. They are responsive to monitor the changing environment, proactive to take self-initiated action, and social to interact with humans and other agents
Learning and reasoning is needed for supply chain members to make individual or joint decisions for operating and planning	Intelligent agents are capable of reasoning based on the rules given by the the user or knowledge learned from an open environment
The structure of the supply chain is flexible. It can be organized differently to implement different strategies	Agent systems are flexible. Agents can be organized according to different control and connection structures
Tasks in the supply chain can be decomposed to subtasks or multiple tasks can be composed to a larger function	An agent can delegate its task to another agent or coordinate with other agents' tasks to form a higher level system
Supply chain is dynamic. Entities may join or leave a supply chain	Agents can be created or discarded from a multi-agent system

two types of agents: functional agents and informational agents. Functional agents include order acquisition agent, logistics agent, scheduling agent, resource agent, dispatching agent and transportation agent. They plan and/or control activities in the supply chain. Information agents support other agents by providing information and communication services. This project address coordination problems at the tactical and operational levels.

- **A Negotiation-based Multi-agent System for Supply Chain Management**[3]. In this framework, functional agents can join in, stay, or leave the system. The Supply Chain Management System functionality is implemented through agent-based negotiation. Virtual supply chain may emerge from the system through automated or semi-automated negotiation process between the agents.

- **Mediator Agent for Enterprise Integration**[12]. Here, a hybrid agent-based mediator-

centric architecture is used to integrate the manufacturing enterprise activities such as design, planning, scheduling, simulation, execution and product distribution with those of its suppliers, customers and partners into an open, distributed environment.

- **Collaborative Agent System Architecture**[13]. Here, a collaborative agent system architecture is proposed in a complex supply chain network. The system is composed of collaborative elements classified as agents, local area coordinators, yellow pages, and cooperative domain servers. The objective of this work is to support collaborative agents softwares by providing domain independent communication and cooperative services over the internet.

Research at modeling dynamic supply chain management policies by means of multi-agents include [5,8]. In [8] for example, the supply chain model is formulated as a discrete resource allocation problem under a dynamic environment, and a virtual market concept with multi-agent paradigm is applied to the model.

In the domain of agent autonomy, various factors entail changes of agent autonomy over time. In some instances, the agent might lack the capability to make decisions [7]. Others tend to make provisions for human intervention to cater for unexpected events where the agent may act irrationally [4]. Reduced confidence in a fully automated system is a common reason where some crucial decisions require user interference. [11] studies the possibility of transferring autonomy for higher quality decisions with minimizing coordination problems. While agent-to-agent transfer of autonomy is considered, there is little work on the needs for such a transfer and the practical advantages and implications of such a strategy. Most of the mentioned work have focused on a single agent and a single user. While [11] has expanded beyond the one-to-one relationships, the author limits the transfer to a complete transfer of autonomy and does not consider situations where some autonomy may be retained. [1] proposes 3 discrete levels of autonomy: "Command-Driven" where the agent does not make decisions and must obey orders, "Consensus" where the agent shares decision making equally with other agents and "Locally Autonomous" where the agent makes decisions alone. In this paper, we consider a real-life military situation where there is a need to vary the autonomy gradually and continuously to suit changing circumstances.

Another area of research closely related to our work is the Dynamic or Distributed Constraint Satisfaction Problem (DCSP) as a formalism for modeling multi-agent coordination and negotiation. For Dynamic CSP, constraints can change over time. [2] developed methods for adapting consistency computations to such changes, and [10] solved this problem by seeking minimal change on the solution. [15] extended this problem to DCSP as well and proposed the ADOPT algorithm which guaranteed a user-defined optimality. [9] also provided an algorithm (OptAPO) to solve DCSP, which experimentally performs better than ADOPT. The coalition formation algorithm presented in this paper is inspired by the above existing algorithms for solving DCSP.

3 Problem definition

Essentially, the problem we study can be divided into two parts, i.e. sense and response. The sense portion deals with correctly translating all sensing data gathered by the agents to new local objectives and autonomy levels for each agent. The response portion is, given the new

objectives and the autonomy levels of the agents, how the agents should interact with each other to respond to the environment.

Let N be the index set of n agents, and each agent $i \in N$ holds a set of mutually exclusive variables $A_i = \{a_{i1}, a_{i2}, ..., a_{il_i}\}$. The domains for the variables held by agent i are $D_i = \{D_{i1}, D_{i2}, ..., D_{il_i}\}$. The values that can be taken by the variables $a_{11}, a_{12}, ..., a_{21}, a_{22}, ..., a_{nl_n}$ are restricted by a set of global constraints $\mathbf{C} = \{C_1, C_2, ..., C_q\}$. Each agent i has a set of *objective functions* $F_i = \{f_i^1, f_i^2, ..., f_i^{p_i}\}$. Each function is defined as $f_i^k : \prod_{1 \leq j \leq l_i} D_{ij} \rightarrow \mathbf{R}$.

