# Designing and Evaluating an Adaptive Trading Agent for Supply Chain Management Applications

**Minghua He, Alex Rogers, Esther David, and Nicholas R. Jennings**

School of Electronics and Computer Science, University of Southampton, U.K.

{mh,acr,ed,nrj}@ecs.soton.ac.uk

## Abstract

This paper describes the design and evaluation of SouthamptonSCM, a finalist in the 2004 Trading Agent Supply Chain Management Competition (TAC SCM). In particular, we focus on the way in which our agent sets its prices according to the prevailing market situation and its own inventory level (because this adaptivity and flexibility are the key components of its success). Specifically, we analyse our pricing model's performance both in the actual competition and in controlled experiments (against both risk-seeking and risk-averse price setting methods). Through this evaluation, we show that SouthamptonSCM performs well across a broad range of environments.

## 1  Introduction

Internet technologies have contributed significantly to e-commerce by increasing the mutual visibility of consumers and suppliers, and by raising the possibility that some of their trading processes may be automated. However, despite these advances, most procurement activities within supply chains are still based on static long-term contracts and relationships. Now, in many cases, such contracts are detrimental because they fail to handle the dynamic nature of these environments, where new suppliers and consumers may enter the market at anytime and where trading partners may fail to fulfill their commitments. To rectify this, we believe agent-based solutions are needed. To date, however, the use of agents within e-commerce has generally focused on simple auctions [3]. Whereas, the supply chain domain typically requires handling a much more complex setting where decisions must be made in the presence of much greater degrees of uncertainty and dynamism [5].

To this end, the International Trading Agents Competition for Supply Chain Management (http://www.sics.se/tac) (TAC SCM) represents an ideal environment in which to test the autonomous agents that we develop. Such multi-agent research competitions present well-defined problems in which alternative solutions can be tested, compared and evaluated. In the TAC SCM scenario, agents are competing as computer manufacturers in a virtual business world to handle three basic subtasks: acquiring components, managing a local manufacturing process, and selling assembled computers to customers. The agents in this scenario are required to operate with severely incomplete and imperfect information and have a high dimensional strategy space. Specifically, the agents must simultaneously compete in separate, but dependent, markets in order to buy the necessary components and compete with other agents for customers' orders. To add to this complexity, the agents' decision-making is constrained by a severe time deadline and thus any proposed solution must also be computationally efficient.

Against this background, we present our work in developing an adaptive agent that was a finalist in the 2004 TAC SCM competition (6 out of 29 participants reached the finals). The key contribution of this work is the techniques that we develop to enable the agent to adapt its price setting to the prevailing market situation, its own internal state (inventory level) and the time that has elapsed. At their core, these techniques employ fuzzy reasoning in order to allow the agent to adapt its prices daily so that it can fully exploit its production capacity, while still maximising its revenue by selling at appropriate prices. Previously, fuzzy techniques have been successfully applied to solve the problems of automated auction [2; 4] and negotiation [6]. So, in this work we also employed fuzzy techniques to tackle the problem.

The remainder of the paper is organized as follows. Section 2 briefly outlines the TAC SCM. Section 3 presents our agent. Section 4 evaluates the performance of the agent (in general) and the price model (in particular). Finally, Section 5 concludes.

## 2  The TAC SCM Game

In this game, six agents (competition entrants) compete with one another to procure raw components and fulfil customer orders for assembled PCs. Each PC is assembled from four components: CPU, motherboard, memory and hard disk (e.g. a PC with a 2GHz IMD processor with 1GB memory and a 300GB hard drive or a PC with a 5GHz Pintel processor with 2GB memory and a 500GB hard drive). The production capabilities of all the agents are equal, in that they are all capable of producing any of the 16 distinct computer types and they all have the same limited production capacity.[1]

---

[1] Different PC types require a different number of production cycles and each agent is limited to 2000 of these cycles per day.
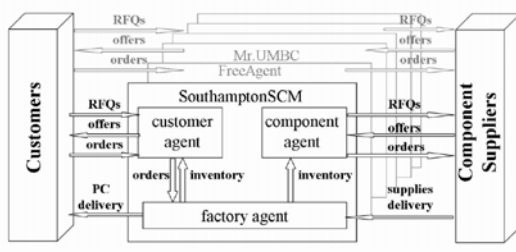
Figure 1: Overview of the SouthamptonSCM agent.

The agents operate simultaneously in separate markets to buy components from a number of suppliers and to sell assembled PCs to customers. Both of these markets operate as follows: (i) the buyer issues Request For Quotes (RFQs) to one or more sellers; (ii) the sellers respond to some or all the RFQs with offers detailing the price, quantity or delivery date; and (iii) the buyer sends orders to accept offers.

