Master Thesis
Computer Science
Thesis no: MCS-2004:14
June 2004

Evaluating Agent Strategies for the TAC Supply Chain Management Competition

Viktor Allblom

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author: Viktor Allblom
Address: Älgbacken 4, 37234 Ronneby
E-mail: pt00val@student.bth.se

University advisor:
Paul Davidsson
School of Engineering

School of Engineeering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : +46 457 271 25

ABSTRACT

The TAC Supply Chain Management game was designed to capture many of the challenges involved in dynamic supply chain practices. To evaluate the game I created four different agents, which operate according to simple but very different strategies. In addition, an advanced agent was created to see if the game was advanced enough not to be dominated by simple strategies. While the game is advanced enough to resist simple strategies, it is so simplified that it will never help solve any real world problems unless it is expanded to include more factors/problems of supply chain management.

Keywords: TAC Supply Chain Management, agents, games, simple strategies

CONTENTS

ABSTRACT ..... **I**

CONTENTS ..... **II**

1 INTRODUCTION **1**

1.1 BACKGROUND 1
1.2 PROBLEM DEFINITION 2
1.2.1 Research questions *2*
1.2.2 Methodological approach *2*

2 GAME OVERVIEW **3**

2.1 GAME MECHANICS 4

3 SIMPLE AGENTS **6**

3.1 AGENT RANDOM 6
3.1.1 Setup *6*
3.1.2 Results *7*
3.2 AGENT SABOTEUR 8
3.2.1 Setup *8*
3.2.2 Results *8*
3.3 AGENT CALM 9
3.3.1 Setup *9*
3.3.2 Results *10*
3.4 AGENT AGGRESSIVE 11
3.4.1 Setup *11*
3.4.2 Results *12*

4 AN ADVANCED AGENT STRATEGY **13**

4.1 ADVANCED AGENT SETUP 13
4.2 COUNTERING AGENT RANDOM 15
4.3 COUNTERING AGENT CALM 15
4.4 COUNTERING AGENT SABOTEUR 16
4.5 COUNTERING AGENT AGGRESSIVE 16
4.6 COUNTERING A MIX OF AGENTS 17
4.7 REDAGENT 18

5 DISCUSSION **20**

6 CONCLUSION **21**

6.1 FUTURE WORK 21

REFERENCES **23**

1 INTRODUCTION

1.1 Background

Michael Wooldridge has a good explanation about what an agent is in his book [2] "Agents are computer systems with two important capabilities. First, they are at least to some extent capable of *autonomous action* – of deciding *for themselves* what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents – not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like."

What this means in plain English is that an agent is a program that has one or more goals it strives to achieve, multiple ways of doing that, and that it is capable of somehow interacting with other agents when doing so. Two examples of agents are; Quake bots (goal is to kill as many other bots and players as possible) and search bots (goal is to find uncharted web pages) used by most search engines on the internet (Google, Yahoo, Altavista and others). The relation between agents and Artificial Intelligence is often misunderstood; AI is a subfield of intelligent agents. AI is focused on the ability to learn, plan, understand and so forth. An agent is not required to have any of those abilities; it does not have to be "intelligent".

Supply Chain Management (SCM) is concerned with planning and coordinating the activities of organizations across the supply chain, from raw material procurement to finished goods delivery. In today's global economy, effective supply chain management is vital to the competitiveness of manufacturing enterprises as it directly impacts their ability to meet changing market demands in a timely and cost effective manner.

While most of the supply chains of today are essentially static, relying on long term relationships among key trading partners, a more flexible approach offer the prospect of better matches between suppliers and customers as market conditions change. However, adoption of such practices has however proven to be elusive, due to the complexity of many existing supply chain relationships and the difficulty in effectively supporting more flexible trading practices.

The Trading Agent Competition (TAC) SCM game [8] was designed to capture many of the challenges involved in supporting dynamic supply chain practices, while keeping the rules of the game simple enough to entice a large number of competitors to submit entries. The game has been designed jointly by a team of researchers from the e-Supply Chain Management Lab at Carnegie Mellon University and the Swedish Institute of Computer Science (SICS).

1.2 Problem definition

My original intention was to include an additional research question: “Is it possible to model the opponents in terms of simple strategies well enough to use this to decide a winning strategy?” However, this question was too easy to answer. The TAC SCM game does not supply any information about the actions each opponent take. The only information the agent is able to gather is the effect of all of the opponent’s actions have on the game environment, but that’s not enough to model them into specific strategies.

1.2.1 Research questions

In this thesis I will try to answer the following questions.

- Is the TAC SCM game advanced enough to model the real world in a relevant way? If not, how can the TAC SCM be expanded to become more interesting and closer relate to the real world?
- Is the TAC SCM advanced enough not to be dominated by simple strategies?