These agents exist in a dynamic environment that are captured by the dynamic environment variables. Let $\mathbf{E}$ be the set of m *environment variables* $\mathbf{E} = \{e_1, e_2, .., e_m\}$ that can be sensed by the agents. The domain for variable e_i is the set D_{e_i} of real numbers. Each agent senses some of the environment variables. Let the function $sensed : N \rightarrow 2^{\mathbf{E}}$ returns the set of variables that an agent can sense, equivalently $sensed(i) \subseteq \mathbf{E}$. This is to capture the idea that each agent has its own sensor, and able to sense its surrounding.

The sense problem is defined as, given the existing values of the environment variables in $\mathbf{E}$, find

(1) the objective that the agent must focus on, i.e. a mapping for each agent i, $\mathbf{M}_i : \prod_{1 \leq j \leq m} D_{e_j} \rightarrow F_i$;

(2) the relative autonomy levels of the agents, i.e. a mapping for each agent i, $\mathbf{L}_i : \prod_{1 \leq j \leq m} D_{e_j} \rightarrow [0, 1]$.

The response problem is defined as, given both the objective f_i and autonomy level al_i set for each agent i, find an assignment for the agent variables $(a'_{11}, a'_{12}, ..., a'_{21}, a'_{22}, ..., a'_{nl_n})$ which is consistent with $\mathbf{C}$ and minimizes the agent local objectives and the global objective which is defined in this work as the sum of the relative "errors" of all agents, i.e. minimizing

$$\sum_{i \in N} \left(al_i * \frac{f_i^* - f_i(a'_{i1}, a'_{i2}, ..., a'_{il_i})}{f_i^*} \right)$$

where $f_i{}^*$ is the value of the objective given the optimal assignment to the variables of agent i.

4 Solution approach

The overall solution approach is given as follows. For the sense problem, we propose a fuzzy framework to capture the relationships between environment variables and the agent objectives and autonomy levels. The role of the fuzzy framework is to continually capture the sensor data from all the agents and in turn translate these data into new objectives and autonomy level of the agents. Then, based on the objectives and the autonomy levels set, we apply a distributed branch-and-bound algorithm to obtain the solution for the response problem.

4.1 Fuzzy framework

The system behavior of a sense-and-response system is often difficult to design. Due to the adaptive nature required of the system, the interpretation of the inputs to the system such that a proper reaction can be designed is a complex task. In this paper, we propose a fuzzy selection process to determine the autonomy and objective of agents. Fuzzy system has been studied extensively in the literature, and has been applied to a wide range of real life applications. The advantages of using a fuzzy system are that it generally captures relationships between non-discrete data and handles incomplete data effectively.

The 5-stage fuzzy process listed below has been extensively reported [14]. In the following, we define the function $\mathbf{L}_i(e'_1, e'_2, ..., e'_m)$ as consisting of:

- **Normalization**, performs a scale transformation of the input metric into a normalized variable. The input variables obtained from the environment will have a wide range of values. These variables must first be normalized to values in the range from zero to one. This may seem to be a simple task but a poor normalization will affect the performance of the fuzzy controller. The objective in normalization would be to obtain values well distributed in the range from zero to one. Extreme values can be truncated to zero or one.

  For each input variable $e_j \in \mathbf{E}$, its domain $D_{e_j} = [0, r_j]$ is normalized into the interval $[0, a]$ for some value a, and the normalization function is:

  $$\mathbf{n}_j(e'_j) = \left(\frac{a}{r_j}\right).e'_j$$

- **Fuzzification**, determines the degree to which the normalized variables belong to each of the appropriate fuzzy sets via membership functions. Fuzzification is the process of assigning a degree of truth to the statement about the input variables. This will show for example the extent that the if statements are true. The number of fuzzy sets can be two or more for each normalized input variable. The first step would be to determine the number of fuzzy sets described in the statements. Since each fuzzy set can be sees as a direct translation from the description of the variables in the relevant application the number of fuzzy sets can similarly be determined in this way. For example a general low/high description will result in two fuzzy sets while a low low/medium/high description will result in 3 fuzzy sets.

  The more the fuzzy sets the better the description of the job shop would be and the fuzzy controllers performance will be enhanced. This can be related to the real world problem where a person gets a better idea when told the queue is long, medium or short rather than just long or short. However increasing the number of fuzzy sets beyond a certain level will not result in significant improvement and will unnecessarily complicate the subsequent computational step. Thus the number of fuzzy sets used in the design of the fuzzy solution is completely dependent on the minimum requirements for the problem. The triangular membership function is used as it is the simplest of the commonly used membership functions both in terms of understanding and computation. Modifying the membership functions used may show a difference in the results obtained.

  The domain $[0, a]$ of each of the normalized variable e_j can be divided into several states, called fuzzy sets: $I_j^1, I_j^2, ..., I_j^{s_j}$, let $\mathbf{I}_j = \{I_j^1, I_j^2, ..., I_j^{s_j}\}$. This applies to the output variable as well.