Consequently, on each of the 220 simulation days of the game, agents receive from the customers a new set of RFQs and, in response to previously sent offers, they receive orders for assembled computers. Likewise, component suppliers that were previously sent RFQs respond with offers. Thus, in each day of the game (lasting 15 seconds), the agent must decide on the following: (i) which new supplier RFQs to issue and which supplier offers to accept; (ii) which customer RFQs to respond to, and what price to offer; and (iii) how to schedule the production of PCs given the availability of components, the limited capacity of the factory and the delivery deadlines of pending orders.

An agent spends money on buying the components, paying for the storage of both components and PCs, paying penalties if it defaults on a promised delivery date and paying overdraft penalties if it is in debt to the bank. The agent earns money by selling PCs and receives interest from the bank if its balance is positive. Success of an agent is measured in terms of its profit (*i.e.*, its bank balance at the end of the game).

## 3 SouthamptonSCM

SouthamptonSCM can be decomposed into three sub-agents (see figure 1).[2] The *component agent* decides which RFQs and which orders to send to which suppliers. The *customer agent* receives RFQs from the customers and decides what offers to respond with. It also communicates with the factory agent to obtain the updated inventory levels and to send the relevant customer PC orders. The *factory agent* receives the supplies delivered from the suppliers, decides based on the available resources (computer components and factory cycles) in what order the customer orders should be produced, and determines the schedules for delivering the finished PCs to the customers. We now deal, in turn, with each of these sub-agents.

---

[2]Here we use the notion of sub-agents (instead of modules) because each of them can autonomously communicate with the suppliers and customers to get the RFQs, can send offers and obtain orders, and can decide how to respond to this information.

### 3.1 The Component Agent

The price offered by a supplier in response to an RFQ is based entirely on its available production capacity and the quantity agents ask for (*i.e.*, price increases as capacity decreases or quantity required increases). On Day 0, all the suppliers have their full capacity available, thus the prices they offer are at their lowest value. Therefore, intuitively, it makes sense to order a large number of components on Day 0 (indeed this was a widely used tactic in the 2003 competition [8]). However, due to a rule change, the components now attract a storage cost. Thus the more the agent stores and the longer it stores it, the higher the storage cost. This means the key challenge of the component agent is to attain an appropriate balance between availability and timeliness. This is hard because if the agent buys more units early (at lower prices) it has to pay for storage and some components may be unused at the end of the game. However if the agent just buys what it needs when it is needed, it may end up without the necessary components at the necessary time (since there is often a delay between the actual delivery date and the one the suppliers promise). Given this, our agent makes a trade-off between placing a big order on Day 0 and buying gradually during the rest of the game.

In more detail, experience from practise games showed that despite the storage cost, having a reasonably big order on Day 0 is still profitable because of the low prices that can be obtained. Specifically, we found it most effective when this number just covers the quantity the agent needs in low demand games (in order to avoid waste). Thus on Day 0, SouthamptonSCM orders a large number of components (2000, 2000, 2500, 3500, 5000) from each supplier with corresponding delivery dates of Day 10, 25, 40, 70 and 110. These dates were chosen in order to try and give the agent a steady stream of components for the early to middle part of the game. The agent accepts the corresponding offer if the delivery date is not too far from the date it asks for. However, if the demand turns out to be greater than what the agent ordered, it can still buy components (at higher prices) during the rest of the game. In particular, after the Day 0 order, the agent keeps asking for small quantities of components from the suppliers and placing orders for them if the offer price is low. At about Day 140, the agent starts to order components for the rest of the game. It does this based on the average daily demand for computers (as a predictor of how many components are needed) and buys gradually if the offer prices from the suppliers are low.

### 3.2 The Customer Agent

The customer agent is the key component in SouthamptonSCM's strategy (because we believe that offering the appropriate price at the right time is vital for success). If the price is too low, the agent will receive a low profit and if it is too high it will fail to win any orders (because customers always choose the lowest offer price among those they receive). Here, the key challenges are to determine which customer RFQs to bid for and at what price. To achieve this, we use inventory driven methods to choose RFQs and soft computing techniques to calculate the price (see below).

## Choosing RFQs and setting prices.

The customer agent uses an inventory driven strategy when selecting customer RFQs. That is, it only offers customers PCs according to what is presently available in its inventory. By doing this, the agent avoids getting penalties for committing to more than it can produce (the quantity of PCs it can produce is constrained by the availability of components and factory cycles).

In more detail, table 1 shows the strategy we use. Given a customer RFQ $(i, q, p_{res}, c_{penalty}, d_{due})$, where $i \in \{1, \cdots, 16\}$ is the type of PC the customer wants, $q > 0$ the quantity, $p_{res} > 0$ the reservation price (maximum it will pay), $c_{penalty} > 0$ the fine if the computers are not delivered on time, and $d_{due}$ the desired delivery date. On each day, the customer agent receives a bundle of such RFQs and sorts them in the order of decreasing $(p_{res} - c_{penalty}/q)$. The intuition here is that the agent will first serve customers with high reserve prices and low penalties. This is because the higher the $p_{res}$, the more profit will be made (compared to selling the same product to a customer with a low $p_{res}$). At the same time, the agent also wants to avoid getting high penalty orders because of the inherent uncertainties that exist in the game.