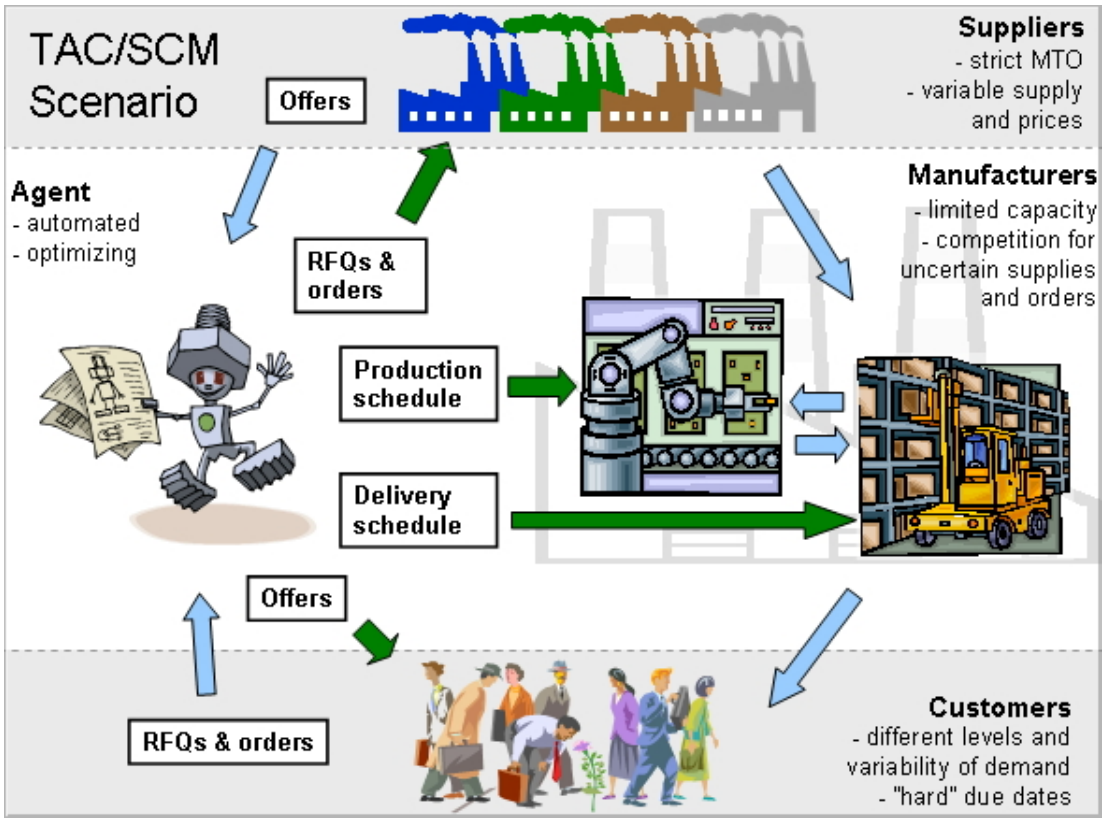
1.2.2 Methodological approach

In order to be able to answer the first question I’ve studied documentation about TAC SCM and compared it to literature about supply chain management in the real world. In addition to this I’ve implemented some simple types of agents for the TAC SCM game to get a better understanding about the platforms possibilities and limitations.

To answer the second question I decided to make the best agent I could to counter the simple types of agents that I had created. Furthermore I downloaded the winner of the TAC SCM 2003 championship and pitted it against my agents to see how well my agents would perform against such a formidable opponent.

2 GAME OVERVIEW

In TAC SCM six personal computer assembly agents compete for customer orders and for procurement of a variety of components over a period of several months. Each day customers issue requests for quotes and select quotes submitted by the agents, based on price and delivery dates. The agents are limited by the capacity of their assembly lines and have to procure components from eight different suppliers. Four types of components are represented in the game: Motherboards, CPUs, Memory and Hard drives. It features a variety of components of each type. Customer demand comes in the form of requests for quotes for some type of PC, requiring combination of different components.