The partial functions $\mu_{I_j^k}$ for $1 \leq k \leq s_j$ measure the degree of membership of the normalized value to the set I_j^k. In the example of Figure 1, we have $\mu_{I_j^2}(0.5a) = 0.25$, $\mu_{I_j^3}(0.5a) = 0.75$, while $\mu_{I_j^1}(0.5a)$ and $\mu_{I_j^4}(0.5a)$ are undefined.

- **Fuzzy Inferencing**, a rule base, also known as the inference engine, links the fuzzy inputs to the fuzzy outputs via linked statements called rules. If there is more than one input then the input statements are linked by the 'or'or 'and'operator. The implication operator 'then'indicates the output statements. The 'and'operator is again used to link the output statements. Each of these output statements leads to a fuzzy set.

  The truncation level of the initial fuzzy sets determines the truncation of the output fuzzy sets. When there are more than one initial fuzzy set then the lowest level of the initial sets indicates the truncation of the output fuzzy sets.

  A rule is an if-then statement of the form:

  $$\begin{array}{c} \text{if } e_j \text{ is in } I_j^k \text{ and } e_p \text{ is in } I_p^q \text{ and } ... \\ \text{then } al_i \text{ is in } O_i^t. \end{array}$$

  for some p, q, t where al_i is the autonomy level of agent i, $O_i^t \in \mathbf{O}_i$, and $\mathbf{O}_i$ is the set of fuzzy sets for the output variable al_i. In other words, the if-part of a rule contains a set of fuzzy sets of the input variable, and the then-part of the rule contains a fuzzy set of the output variable. A rule will fire if $\mu_{I_j^k}(\mathbf{n}_j(e'_j))$ is defined for all I_j^k contained in the if-part of the rule. The result of firing a rule is the state (fuzzy set) that the output variable in and the degree of the membership which is $min\left\{\mu_{I_j^k}(\mathbf{n}_j(e'_j))\right\}$ for all I_j^k in the if-part of the rule.

- **Defuzzification**, the fuzzy sets for the output from different rules that fire are merged and the center of area of the combined fuzzy sets is found to get a defuzzified value called the centroid c_i. In this stage the fuzzy sets for each fuzzy output are merged and the center of area (centroid) of the combined fuzzy sets is determined to evaluate a defuzzified value. In the case of defuzzification the output fuzzy sets for each fuzzy variable are merged and various methods are used to defuzzify the result to obtain a single value.A popular method would be to find the centroid or center of area of the combined fuzzy set . An example can be seen in Figure 2.

- **Denormalization**, is the reverse of the normalisation process where the scaling biases are removed. The coefficients which are obtained for each agent, can be viewed as weights that have to be applied to determine the autonomy of the agent, i.e. $al_i = \left(\frac{1}{a}\right).c_i$

The value of a, the membership functions and the rules are defined based on empirical methods. $\mathbf{M}_i$ is defined in a similar way.

4.2 Coalition formation based algorithm

Here, we propose a coalition formation based algorithm for the agents to find the assignment to all the variables $(a'_{11}, a'_{12}, ..., a'_{21}, a'_{22}, ..., a'_{nl_n})$ that is consistent with $\mathbf{C}$ while optimizing the measure of objective described above, or to output no solution if such solution is not found.

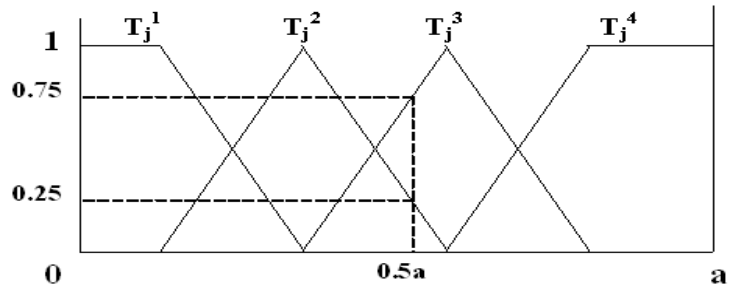

Fig. 1. Fuzzification

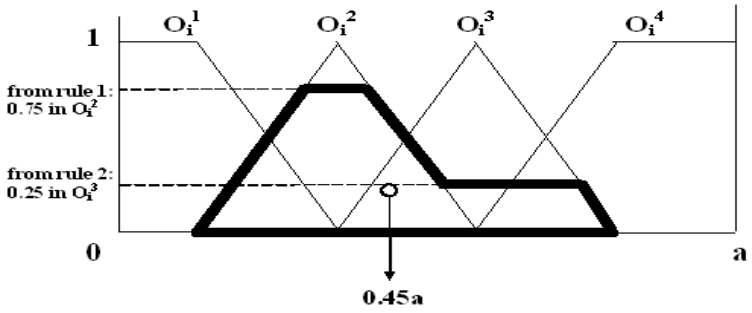

Fig. 2. Defuzzification

This algorithm is based on distributed branch-and-bound. The idea is to have the agents form coalitions, and to find local solutions. If these local solutions do not conflict each other with respect to **C**, then the solution is found. If conflicts between these local solutions occur, then the agents will try to form a bigger coalitions, and then find the consistent solutions within this bigger coalitions. In the worst case, all the agents will form one big coalition, and if consistent solution still can not be found, the algorithm will output no solution.