The next consideration relates to the agent's production capacity. Specifically, as there is only limited production capacity per day, the agent needs to calculate the number of cycles that can be offered to respond to the customer RFQs of that day.[3] Thus, it updates the available production cycles for each day based on the customer orders that have just been received. Specifically, for each RFQ, the agent first checks whether it can be supplied from its stock of finished PCs (see Section 3.3). If it can, the corresponding PC inventory is decreased. Otherwise, the agent checks whether it holds enough components in its inventory and whether it has a sufficiently high remaining production capacity $C[d_{due} - 2]$ on day $(d_{due} - 2)$, which is the latest the PCs can be produced.[4] If it does, the agent decreases its component inventory and *reservedCycles* for day $(d_{due} - 2)$ accordingly and increases the number of cycles offered ($q \times o^i$, where $o^i$ is the cycles needed for PC type $i$) on that day.

Now the agent needs to consider what price can be offered to the RFQ. Based on the demand in the market, the inventory level, and how far we are into the game, the agent first computes a *reference price* ($p_{ref}^i$) that corresponds to a reasonable current market price. Thus for PC type $i$:

$$p_{ref}^i = p_{low}^i + (p_{high}^i - p_{low}^i)r \qquad (1)$$

where $p_{low}^i$, $p_{high}^i$ are the lowest and highest transaction prices of PC type $i$ on the previous day, and $r \in [0.4, 1.2]$ is an *adjustment factor* that determines how far away the reference price is from the lowest price. This adjustment factor is set

[3]Note here the agent does not offer the exact number of cycles that are available ($C[d_{due} - 2]$) on day $(d_{due} - 2)$, but rather it includes a risk factor ($\lambda \times C[d_{due} - 2]$) which enables it to offer more than it actually has in order to maximise the production utilisation. Here $\lambda > 1$.

[4]Note that for an RFQ with the due date $d$, the agent checks whether it can be produced on the latest possible day $(d - 2)$ because this has previously been shown to be effective in this scenario [1].

---

Table 1: Pricing strategy on day $d$.

- list RFQs in decreasing order of $(p_{res} - c_{penalty}/q)$
- update the production capacity $C[k]$ of each day $k$
- $offeredCycles = 0$ and $reservedCycles[k] = 0$
- calculate the reference price for each kind of PC $p_{ref}^i$
- for each RFQ in the list
  - $p_{offer} = max\{p_{ref}^i \times (1 + f(d_{due})), p_{base}^i\}$
  - if PC inventory $\geq q$ then
    - offer $q$ PCs at $p_{offer}$
    - decrease PC inventory by $q$
  - else if component inventory $\geq q$ and
    $reservedCycles[d_{due} - 2] + q \times o^i \leq C[d_{due} - 2] \times \lambda$ then
    - offer $q$ PCs at $p_{offer}$
    - increase $offeredCycles$ by $q \times o^i$
    - decrease $reservedCycles[d_{due} - 2]$ by $q \times o^i$
    - decrease component inventory accordingly
- else do not offer PCs to this customer

---

through the fuzzy reasoning mechanism and is adapted according to the quantity of orders received and the number of orders expected (see Section 3.2 for more details). However, given an RFQ, the offer price is not the reference price of PC type $i$. Rather, $p_{offer}$ is the maximum of the cost for PC type $i$ ($p_{base}^i$ is the money spent buying the constituent components) and the reference price modified by a factor related to the requested delivery date. This ensures the agent sells the PC at least for its cost. The use of $d_{due}$ means that the sooner the due date, the higher the offered price is compared to the reference price (because the agent has little time to produce the computers with a bigger risk of being penalised for being late).

In more detail, the fuzzy reasoning inference mechanism employed to set the adjustment factor in Equation (1) is based on the standard Sugeno controller [7] and the following is a representative rule for determining it:[5]

$R_j$: if $D$ is *high* and $I$ is *high* and $E$ is *far* then $r_j$ is *big*

where the customer demand ($D$) is expressed in the fuzzy linguistic terms *high*, *medium*, and *low*, the inventory level ($I$) in the terms *very-high*, *high*, *medium*, and *low*, and days to the end of the game ($E$) in the terms: *far*, *medium*, and *close*. $r_j$ is the output of the individual rule $j$ (*i.e.*, the adjustment factor discussed above). Thus, the above rule captures the fact that if the type of PC is in high demand in the market, the agent has a high inventory for this kind of PC and there is a long time until the end of the game, then the adjustment factor should be big (thus resulting in a higher bid price). The firing level $\alpha_j \in [0, 1]$ of rule $R_j$ is computed in the standard way by using the *Min* operator on the membership values of the corresponding fuzzy sets. According to the Sugeno controller definition, the crisp control action (*i.e.*, the output of the fuzzy rule base fed into Equation (1)) is:

$$r = \frac{\sum_{j=1}^{n} \alpha_j r_j}{\sum_{j=1}^{n} \alpha_j} \qquad (2)$$

[5]Our agent incorporates some 20 rules which vary the price according to the market demand, its inventory level and time into the game.