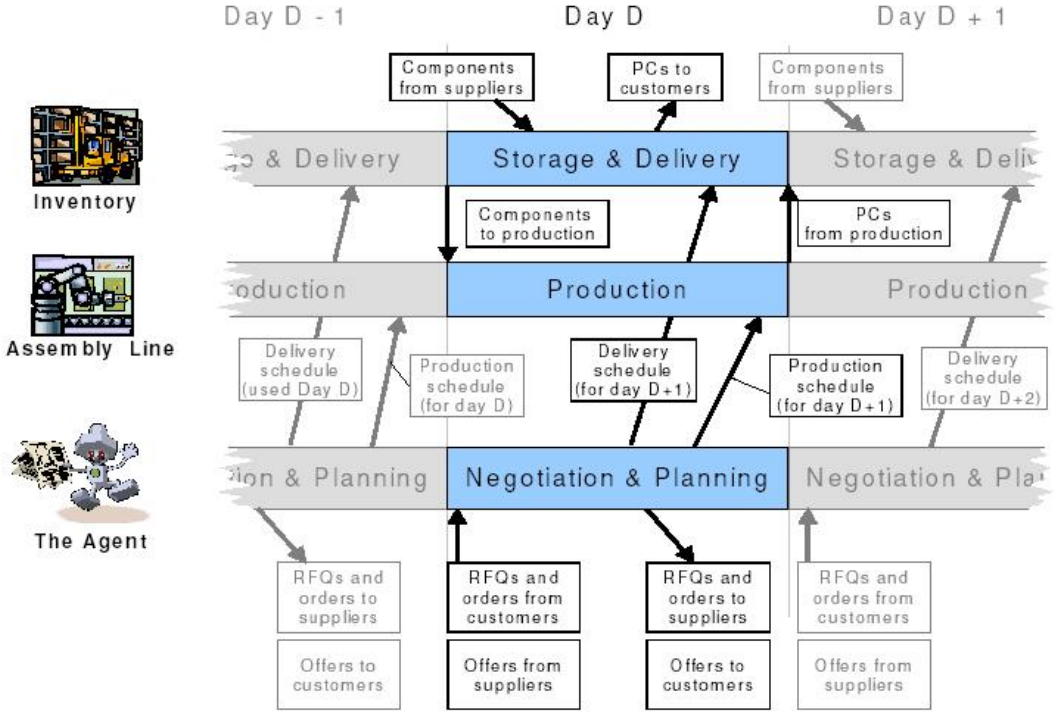
Picture taken from The Supply Chain Management Game for the Trading Agent Competition 2004 [8].

2.1 Game mechanics

Each agent is responsible for the following tasks:

- Negotiating supply contracts
- Bidding for customer orders
- Managing daily assembly activities

The three tasks are performed daily, as shown in the picture.



Picture taken from The Supply Chain Management Game for the Trading Agent Competition 2004 [8].

At the start of each day, each agent receives:

- From Customers:
  - Request for Quotes (RFQs) for PCs
  - Orders won by the agent in response to offers sent to the customer

- From Suppliers:
  - Quotes/Offers for components in response to RFQs sent to them the day before.
  - Delivery of supplies it had ordered earlier.

- From Bank:
  - Statement of the account

- From Factory:
  - Inventory report (quantity of different components and finished PCs available)

During the course of the day, the agent makes decisions regarding:

- Which customer RFQs to bid on.
- Which components to procure.
- Which supplier offers to accept.
- Which orders to put on the assembly line.
- Which assembled PCs to ship to which customers.

Each agent is endowed an identical factory containing an assembly cell capable of assembling any type of PC, and an inventory storing both components and finished PCs. Each PC type requires a pre-specified amount of time on the assembly line and each agent has a fixed amount of time for assembly available each day. Each day the agent makes a production schedule for its assembly cell. PCs in the assembly cell are processed sequentially until either capacity has been exhausted or no more components are available. At the end of each day, produced PCs are moved to inventory, ready to be shipped to customers the next day.

Shipping is controlled by the shipping schedule. The shipping schedule is processed sequentially until either all shipments have been made or no more shipments can be made due to lack of finished PCs in the inventory. An example:

Day D: At the end of the day, the agent sends a production schedule for day D+1, to the factory.

Day D+1: During the day the factory produces the requested PCs. At the end of the day, the agent sends a shipping schedule for day D+2, to the factory.

Day D+2: During the day, the PCs arrive to the customer.

3 SIMPLE AGENTS

In this section I will explain how 4 different agents using simple strategies work, and their results against one of the strategies, Agent Random, who makes most of its decisions randomly. The other three strategies are simple ad-hoc strategies that do not build upon any deep theory.

I chose these specific types of simple strategies because they are the extremes of their respective strategies. One agent based on randomization to add a bit of randomization to the game, another focused on sabotaging the environment, a third that is purely adaptive to the game environment and a fourth that play the game extremely aggressive.

3.1 Agent Random

This is a simple random agent. It makes key decisions based on randomization. However, it is not entirely random, as some decisions could never possibly be positive to the agent.

3.1.1 Setup

Decisions Agent Random does entirely random are:

- Decide from what factory to order from.
- Decide what customer RFQs to bid on.

Decisions Agent Random does semi-random are:

- Apply a random price discount for a bid to a customer, but never over 10%.
- Bid on between 20 and 70 customer RFQs everyday.

The reason for choosing 10% as the largest price discount given is that giving a larger discount will only result in loss of money for the agent and since this is supposed to be an agent that makes random decisions, it is impractical, if not plain stupid, to make decisions that are guaranteed to hurt the agent's results.

The factory every agent is given in the beginning of the game has got capacity to manufacture somewhere between 300 and 400 computers every day depending on the type of computers being made. Every order from a customer contains an average of about 10 computers, so if the agent receives more than 40 orders at any given day, chances are that it has taken on too much work. The reason I set the maximum to 70 orders is that only about half of every bid on a customer RFQ results in an order from that customer, so 70*10 / 2 = 350 computers which are within the agent's capacity to build. The reason I set the minimum to 20 is that if an agent has taken on more orders than it can handle, it starts getting late fees and in order to minimize those late fees the agent has to bid on fewer orders while still manufacturing computers for the late orders.