The algorithm is divided into 3 parts, namely initialization, coalition formation, and negotiation.

During intialization, each agent first constructs an optimal assignment to its variables using a branch and bound search that maximizes its given objective function f_i. This optimal value is then stored in the variable f_i^*. G_i is the set that stores all the agents that are currently in coalition with agent i. The set is initialized to i itself. The variable L_i indicates the agent that has the role as the leader in the current coalition of agent i, this agent is the one with the highest autonomy level in the coalition. The set K_i stores all the agents having conflict with agent i, initially set to empty. Variable $state_i$ indicates the state of the agent i. After initialization, the agents will enter into coalition formation stage, as shown in algorithm 1.

In coalition formation stage, each agent first checks for any external conflicts. The function NEIGHBORS(i) returns all the agents that share some constraints with agent i, and the function CONFLICT(i, j) check whether any two agents are in conflict, which is done through exchange of variables. After detecting conflicting neighbors, each agent will send invitations to all its conflicting neighbors to form a coalition(lines 6-7). Along with the invitation, the agent also sends the variable al_{L_i} which is the autonomy level of the leader of the current coalition, or the highest individual autonomy level in the set G_i. In lines 9-15, we see that An agent will tend to

ALGORITHM 1: Coalition Formation

```
FORM-COALITION()
1   ∀j ∈ NEIGHBORS(i), if (CONFLICT(i, j))
2               then K_i ← K_i ∪ {j}
3   n ← #K_i
4   if (n = 0)
5               then state_i ← no_conflict
6   else SEND(invite,al_{L_i}) to all j ∈ K_i
7               state_i ← conflict
8
9   when received(invite,al_{L_j}) do
10              if(al_{L_i} > al_{L_j} or state_i = forming_coalition)
11                      then SEND(reject,al_{L_i}) to j
12              else if(al_{L_i} < al_{L_j})
13                      then SEND(accept,al_{L_j}) to j
14                      SEND(leave,al_i) to all agent k ∈ G_i
15                      state_i ← forming_coalition
16  when received(reject,al_{L_j}) do
17              n ← n − 1
18  when received(accept,al_{L_j}) do
19              G_i ← G_i ∪ {j}
20              L_i ← j where al_j ≥^{max}_{k∈G_i} {al_k}
21              SEND(confirm,al_i,G_i) to all j ∈ G_i
22              n ← n − 1
23  when received(confirm,al_j,G_j) do
24              if (G_i ≠ G_j)
25                      then G_i ← G_j
26                      SEND(confirm,al_i, G_i) to all j ∈ G_i
27              L_i ← j where al_j ≥^{max}_{k∈G_i} {al_k}
28              state_i ← negotiating
29  when received(leave,al_j) do
30              G_i ← G_i − {j}
31              L_i ← j where al_j ≥^{max}_{k∈G_i} {al_k}
32
33  if(n = 0) then NEGOTIATE()
```

join a coalition which has a member with the higher autonomy level. This is important for the convergence of the algorithm. Lines 16-31 of the code deals with concurrent invite and accept, and to make sure that any agent is in at most one coalition at any point of time. The algorithm uses concurrent invite and accept to ensure each agent is in at most one coalition at any point of time. When the agents have formed the coalitions, they will enter the negotiating stage of the algorithm given as shown in algorithm 2.

If all the agents are in *no_conflict* state, then a consistent solution has been reached, and each agent will return their current assignment to the variables as the solution (lines 1-2). If the agents are in negotiating state, then the agent with the highest autonomy level in a coalition will gather

ALGORITHM 2: Negotiation

```
NEGOTIATE()
1   if (all agents are in no_conflict state)
2         then return (a_{i1}, a_{i2}, ..., a_{il_i})
3
4   L_i ← j where al_j ≥^{max}_{k∈G_i} {al_k}
5   if(i ≠ L_i)
6         then SEND((D_{i1}, D_{i2}, ..., D_{il_i}),f_i,f_i^*) to L_i
7   else
8         n ← #G_i
9         when received((D_{j1}, D_{j2}, ..., D_{jl_j})f_j, f_j^*) do
10              n ← n − 1
11              if(n = 0)
12                    then do branch and bound search to find an
                            assignment d_{pq} to all the variables
                            a_{pq} ∈ A_p for all p ∈ G_i
                            that minimizes Σ_{p∈G_i} ( al_p * (f_p^* − f_p)/f_p^* )
13                          if (no consistent solution found)
14                                then return "NO SOLUTION"
15                    SEND(d_{p1}, d_{p2}, ..., d_{pl_p}) to all p ∈ G_i
16
17  when received(d_{i1}, d_{i2}, ..., d_{il_i}) do
18        assign (d_{i1}, d_{i2}, ..., d_{il_i}) to a_{i1}, a_{i2}, ..., a_{il_i}
19
20  FORM-COALITION()
```

the data from all other agents in the coalition. A branch and bound search will then be performed to find a consistent solution that minimizing the sum of the relative error of each agent in the coalition. If a consistent solution cannot be reached, the algorithm returns no solution. When a solution is reached, it is broadcasted to every member of the coalition, and the agents return to coalition formation stage.