Table 2: Adaptation of the offer prices.

- update *receivedTotalCycles*;
- calculate *receivedCycles*;
- $expectedCycles = min\{2000, offeredCycles \times \mu\}$;
- if $receivedCycles < \overline{expectedCycles}$ then $r = r - \delta$;
- else if $receivedCycles > \overline{expectedCycles}$ then $r = r + \delta$.

Table 3: Production scheduling for day $d$.

- list the orders with due date $d + 2$ in list 1;
- list late orders (but still valid $d - 3 \leq d_{due} \leq d + 1$) in the decreasing order of the due date into list 2;
- list the future orders (due date $\geq d + 3$) in the increasing order of the due date into list 3;
- append list 2 to list 1 and list 3 to list 2;
- for each order in the combined list
  - if computers in the inventory can fill the order then deliver the computers;
  - else if components are available and factory capacity is not full then produce more PCs to fill the order;
- if there is extra factory capacity left and enough components, then check whether additional PCs should be produced.

**Adaptation of offer prices.**

Given the uncertainty in TAC SCM, we believe it is essential for the agents to be responsive to the prevailing situation during the course of bidding for customer orders. The idea is that the agent can only use 2000 production cycles every day, so, to maximise throughput, the number of cycles necessary to produce the received customer orders should also be 2000. Thus if the received orders require more than this figure, it means that the agent has set its offer price too low. In contrast, if the number is too small, it means the agent is not winning enough customer orders (which implies that its offer price is too high). However, we cannot just base our decision on 2000 cycles because some of that day's production cycles might be reserved by the orders of previous days (because more than 2000 cycles were needed previously). In this case, the number of expected cycles for the day's order is only part of the offered cycles of the previous day (because all agents compete for customer orders and only the lowest price can be accepted). With this information, the agent can adapt its offer prices in order to try and keep the factory working at high capacity, but still be responsive to the prices other agents offer (based on the highest and lowest transaction prices of the previous day). Specifically, the adaptation rule is if the orders the agent receives need more cycles than it expected, it will increase its price, otherwise it will decrease it.

Table 2 shows how the adaptation of the offer prices works. Here, *receivedTotalCycles* represents the total number of cycles needed to produce the PCs for the orders just received; *receivedCycles* represents the cycles needed for the orders that the agent offers from the component inventory rather than the finished PCs (finished PCs do not count since they do not require more cycles to produce them); *offeredCycles* is the actual total number of cycles offered on the previous day (as per table 1) and *expectedCycles* is *offeredCycles* multiplied by the expected acceptance rate ($\mu = 0.75$), *i.e.*, how many cycles are expected to win customer orders among all the cycles offered. Now if *receivedCycles* is much less than the expected number of cycles, the agent will decrease the adjustment factor (thus the price is decreased, see Equation (1)) by $\delta$ (here $\delta = 0.02$), otherwise it will increase the adjustment factor (thus the price is increased). However sometimes if the expected number of cycles is only slightly smaller than the actual number of received cycles, we do not decrease the offer prices (since this is a close enough approximation in a noisy environment). To realise this, we view *expectedCycles* as a fuzzy number [9].

### 3.3 The Factory Agent

One of the main challenges for the factory agent is scheduling what to produce and when to produce it (*i.e.*, how to allocate supply resources and factory time). The strategy we use includes: manufacturing PCs according to customer orders and satisfying orders with an earlier delivery date (see table 3 for more detail). Now, since the computers stored in the factory will be charged storage cost, each order will be delivered as soon as it is filled. The agent builds the PCs according to the customers' orders it has obtained (which has the advantage of ensuring that the factory always produces the needed computers on time). However, if there are still factory assembling cycles left and the numbers of finished PCs are below a certain threshold then the agent produces additional PCs of each kind uniformly (if there are enough components) to maximise the factory utilisation. In particular, this strategy benefits the agent when there is a low demand in the market (because there are actually spare cycles) and it works well in the final stages of the game. For example, on Day 217, the agent can bid on customer orders that come in on that day, meaning it gets the orders on Day 218 and delivers the computers on the last day of the game. If it just used the *build-to-order* strategy, the agent would not be able to bid for the customer orders on Day 217 because after it wins the order, there would be no time for it to buy the necessary components and produce the PCs.

## 4 Evaluation

Our evaluation is composed of three components: (i) the results from the 2004 competition; (ii) our post-hoc analysis of some of the games in the actual competition; and (iii) a systematic range of controlled experiments.