In order to keep this a simple agent, it has a fairly basic behavior.

Suppliers:

- It only accepts the earliest complete offer, rejecting partial deliveries.
- It always accepts bids from suppliers.
- It does not order components it does not need to complete its orders.

Manufacturing:

- Orders with the earliest due date get priority.

Delivery:

- When able to deliver an order to a customer, it always does.

Canceled orders:

- If a customer cancels an order, all the components needed for that order are made available for use in other orders.

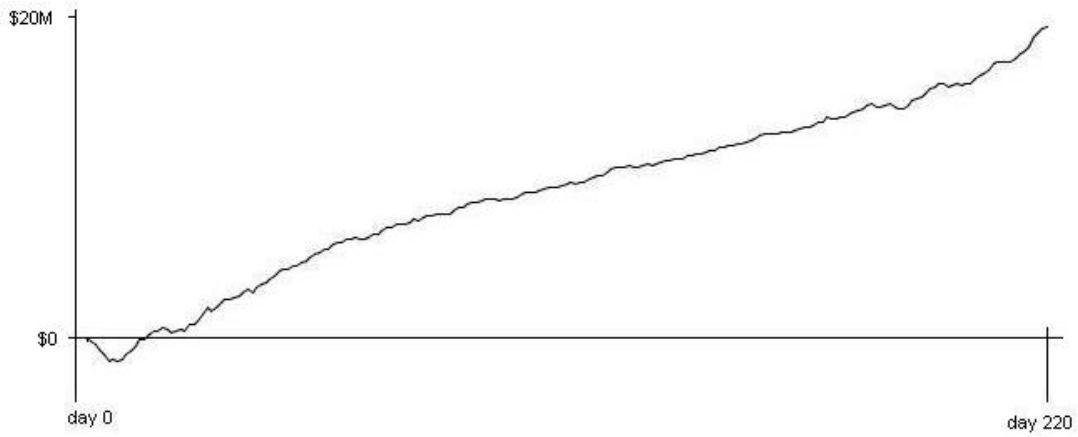
3.1.2 Results

After ten games with this agent I can say that its end results are quite random, ending up with between 10 and 30 millions (see table 1 for an average). However, as diagram 1 show, there is very little fluctuation in terms of profit during the entire game; every Agent Random's result looked like this in all the ten games. Even though they all make quite random decisions, they do not make many very stupid ones, so they are making a small profit almost every day.

[Table 1]

Agent Name	Average score (MDollars)	Standard deviation
AgentRandom	18,7	3,2
AgentRandom1	19,2	2,8
AgentRandom2	19,3	3,4
AgentRandom3	19,2	2,5
AgentRandom4	19,0	2,9
AgentRandom5	18,7	4,7

[Diagram 1] Overall result of AgentRandom of game 8 out of 10.



3.2 Agent Saboteur

This agent tries to sabotage the game environment by ordering all of a certain type of component. This agent should end up victorious in all games where the other agents have no protection against this kind of behavior.

3.2.1 Setup

To minimize the penalty for ordering all of one type of components, I chose the memory parts, because they were the cheapest. On the first day of the game, day 0, Agent Saboteur orders one million pieces of every kind of memory with the last day of the game as due date. Since the manufacturers cannot possibly create that many components, they estimate how many they can create until the last day of the game, and offer that amount (around 110 000). Agent Saboteur orders them. This means that every other RFQ for memory gets either no response, since the earliest day the manufacturers can deliver is the day after the game has finished. Should the manufacturer manage to produce more memory than originally expected (50/50 chance), the due date of these is most likely delayed to the point where the agent who ordered them no longer has any use for them.

3.2.2 Results

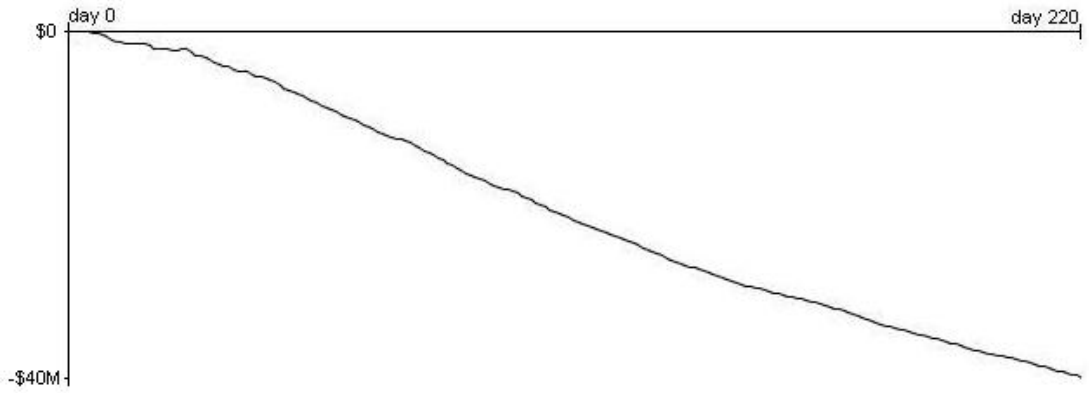
The downside of this approach is that on the last day of the game, Agent Saboteur gets all the memory pieces produced so far by all the memory manufacturers. This effectively puts the agent in debt (around $30 million), but since the other agents have played out all the days failing to deliver to customers, and stockpiling on the other kinds of components, Agent Saboteur usually wins by the end of the game anyway.