5 Application in military storage facility

In this section, we apply our proposed framework to solve a real-time control problem within a military storage facility for hazardous materials. The facility has a constantly changing environment that affects the movement of goods and unexpected events which essentiates a deviation from the original plan.

In this problem, a wide range of hazardous products are to be stored in containers. A job comprises a combination of products of given quantities retrieved from a given set of containers. One or more vehicles will be used to transport the requested products out of the facility. Demands are dynamic since their arrival timings are not known apriori. The rate of arrival of demand is also not constant contributing to the dynamism of the facility. The paths leading to the entrance are

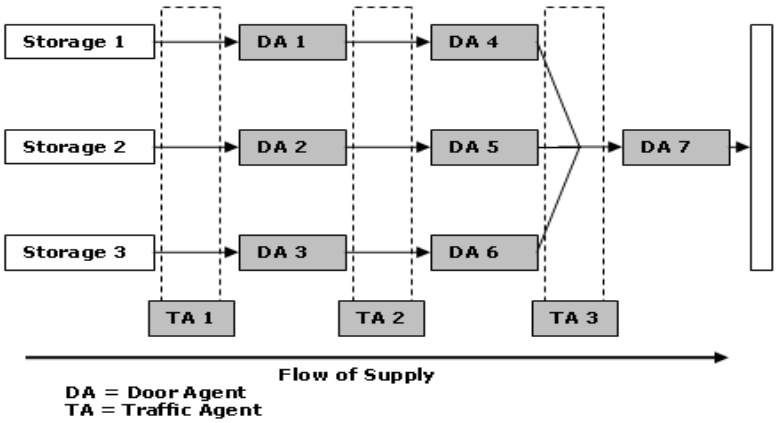

Fig. 3. Storage Facility

defined by a road network and there is a restriction on the number of vehicles that can travel on a particular road segment at any time point. Due to the nature of the goods to be transported, there are safety measures in terms of doors which limit the exposure area at any one time. These doors open to allow goods through and close to maintain safety.The doors separate road segments and should be kept closed for safety of the facility, and opened only to allow vehicle moments. The number of door opening and closing should also be minimized because they generate heat within the facility. The door temperature increases each time the door is opened or closed, and decreases gradually over time in between. There is also a safety consideration that discourages doors from opening concurrently (to contain possible leakages and explosions).

5.1 Experimental setup

We perform experiments on a prototype of the storage facility problem. In this paper, we report results for a 7-agent problem responsible for the 7 doors (not including the main door). Each agent has two objectives, namely to minimize request tardiness (by minimizing the waiting time of the vehicles), and to minimize the heat caused by opening/closing. The constraints are: the exposure of the facility due to the opening of the doors must not exceed certain level. The exposure is measured by the number of road segments that are revealed because of the opening of the doors. In this case, it must not be more than 4. There are 3 traffic agents with 1 overlapping road segment. The congestion level of this segment is limited to 15 vehicles at any point of time. The traffic agents have a single objective, that is to minimize the delay of jobs. The outline of the system is given in Figure 3.

The environmental variables captured in the experiment are the war level and the temperatures of each door. These two inputs are fed into the fuzzy system. The domain of the temperature $[25, 65]$ is normalized to $[0, 1]$ and divided into 5 fuzzy sets {L,LM,M,MH,H} as shown in Figure 4. Similarly, the domain of the input war-level $[0, 100]$ is normalized and divided into 3 sets {L,M,H}. The output variables of the fuzzy system are: autonomy level, with domain $[0, 1]$, and objective function, with domain $\{0, 1\}$. The domain of the autonomy level is also normalized and divided into 5 sets {L,LM,M,MH,H}. The rule base inside the fuzzy system are obtained from empirical and historical data.

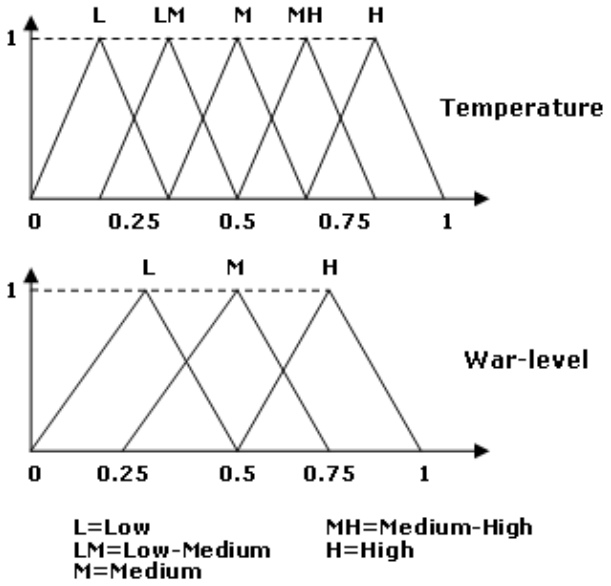

Fig. 4. Fuzzification of Input Variables

Input		Output	
Temperature	War-level	Autonomy Level	Objective Function
Low	Low	Low	High
Low	Medium	Low-Medium	Low
Low	High	Medium	Low
Low-Medium	Low	Low	High
Low-Medium	Medium	Low	Low
Low-Medium	High	Medium	Low
Medium	Low	Medium	High
Medium	Medium	Medium	Low
Medium	High	Medium	Low
Medium-High	Low	Medium-High	High
Medium-High	Medium	Medium-High	High