### 4.1 TAC SCM Results

TAC SCM consists of a preliminary round (mainly used for practice and fine tuning), a seeding round, quarter-finals, semi-finals, and final. The seeding round determined groupings for the quarter-finals. The top 24 agents were organised into 4 "heats" for the quarter-finals based on the positions in the seeding round and the first 3 teams for the quarter-finals of heat 1 and 3 entered into semi-final 1 and, similarly, the first 3 teams from heat 2 and 4 were entered into semi-final 2. Finally, the first 3 teams in both semi-finals entered into the final round. In the seeding round, SouthamptonSCM obtained the third highest score among all the participants and entered heat 1 for the quarter-final. In the quarter-final, we had the second highest score and we were first in our semi-

final. In the final, our agent finished in 6th position. In the final, our agent was adversely affected by the fact that several agents sent RFQs on Day 0 for huge quantities of components. Then, if the corresponding offers were expensive they declined to buy them or if they were cheap they took up the offers. However, in the meantime, since the suppliers have limited capacity they scheduled other Day 0 orders for much later in the game. Thus when this happened our Day 0 bidding was severely effected (sometimes up to Day 70) and we received severely delayed delivery dates for our orders. In such cases, we were simply unable to obtain the components we needed through our Day-0 procurement policy and so we made very few sales.

## 4.2 Competition Game Analysis

To complement and better understand the competition result and to evaluate the effectiveness of our pricing model we conducted a post hoc analysis. However it is hard to see how the pricing works from only the game results since the competition entrants contain a variety of interrelated strategies (for the different facets of their operation). Thus we decided to compare for the RFQs that the agents responded to during the game, how the price varies among different agents.[6] To do this, we analysed competition games and we were especially interested in those cases where there were strong agents. Here we take a randomly chosen representative game in the semi-final (game 1136) and analyse it in more detail.[7] In this game, we compare our agent with FreeAgent and Mr.UMBC which were the first and second placed agents in the final. Thus, in each such competition, we extracted from the game data, details of the RFQs that were received by the competing agents, the offers that they sent to the customers in response and the orders that resulted.[8] This data enabled us to compare the orders that the agents were winning with the prices that they offered. Specifically, figure 2 shows for each simulation day, the daily price (per production cycle, see figure 2 (a)) offered by each agent and the average daily number of orders that each agent won (again measured in cycles, see figure 2 (b)). These values are averaged over all PC types. Since the ultimate profitability of the agents depends on both these factors, we also calculate the average daily revenue (i.e. the number of PC orders multiplied by their prices, see figure 2 (c)).

Throughout the game, SouthamptonSCM adaptively adjusts the price offered to the customer to ensure that the factory maintains as close to full production as possible (the factory utilisation for our agent, FreeAgent and Mr.UMBC are 76%, 58%, and 61%). Generally, having a high factory util-

---



Figure 2: Comparison of daily offer prices, order quantity and revenue in game 1136.

---

isation means the agent can produce more PCs and thus win more customer orders. For example, in this game, the number of orders for these three agents are 5405, 4011, and 4300. In this example, all three agents have sufficient components to allow them to compete for the same orders. However, our pricing model is particularly successful. The prices offered by SouthamptonSCM are just low enough that the offers of the competing agents are undercut, but high enough that the resulting orders generate as much revenue as possible.

After analysing more semi-final games, we found that the prices SouthamptonSCM offers follow the same broad trend compared with the other two. And, in particular, the trend is when the customer demand is high, the prices are high, and vice versa. This can be seen from figure 2 (a), where the demand for the first half of the game is high, and the demand decreases gradually till Day 160 and increases again. Ac-

---

[6]We aim to compare the pricing model and the revenue made by responding to the customer RFQs. Thus the price paid for the components and any late penalties need not be considered here.

[7]We did not choose a game from the final because of the skewing introduced by the Day 0 bidding strategies used by some of the agents. Also it's impossible to compare the pricing of multiple games in one figure, thus we only show one representative game in the figure. However, the following discussion also applies to the other games we analysed.

[8]For clarity, we omit from this plot the other three agents, and just show data for SouthamptonSCM, FreeAgent and Mr.UMBC. The plots of the other agents show they were less effective.
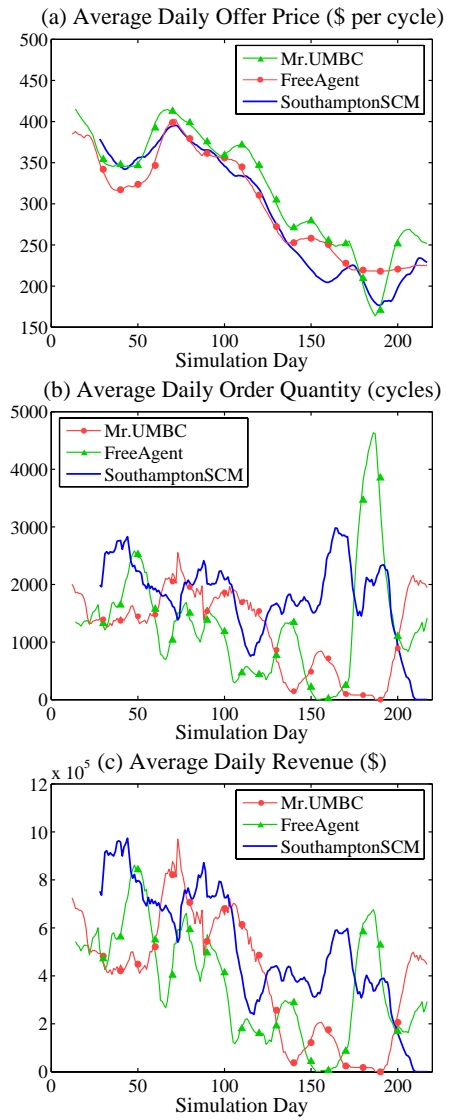
cordingly, the prices are high before Day 110 and then start to decrease gradually. At the end of the game, although the demand is increasing, the agents do not increase their prices because they want to offload their stock. Moreover, in most of the games we considered, the prices SouthamptonSCM offered just undercut the other two. This is also reflected by the quantity of orders our agent won which was again usually the highest.