After ten games against Agent Random it is safe to say that if an agent does not check that it is able to receive all components for a specific order in time before accepting the order and ordering the components from the manufacturers, Agent Saboteur will end up with the least amount of loans from the bank. Table 2 shows the average after ten games. Even if Agent Saboteur will never be able to end up with any profit there is still a very good chance that it'll come out the winner. Diagram 2 shows the typical result of an Agent Random against Agent Saboteur.

[Table 2]

Agent Name	Average score (MDollars)	Standard deviation
AgentSaboteur	-29,7	1,6
AgentRandom1	-39,5	5,5
AgentRandom2	-40,0	5,3
AgentRandom3	-39,4	7,3
AgentRandom4	-40,6	3,6
AgentRandom5	-40,3	4,0

[Diagram 2] Overall result taken from AgentRandom1 of game 3 out of 10.



If there are two of Agent Saboteur in the same game focusing on the same type of component, one of them (the one who does not get the large order) will come out the winner, and all other agents will end up in the negative, see Table 3 for an average after 10 games..

[Table 3]

Agent Name	Score (MDollars)
AgentSaboteur	0
AgentSaboteur2	-30,3
AgentRandom1	-43,9
AgentRandom2	-44,7
AgentRandom3	-46,7
AgentRandom4	-44,5

3.3 Agent Calm

The idea behind Agent Calm is simply to adapt to the environment. If there is a shortage of a specific type of components, Agent Calm increases the number of that component to have in storage as a buffer. If there are not enough customer orders coming in, it increases the discount given to customers, if there are too many orders coming in, it decreases the discount given to customers. I named it Agent Calm because it does not do anything on its own accord; it only reacts to changes in the environment.

3.3.1 Setup

Agent Calm starts by ordering one of every component from every supplier to see what the current maximum amount of delay from the suppliers is. It then waits for the first market report which is sent to all agents every third day. Agent Calm checks the market report for average customer demand, and acts upon that information.

Agent Calm sets the amount of customer RFQs to respond to, to the total customer demand per day / 4 (there are six opponents in the game) in order to bid on a decent amount of customer RFQs. In doing so, the agent has a moderately low risk of receiving too many orders, but still gets more than its fair share of the market.

Furthermore, it sets the amount of components to have in store as a buffer by dividing the customer demand per day with 4 (the amount of orders bid on), then multiplying it with the current maximum delivery time of components from the suppliers, (the current maximum is set by saving the latest delivery time of every supplier, then searching for the maximum) and last multiplying by 2 to be able to deal with some deviations from the average delays.

Additionally, the agent sets the discount rate every day by checking how much of the factory capacity is used, if more than 80% is used, the discount given to customers is decreased by 1% to a minimum of 0%. If the factory is used to less than 50% of its potential, the discount is increased by 1% to a maximum of 20% discount. More than 20% discount would result in extremely small profits or even a loss at every sale.

The last thing that is dynamically set by observing the environment is what supplier to order from. If two suppliers are able to produce the component wanted, Agent Calm will 3 out of 4 times choose the one which, on the last order, had the least amount of delivery delay. The reason for doing this is to frequently test if the delivery delay has decreased for the other supplier.

3.3.2 Results

After ten games against Agent Random I can say that this agents ability to adapt when the environment is changing is great. One game even resulted in every other agent getting a negative score ranging from $-7 million to $-800,000 while Agent Calm still managed to get a positive $12 million.

This approach seems to be working very well. Table 4 shows an average out of ten games against Agent Random. If combined with an aggressive style this should become a very competitive agent.

[Table 4]

Agent Name	Average score (MDollars)	Standard deviation	Customer orders received
AgentCalm	24,7	3,2	4193
AgentRandom1	14,6	3,1	2490
AgentRandom2	15,0	3,1	2565
AgentRandom3	14,4	3,3	2485
AgentRandom4	14,6	3,2	2497
AgentRandom5	14,8	3,0	2524

3.4 Agent Aggressive

Agent Aggressive applies a very aggressive style right from the beginning. A bit like Agent Saboteur, but with a focus on delaying other agent's orders to get them to have some amount of late deliveries every day, giving them severe penalties through the entire game, while still being able to manufacture and deliver computers to customers and thus ending up with more money than it started with.

3.4.1 Setup

From the start Agent Aggressive orders 2000 pieces of memory and 1000 pieces of every other type of component from the different suppliers. Throughout the game, Agent Aggressive uses the 1000 initial pieces of equipment as a buffer for the occasional shortage of some component type. The 2000 pieces of every type of memory serve as an initial attempt to ruin production schedules for the other agents, because of the extra delay induced. Then Agent Aggressive then proceeds in bidding for customer orders and ordering components from factories in a normal fashion.