Fig. 5. Subset of The Inferencing Rules

A subset of the rules for determining the autonomy level and objective function of the door agents is shown in Figure 5, where a low objective function refers to minimizing tardiness and high refers to minimizing temperature.

For each pair of environmental variables, we generate 3000 requests with different deadlines. The two output variables that we observed are: the tardiness of the requests (how late the requests are satisfied) and the safety level of the system, by measuring the exposure caused by the opening of the doors.

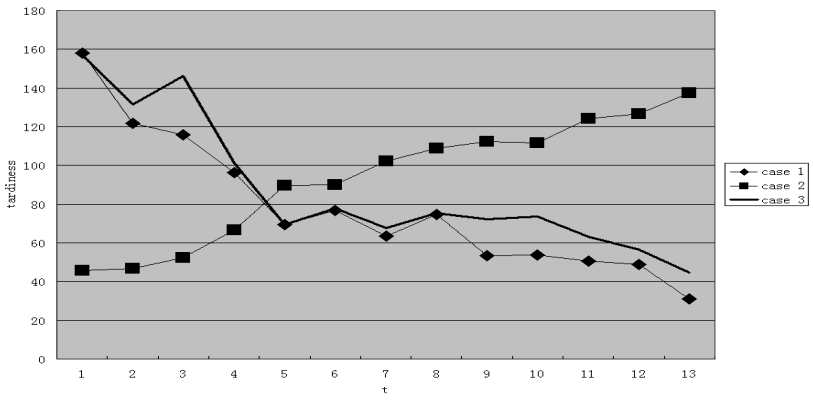

Fig. 6. Comparison of Tardiness

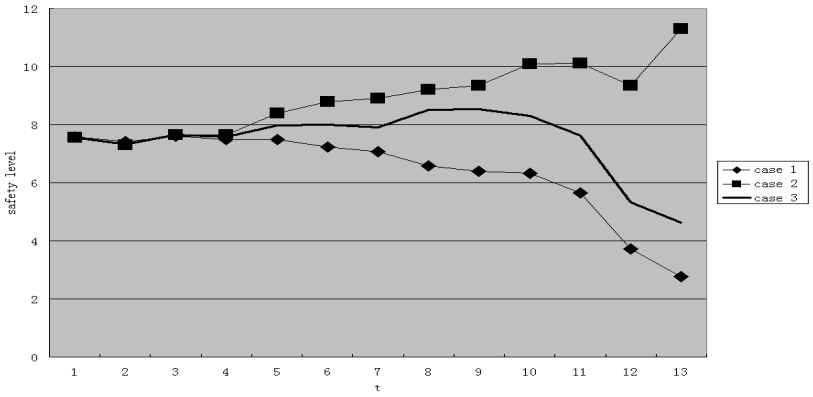

Fig. 7. Comparison of Safety

5.2 Experimental results

The test cases are divided into 3 parts. First, we fixed the average temperature to 25^{o}C, and varied the war level. Second, we fixed the war level, to 15% and varied the average temperature. Thirdly, we varied both the war level and the average temperature.

Figure 6 shows the measure of tardiness for 3 different cases. The first case shows that as the war level increases over time, tardiness decreases but at the same time the security level of the whole system also decreases, as seen in Figure 7. In the second case, as we increase the temperature over time, the effect is the opposite of the first case, as seen in both figures. For the third case, the variations in the observed output variables are shown in Figures 6 and 7. It can be observed that the system adapts to changes in environmental variables and respond accordingly.

6 Application in Crisis Management

The recent series of world disasters and other emergencies have lead the international community to react in terms of providing medical aid, food and other basic necessities. Unfortunately due