## 4.3 Controlled Experiments

To evaluate the performance of our agent in a more systematic fashion than is possible in the competition, we decided to run a series of controlled experiments. As mentioned before, we attribute the success of our agent to the adaptive control of the offering price and this is what we are most interested in here. Thus, we decided to analyse how the pricing works compared with other methods. To do this, we devised two competitor agents that adopt identical strategies to SouthamptonSCM except for the method they use to offer prices. The alternative methods we consider are consistent with the broad classes of behaviour that were adopted by several of the agents in the competition:

- **Risk-seeking agent (RS-agent)**. This agent bids aggressively at high offer prices to obtain a higher profit margin in selling the PCs. It will take the risk of stocking a large number of PCs and components in the factory and paying storage cost for them. But when its PCs are sold they fetch high prices and mean it can very quickly build up profits. In more detail, the prices that RS-agent offer are the *maximum* of the cost of the computer plus a fixed profit margin (here it is 300) and the computers' reserve price minus 1. Thus, at the end of the game it sells all its computers at very low prices since it is better to sell than retain stock.

- **Risk-averse agent (RA-agent)**. This agent bids cautiously and only seeks to attain a reasonable profit margin. This means that the agent wants to sell its PCs quickly and it does not want to take the risk of stocking components or PCs (especially in games with low customer demand). Specifically, it offers the computers at the *minimum* of the cost of the computer plus a small margin and the reserve price minus 1. Here the margin is set to 300 in the first 180 days of the game and this is then decreased to 0 linearly till the end of the game. This policy is adopted because the agent hopes that it can sell all the computers by the end of the game.

Besides these two kinds of agents, the other competing participants are the dummy agents provided by the organisers. These use a build-to-order method and offer prices which are chosen uniformly from $80 - 100\%$ of the reserve prices. Generally, the dummy agent can be viewed as being risk averse because it often offers a low price (but it differs from our RA-agent in that it uses the build-to-order method). Given this background, three groups of experiments were conducted to examine the performance of each kind of agent in various situations. In experiment A, there is one SouthamptonSCM, one RS-agent, one RA-agent and three dummy agents. In experiment B, we increase the number of RS-agents to 2 and



(a) Revenue of Agents in Experiment A

(b) Revenue of Agents in Experiment B

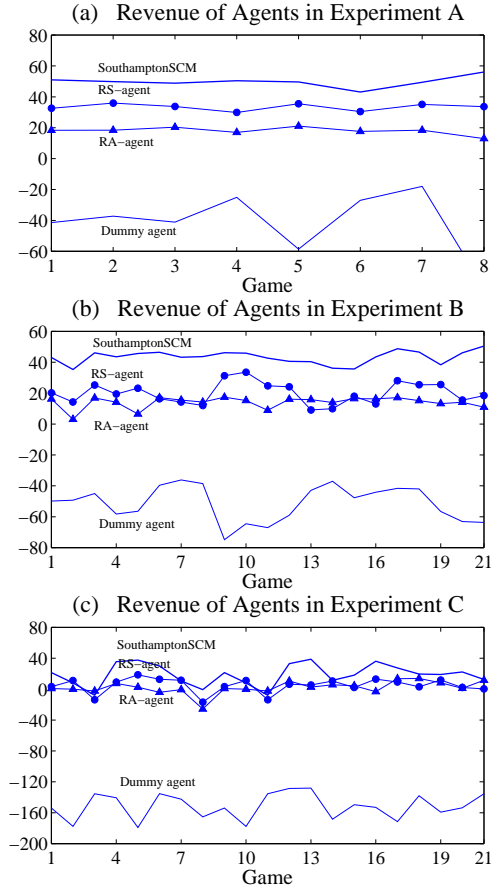(c) Revenue of Agents in Experiment C

Figure 3: Revenue of each kind of agent.

decrease the number of dummy agents to 2. In experiment C the number of RS-agents is 3 and the number of dummies is 1. The average revenue of each kind of agent in each of the experiments are then plotted.