Day D: Bids on 50 customer RFQs with a 10% discount.

Day D+1: Receives orders from customers, send RFQs for all the components to factories, and start manufacturing the ordered PCs. Additionally, Agent Aggressive takes care of the activities on day D.

Day D+2: Ships completed orders to customers, as well as carrying out the activities on day D+1.

Every 40th day of the game Agent Aggressive orders 2000 pieces of memory to sabotage the environment. If the buffer of components goes below 200 pieces, 50 components of that type are added to the next order to increase the buffer size. If the buffer of components exceeds 1200 pieces, 50 components of that type are removed from the next order to decrease the buffer size and reduce storage costs.

If Agent Aggressive ever gets a penalty for a late order, no more customer RFQs are bid on until the day Agent Aggressive no longer has any late orders outstanding.

When there are 20 days left of the game Agent Aggressive tries to sell all of the components in the buffer before the game ends, by simply removing the number of buffered components from subsequent orders.

3.4.2 Results

After ten games against Agent Random it is safe to say that this approach is very profitable, every game was won by Agent Aggressive.

[Table 5]

Agent Name	Average score (MDollars)	Standard deviation
AgentAggressive	30,8	3,6
AgentRandom1	15,3	2,5
AgentRandom2	15,2	2,2
AgentRandom3	15,4	2,0
AgentRandom4	15,2	2,1
AgentRandom5	15,2	1,9

The attempt to sell off the buffer was moderately successful, it is able to sell off about half of the components before getting a shortage of one type required to manufacture more PCs.

With two Agent Aggressive agents and four Agent Random in the same game, the two Agent Aggressive will win every time because of the large penalties induced on the other agents due to the lack of memory components. You can see the results of such a game in table 6.

[Table 6]

Agent Name	Score (MDollars)	Late Orders
AgentAggressive	21,4	150
AgentAggressive2	16,5	436
AgentRandom1	-1,6	1339
AgentRandom2	-0,5	1075
AgentRandom3	-0,3	1113
AgentRandom4	-0,1	1031

4 AN ADVANCED AGENT STRATEGY

Advanced Agent takes the behavior of both Agent Aggressive and Agent Calm, and improves it to a standard where it is able to beat any of the 4 simple types of agents.

4.1 Advanced Agent Setup

The very first day in the game, Advanced Agent orders 1000 of every component from every supplier to get a starting buffer so it will be able to manufacture computers from the day the order from the customer is received. In the same vein as Agent Aggressive, Advanced Agent orders 4000 extra units of memory to induce some lag to the other opponents right from the beginning. Also, every day until the agent has received all of the components ordered the first day; it orders 50 of every type of component. In doing so, the agent will have a steady supply when it starts to manufacture computers.

Every 40^{th} day, with the exception of the last 30 days in the game, Advanced Agent orders 4000 extra units of memory to sabotage the environment and induce some lag on its opponents. However, if Advanced Agent already has more than 2000 units of any memory part in storage, it does not order the extra 4000, because if the agent has a good supply of memory parts, so does every other agent, and ordering 4000 extra components will not make a difference.

Every day, except for the first 15 and the last 20, the agent checks the inventory for each component. If the amount in storage is less than 700 pieces, the agent adds 100 pieces to the next order of that component type; if it is less than 400, it adds 300 pieces to the next order. If the amount of components in storage is above 1000, it removes 50 pieces from the next order, and if it is above 1500, it removes 250 pieces from the next order. This procedure keeps the buffer somewhere between 700 and 1000 components for every component, all of the time. The exception is that memory components are not checked for 30 days after every order of 4000 extra memory units, because if they were, Advanced Agent would end up with a 0 buffer for memory components since every day it has got more than 1500 units in storage the buffer for memory components will decrease by 250 units. The last 20 days, all buffers are reset to 0 to use up as many components in storage as possible. However, since the most expensive component, the CPU, usually costs between 3 and 4 times as much as any other component, Advanced Agent continues to use buffers for all components except the CPU's to use up as many of those as possible.

If, at any time, the number of a component in storage reaches 0, 1000 of that component are ordered the first day, and until the day it has more than 100 components in storage, Advanced Agent orders 100 pieces of that component. During this time the agent does not bid on any customer orders since it is unable to manufacture computers with that component.

Every day, except the first few days until the agent has received the buffer components, the amount of customer RFQs to bid on is decided by 2 things. The first is the customer demand shown in the market report that is sent to all agents every third day, the agent bids on one quarter of the customer demand to get a reasonable market share. The second is a function that is designed to keep the agent from bidding on too many or too few orders in situations where there are very few or very many customer RFQs coming in. The function looks like this:

Computers ordered the last day = x
Number of RFQs bid on last day = y
((350 – x) / (x / y)) * 2

350 is the target number of computers the agent wants to receive orders of. The function is multiplied by 2 since far from every extra response to an RFQ results in an extra order. Also, the amount of RFQs bid on is never allowed to go above 80 or below 45 (1 RFQ roughly translates into 10 computers).