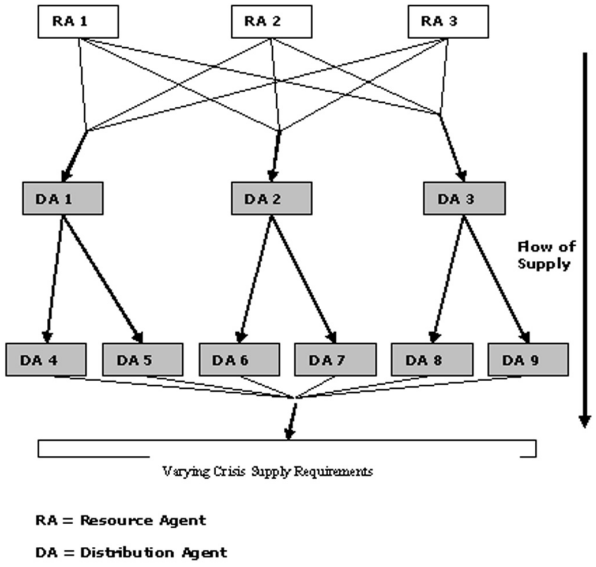

Fig. 8. Supply Chain

to the unpredictable nature of such events and the inability to react quickly there is often a very costly delay in supplies reaching their destinations While there is generally a shortage in terms of re-deployable resources a more effective management of resources available in terms of utilizing a real-time response has a tendency to reduce the delays. In this section, we apply our proposed framework to solve the real-time control problem in crisis management. In this problem, a wide range of supplies are to be collected by various collection centers from a range of supply locations. The supplies are to be sent to crisis locations based on the needs at each location. A job or supply requests comprises a combination of products of given quantities. One or more vehicles will be used to transport the requested products from the supply locations to the collection centers and similarly to the crisis locations. The same set of resources are in the control of the crisis manager. Demands are dynamic since crisis locations and extent are not known apriori. We can assume that the collection centers are fixed in terms of location and there is a known fixed number of regions that a collection center serves.

It can be assumed that there is always adequate supplies to meet the demand. However supplies are also dynamically changing(increasing) as demand occurs. Supply locations however are assumed to be regional and fixed.

6.1 Experimental setup

The overall setup can be seen at figure 8. It is a 3-echelon supply chain with 12 agents interacting with each other. There are 3 types of agents involved: (1) Resource Agents, which are the suppliers with unlimited storage capacity. The amount of supply goods at each of the resource agent increases at a constant rate. In this experiment, we assume 3 types of supply goods, each supplied by one of the resource agents. (2) Mid-level Distribution Agents, which serve as holding area and

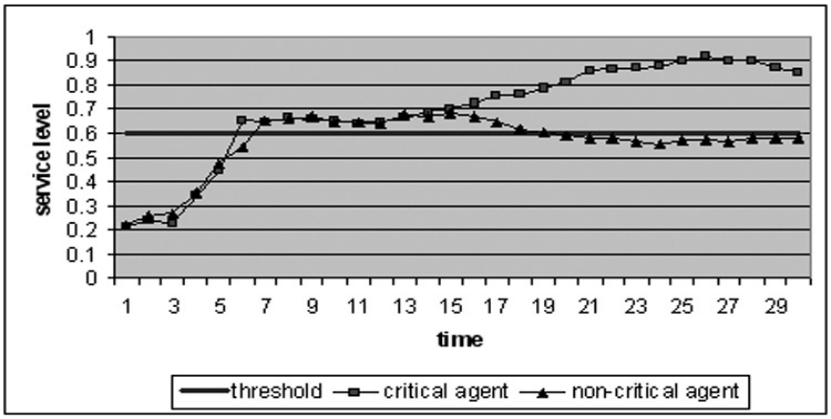

Fig. 9. Demand satisfaction (service level) over time

assembly points where the 3 types of supply goods into the final goods to be distributed to the lower-level distribution agents. (3) Lower-level Distribution Agents, where the demands (crisis) occur. Demand occurs dynamically during the running time of the system, and sensed by the lower-level distribution agents. The lower-level distribution agents will then send the changes in demands to the central controller which will determine the autonomy level of each agent. All distribution agents have limited storage capacity.

There are 3 types of cost in this setting: storage cost on each agent, transportation cost, and processing cost at mid-level distribution agent. The resource and mid-level agents have a single objective, that is to minimize cost while satisfying the service level constraint. Service level is a user-defined value that specifies the percentage of the demand satisfied during a certain period of time. This is a soft constraint, which the system is allowed to violate. The lower-level agents, however have two objectives, minimizing cost while maintaining a certain service level and maximizing service level while maintaining a certain threshold on cost. The lower-level agents can switch between objectives during the running time of the system. Requests for supply arrive from the lower-level agent to the higher-level agents and decisions are made by coalition formation process described in section 4.2. Each agent maintains its own inventory policy and the coalition formation occurs only where there are conflicts between agents.

6.2 Experimental results

The result of the experiment can be observed in figure 9 and 10. The time can be divided into 3 phases: (1) first phase (from time point 1 to 7) is the initialization phase, (2) second phase (from time point 8 to 15) is the behavior of the system under normal situation, and (3) third phase (from time point 16 onwards) is when there is a sudden drastic increase in one of the lower-level distribution agent, we call this agent the critical agent, and other lower-level distribution agents as non-critical agents. Figure 9 shows the service level of the critical agent and the average service level of the non-critical agents. Under normal situation, the threshold on the service level is maintained while the agents tries to minimize its cost. On the critical situation, however, when changes is detected, the critical agent switch its objective from cost to service level. The changes also caused this agent to receive a high level of autonomy, causing the service level to increase substantially, and the cost also increases towards the threshold given, as shown in figure 10.