We now start to analyse the performance of the different agents as shown in figure 3.[9] In experiment A, it can be seen that SouthamtptonSCM performs significantly better than the other two agents and that the RS-agent is better than the RAs. In experiment B, SouthamptonSCM is significantly better than both RS-agents and RA-agents and the RS-agents are better than the RAs. In experiment C, SouthamptonSCM is significantly better than the other two, however we cannot differentiate statistically which agent is better between RS and RA agents. Now, in all cases, we can attribute this success of SouthamptonSCM solely to the adaptivity aspect of its pricing (because this is the only difference between the agents). Moreover, we found that the average revenue Southampton-SCM obtained is 49.7% higher than RS-agents in experiment A, 129.7% higher in experiment B, and 58% higher in experiment C. This means, relatively speaking, Southampton-SCM does best in experiment B. It is interesting that there are more RS-agents in experiment B than in A (*i.e.*, our agent

---

[9]Statistical significance is computed by a Students t-test and this shows all results are significant ($p < 0.05$).

performs better in a more uncertain environment). This further shows that the adaptivity of prices are effective in this case. However, in experiment C, more agents use the Day-0 bidding strategy and this affects all the agents greatly (see the discussion below). To understand better about how the
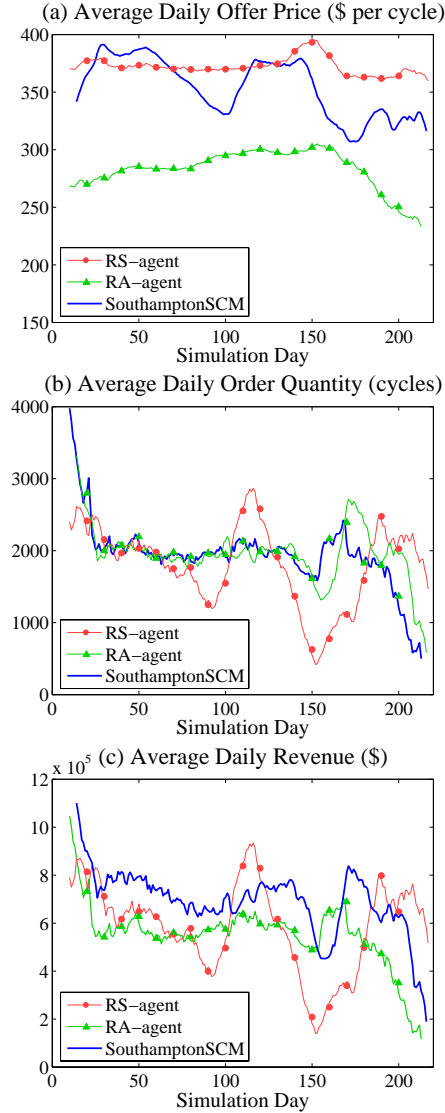


Figure 4: Comparison of daily offer prices, order quantity and revenue in the controlled experiment.

pricing of SouthamptonSCM works, we further observed for each simulation day, the daily price (figure 4 (a)) offered by each agent and the average daily number of orders that each agent won (figure 4 (b)). These values are averaged over all PC types. We then plot the average daily revenue (figure 4 (c)). Here, again, we take a randomly chosen representative game to show how the pricing of these three kinds of agents operates. As expected, the prices that SouthamptonSCM offers are roughly between the other two (below that of RS-agents and above that of RA-agents). For an RA-agent, the offer prices are very low, thus, although it can sell a large

quantity of PCs, it cannot make much profit. Specifically, we found that the RA-agent can almost always win orders (the ratio of the number of orders offered to the quantity of orders won is almost 1 : 1 and the factory utilisation is almost 100%). For the RS-agent, however, the prices are always high, meaning they build up a large stock of PCs and components in the factory. Thus only a small number of their orders make much profit although selling prices are high. Through adaptation, SouthamptonSCM can make its offer prices high enough (sometimes the average prices are even higher than RS-agents, see figure 4 (a)), but, at the same time, guarantee a large number of orders (see figure 4 (b)). This is demonstrated by the fact that its factory utilisation is almost 100%. Consequently, its revenue is higher than the other two (see figure 4 (c)).

Besides these observations about the performance of each agent, the following general observations can be made from these experiments. First, in all cases, the three kinds of agents perform much better than the Dummy agents. This means that our Day-0 procurement strategy can be viewed as being more effective than build-to-order procurement. This happens because when the Dummy agent starts to order the components after it wins the customer order, there will always be a delay between the delivery date the agent asks for and the real one. Thus the Dummy agents are often penalised for being late or missing the delivery deadline. Moreover, as shown in figure 3, the more risky agents there are, the worse the Dummy agent behaves.