The price offered to customers is influenced by how much capacity the agent's factory has left after manufacturing computers for all the current orders. If the factory is working above 80% of its capacity, the discount given to customers is reduced by 1%. If the factory is working at less than 50% of its capacity, the discount is increased by 1%. The maximum amount of discount given is 20% and the minimum is 0%.

The market report is also used to set the overall buffer level of all the components, as a function of the customer demand and the max delay from the last order from the manufacturers. The function looks exactly like the one used in Agent Calm:

Customer demand / 4 * maximum delay * 2

The maximum size of the buffer is set to 1500 and the minimum to 500, to reduce the effect of extreme changes in the environment.

In the last 20 days of the game, this overall buffer is set to 0 to use up as many of the components in storage as possible.

When a customer order is received, the amount of components is added to the next order from the correct manufacturer. Production is started right away and deliveries are made as soon as the computers are ready. If two manufacturers are supplying that type of component the one with the shortest delivery time on the last order will be used three out of four times. The reason is to check from time to time if the one with a longer delivery time has become better.

When Advanced Agent acquires more than 20 late fees from orders not delivered yet, no more customer RFQs are bid on until the day there are less than 20 late fees to pay.

4.2 Countering Agent Random

Agent Random is quite easy to counter. While almost anything can happen in a game with many Agent Randoms, the likelihood of any extreme situations are very low. Furthermore, Agent Random is in no way capable of dealing with those extreme situations. So, to counter Agent Random one has to make either a big unexpected change to the environment, like buying a lot of memory parts and thus delaying orders for the other agents, or adapting to the environment and using it to its maximum potential. The Advanced Agent does both, and as you can see in Table 7, which shows the average score after ten games, this approach is very successful against Agent Random.

[Table 7]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	33,2	4,8
AgentRandom1	13,4	3,5
AgentRandom2	13,6	3,0
AgentRandom3	12,8	3,0
AgentRandom4	12,9	2,9
AgentRandom5	13,1	3,3

4.3 Countering Agent Calm

Agent Calm is a bit trickier to counter. It is just not enough to create large changes to the environment, since Agent Calm will just adapt and continue working. In order to defeat Agent Calm one has to attempt to change the environment around as much as possible during the game. This is most easily done by ordering a lot of one specific component at regular intervals. Advanced Agent orders a large amount of memory components every 40^{th} day, making it sufficiently capable of countering Agent Calm. As you can see in Table 8, which shows the average score after 10 games, this strategy defeats Agent Calm.

[Table 8]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	27,8	4,7
AgentCalm	19,6	3,0
AgentRandom1	7,8	4,4
AgentRandom2	8,0	5,0
AgentRandom3	7,6	4,8
AgentRandom4	8,3	4,3

4.4 Countering Agent Saboteur

To win against Agent Saboteur one only has to limit the amount of penalty received from late orders. The simplest solution is refusing any orders you are unsure about completing. However that approach will hamper the agent's performance more than necessary. Instead, the ideal approach is to adapt to the changing environment and start to more aggressively order components of the type that there is a shortage of, and as long as the agent has none of that component in storage, not accept any more customer orders. When Advanced Agent gets low on a component, it makes more and larger orders for that component If that does not help and the storage for that component drops to 0, a very aggressive ordering from manufacturers starts and no more customer orders are accepted until the storage level for that component is at a minimum of 100 parts. As you can see in Table 9, which shows the average out of ten games, this approach even manages to end up with a positive balance most of the time (only one game out of ten ended with AdvancedAgent having a negative score)

[Table 9]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	3,3	3,0
AgentSaboteur	-29,1	1,3
AgentRandom1	-48,0	7,2
AgentRandom2	-47,9	7,4
AgentRandom3	-48,4	8,3
AgentRandom4	-47,9	7,3

4.5 Countering Agent Aggressive

Beating Agent Aggressive in every game is almost impossible. However, since Agent Aggressive does not have many countermeasures against large changes to the environment, there are two ways of beating it. Adapting to the environment and using it better than Agent Aggressive, or inducing massive changes to the environment to penalize Agent Aggressive for not taking enough preventive measures (although there are other ways to deal with extremely aggressive agents [6], these are the simplest, and simplicity is often a determining factor in success). These ways will not always succeed though, since if there is a large surplus of components, then Agent Aggressive will be very effective as it does not really slow down while it still has some components in the inventory. The way to defeat Agent Aggressive is not taking as many huge risks as it does, and although this approach won't win every game, most games will be won and the overall score will be better. Table 10 shows the average of ten games where AdvancedAgent won seven and AgentAggressive won three.