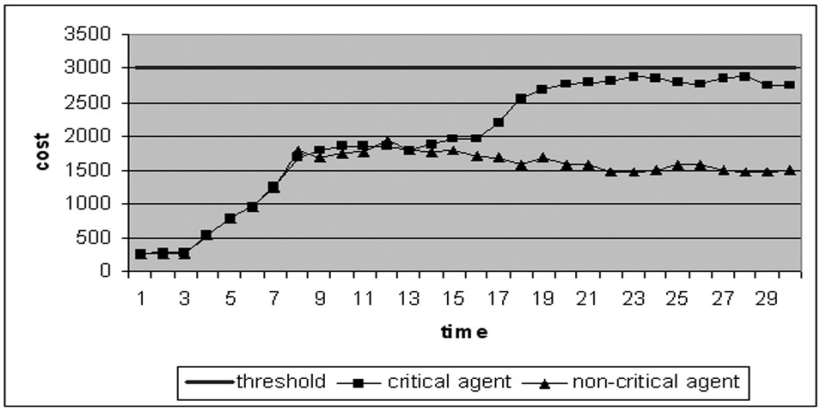

Fig. 10. Cost over time

The increase in service level of the critical agent causes the service level of the non-critical agents to drop gradually. The effect conforms to what we expected, i.e. the cost of increasing the service level of one critical agent is shared equally by the other non-critical agents. It is also observed that when the demand of the critical agent drops to normal level, the whole system behavior return to normal state.

7 Conclusion

We studied the problem of adjustable autonomy in the context of sense and respond to dynamic environmental changes. Our solution approach has shown promising results for real-world problems involving the design of a control system for a military storage facility under a peace-to-war transition, as well as the management of a supply chain under sudden crisis. We believe our proposed approach can be applied to solve other real-time supply chain management problems where drastic changes in environmental variables (such as peace-to-war) is an issue.

References

[1] Barber K.S., Goel A., and Martin C.E., Dynamic Adaptive Autonomy in Multi-Agent Systems. Journal of Experimental and Theoretical Artificial Intelligence, Special Issue on Autonomy Control Software, 12, 2 (2000), 129- 147.

[2] Bessiere C., Arc-Consistency in Dynamic Constraint Satisfaction Problems. In Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), Anaheim, CA, USA. AAAI Press, 221?26.

[3] Chen Y., Peng Y., Finin T., Labrou Y., Cost S., Chu B., Sun R. and Wilhelm B., A negotiation-based Multi-agent System for supply chain management. Third International Conference on Autonomous Agents (Agents-99), Workshop on Agent based Decision-Support for Managing the Internet-Enabled Supply-Chain, Seattle, WA (May 1999).

[4] Dorais G,, Bonasso R,, Kortenkamp D,, Pell P,, and Schreckenghost D., Adjustable autonomy for human-centered autonomous systems on mars. Presented at the Mars Society Conference (1998).

[5] Fox M.S., Barbuceanu M., and Teigen R., Agent-Oriented Supply-Chain Management. International Journal of Flexible Manufacturing Systems, Vol. 12, No. 2/3, pp. 165-188, 2000.

[6] Fox M.S., Chionglo J.F., and Barbuceanu M., The Integrated Supply Chain Management System. Internal Report, Department of Industrial Engineering, University of Toronto (1993).

[7] Gunderson J.P., Adaptive Goal Prioritization by Agents in Dynamic Environments. IEEE Systems, Man, and Cybernetics 2000, pgs 1944 - 1948, Oct 8 - 11, 2000.

[8] Kaihara T., Multi-agent Based Supply Chain Modelling with Dynamic Environment. International Journal of Production Economics, Vol. 85 (2003), pp. 263-269.

[9] Mailler R., and Lesser V., Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), pp. 438 - 445.

[10] Ran Y.P., Roos N., and Herik H.J. van den., Approaches to Find a Near-minimal Change Solution for Dynamic CSPs. In Proceedings of the Fourth International Workshop on Integration of AI or OR Techniques in Constraint Programming for Combinatorial Optimisation Problems, Le Croisic, France.

[11] Scerri P., Pynadath D.V., and Tambe M., Towards Adjustable Autonomy for the Real World. Journal of AI Research (JAIR) 2002, volume 17, pp. 171-228.

[12] Shen W. and Norrie D.H., An Agent-Based Approach for Manufacturing Enterprise Integration and Supply Chain Management, G. Jacucci, et al (eds.), Globalization of Manufacturing in the Digital Communication Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise, Kluwer Academic Publishers (1998).

[13] Shen W., Ulieru M., and Norrie D., Implementing The Internet Enabled Supply Chain Through A Collaborative Agent System, Agents'99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain, Seattle, WA.

[14] Wang H.G., Rooda J.E. and Haan J.F., Solve scheduling problems with a fuzzy approach. In Proceedings of the 5th IEEE International Conference on Fuzzy Systems 1996. Part 1 (of 3), New Orleans, USA. Pp 194-198.

[15] Modi P., Shen W., Tambe M. and Yohoo M., ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. Artificial Intelligence. To appear.