Second, as more agents use the same broad strategy of Day-0 procurement, it is more likely that there will be a bigger delay between the original delivery date and the actual one (because each agent sends RFQs with a big quantity of components and the production capability of the supplier is limited, see Section 3.1). Thus, this phenomena greatly increases the uncertainty in the game and the performance of all the agents are negatively affected, (*i.e.*, the performance of all the agents is getting worse from experiment A to B and B to C). This can be seen clearly in figure 3 and explains why SouthamtptonSCM sometimes got the second or third position in a game. Through the analysis of the game data, we found that in those games, there is a significant delay in the component delivery and the factory stops working for about 20 days. This is also what happened in the final of the competition (as detailed in Section 4.1).

Third, as more agents use the risk-seeking strategy, the performance of the RS-agents is more negatively affected. This happens because the RS-agents are mutually destructive. In this situation (*e.g.*, in experiment C), although RS-agents sell PCs at high prices, the quantity of PCs sold is not sufficient to make up the cost they have spent on the raw materials of the PCs they produce. In contrast, RA-agents sell many PCs at reasonably low prices and their revenue remains high. Thus, as we can see in figure 3 (c), it is sometimes the case that the RA-agent is doing the best.

Fourth, the agent that can best adapt its offer price to the changing environment will thrive best in the game. This is because the random nature of the customer demand and the strategies of other participants make the environment highly unpredictable in terms of what is the appropriate price to set

for the PCs. As can be seen from the above experiments, neither the agent that seeks a high price, nor the one that only pursues a fixed margin are effective in all cases. Thus adaptivity is a critical requirement for effective performance in dynamic games.

## 5 Conclusions

This paper provides a number of insights into building agents for supply chain applications. Specifically, it details the design, implementation and evaluation of SouthamptonSCM; an agent that successfully participated in the 2004 trading agent competition. The agent employs fuzzy techniques at its core. In particular, it uses fuzzy reasoning to determine how to set prices according to its inventory level, the market demand and the time into the game. Moreover, the parameters involved in the fuzzy rules can be adapted according to the quantity of the received customer orders and the expected number of orders so as to maximise the factory utilisation. To evaluate the efficiency of our pricing model, we analysed actual competition games and conducted controlled experiments where we compete our agents with various numbers of risk-seeking and risk-averse agents. The actual game analysis shows that our agent is able to obtain a high revenue by offering high prices that are, nevertheless, low enough to win customer orders. In the controlled experiments, we show that in all environments we considered, SouthamptonSCM is significantly better than the other two kinds of agents (with highest average performance and lowest variance). When taken together, these evaluations show that out pricing model is both efficient and robust.

We also believe several aspects of our agent design and strategy are applicable outside the confines of this competition. First of all, the general idea of the component agent is to periodically request large orders to cover the baseline quantities needed in low demand (steady state) markets and, at the same time, buy smaller amounts of supplies when the selling price is low during the rest of the production. This mixture of baseline and opportunistic purchasing behaviour is a common strategy in this domain and the technology we develop for achieving this can be readily transferred. Second, we believe our pricing model technology will also be useful in real SCM applications where just undercutting competitors' prices can significantly improve profitability. Specifically, to apply our model in other domains, the designers of the rule base would need to adapt the fuzzy rules to reflect the factors that are relevant to their domain. Now we believe that customer demand and inventory level are highly likely to be critical factors for almost all cases and thus these rules can remain unaltered. However, the time into the game is not so broadly applicable since there is not always a rigidly fixed deadline to real life supply chains (thus some changes may be needed here). Third, the strategy employed by the factory agent for managing resources in uncertain and dynamically changing environments is generally applicable. In this case, it incorporates little in the way of domain specific knowledge and so it can remain broadly as is.

## References

[1] E. Dahlgren and P.R. Wurman. PackaTAC: A conservative trading agent. *SIGecom Exchanges*, 4(3):33–40, 2004.

[2] M. He and N. R. Jennings. Designing a successful trading agent: A fuzzy set approach. *IEEE Transactions on Fuzzy Systems*, 12(3):389–410, 2004.

[3] M. He, N. R. Jennings, and H. F. Leung. On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):985–1003, 2003.

[4] M. He, H. F. Leung, and N. R. Jennings. A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1345–1363, 2003.

[5] K. Kumar. Technology for supporting supply-chain management. *Comms of the ACM*, 44(6):58–61, 2001.

[6] X. Luo, N. R. Jennings, N. Shadbolt, H.F. Leung, and J.H.M. Lee. A fuzzy constraint based model for bilateral, multi-issue negotiation in semi-competitive environments. *Artificial Intelligence*, 148(1-2):53–102, 2003.

[7] M. Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36:59–83, 1985.

[8] M.P. Wellman, J. Estelle, S. Singh, et al. Strategic interactions in a supply chain game. *Computational Intelligence*, 21(1):1–26, 2005.

[9] H.-J. Zimmermann. *Fuzzy Set Theory and Its Applications*, chapter 11, pages 203–240. Kluwer Academic Publishers, 1996.