[Table 10]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	25,6	4,1
AgentAggressive	24,6	3,3
AgentRandom1	8,2	4,9
AgentRandom2	7,9	5,2
AgentRandom3	7,5	4,8
AgentRandom4	7,8	5,0

4.6 Countering a mix of agents

The way to win against a mix of agents using a variety of different strategies is simply to use every countermeasure against the other types at the same time. Adaptation to the changing environment and some aggressiveness is required. This will not result in the best score every time, but the overall score will be the best and most of the games will be won. In Table 11 you see the result of ten games with Advanced Agent against Agent Random, Agent Calm, Agent Aggressive and Agent Saboteur. Of those games AdvancedAgent won nine and AgentAggressive won one. Table 12 shows the results of ten games with Advanced Agent against Agent Random, Agent Calm and Agent Aggressive. Since Agent Saboteur is using a very special strategy I thought it would be interesting to pit every agent except that one against each other, as most games where every agent is playing to win will not have an Agent Saboteur in it. Of the ten games in Table 12 AdvancedAgent won every single one.

[Table 11]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	1,2	1,7
AgentAggressive	-11,9	5,2
AgentCalm	-21,6	2,2
AgentSaboteur	-28,9	0,9
AgentRandom1	-51,9	6,5
AgentRandom2	-51,7	6,3

[Table 12]

Agent Name	Average score (MDollars)	Standard deviation
AdvancedAgent	25,2	2,5
AgentAggressive	19,4	1,8
AgentCalm	16,7	4,1
AgentRandom1	6,4	4,5
AgentRandom2	6,4	4,2
AgentRandom3	6,2	4,4

4.7 RedAgent

RedAgent was the winner in the TAC SCM ’03 championship, so I thought it would be interesting to pit my Advanced Agent against it. Not totally surprising, RedAgent outperforms AdvancedAgent by far. The most significant difference between them is that RedAgent is a multi agent system, while AdvancedAgent is simply one agent trying to do everything. However, there are also large differences in their strategies that I took notice of. RedAgent starts the game by ordering between 10 000 and 25 000 components of each type, and starts making computers whenever it has enough components in storage to do so. Then it only accepts orders for computers it has ready for shipping. This successfully helps RedAgent use all of its factory’s potential every day which in turn maximizes its profits. The downside of this approach is that RedAgent will have a very low score after ¼ of the game, and not get a positive score until about half way into the game, but since the only score that matters is the one the agent has when the game is over, this does not hurt much. Table 13 shows the average score of ten games between RedAgent, AdvancedAgent and four AgentRandom.

[Table 13]

Agent Name	Average score (MDollars)	Standard deviation
RedAgent	62,7	13,1
AdvancedAgent	5,9	3,1
AgentRandom1	2,7	4,3
AgentRandom2	2,8	4,0
AgentRandom3	2,9	4,1
AgentRandom4	2,7	4,2

Since RedAgent is so focused on having a lot of components in storage at any time I thought it would be interesting to pit it against Agent Saboteur, since that agent will probably damage RedAgent’s initial tactic seriously. Table 14 shows the average of ten games between RedAgent, AdvancedAgent, AgentSaboteur and three AgentRandom. It is interesting to note that RedAgent only won 8 out of 10 games, and AdvancedAgent won the other 2 games.

[Table 14]

Agent Name	Average score (MDollars)	Standard deviation
RedAgent	8,5	29,6
AdvancedAgent	-19,3	3,6
AgentSaboteur	-28,2	1,5
AgentRandom1	-47,8	6,2
AgentRandom2	-49,0	6,4
AgentRandom3	-49,1	7,0

The reason RedAgent's score is so fluctuating is of course because of AgentSaboteur's act of sabotage to the game environment by ordering every memory component at day 0. If the memory suppliers choose to expedite RedAgent's order first, it will end up with a positive balance. However, if the suppliers choose to expedite AgentSaboteur's order first, RedAgent will end up with a lot of other types of components, but no memory. The only way out is then to order as many pieces of memory as possible of the extra ones the suppliers happen to produce in addition to the ones already ordered by AgentSaboteur. To increase the chance of the suppliers expediting AgentSaboteur's order before RedAgent's, I ran ten games with 1 RedAgent, 1 AdvancedAgent and 4 AgentSaboteurs. The result of these ten games is shown in Table 15.

[Table 15]

Agent Name	Average score (MDollars)	Standard deviation
RedAgent	-33,2	25,9
AdvancedAgent	-28,1	3,7
AgentSaboteur1	-7,1	4,2
AgentSaboteur2	-6,9	5,8
AgentSaboteur3	-6,7	4,7
AgentSaboteur4	-7,1	6,4

RedAgent won 2 games out of 10, and six games out of four, AdvancedAgent finished with a score higher than that of RedAgent. However, AdvancedAgent was never able to get a positive score, and hence, it never won a single game. The reason for this is that with four AgentSaboteur in a game, all trying to, at day 0, order every memory component the suppliers will produce during the game, at least one of them gets refused by the suppliers because of lack of memory components. The agent, whose orders are refused, will end up with a score of 0. The trick RedAgent uses to get a positive score is to place orders not just for the initial component procurement but for most of the game, at day 0. The effect of this plus Agent Saboteur, who orders almost every piece of memory from day 0 to the last day in the game, is an incredibly hostile environment. It is impressive that RedAgent manages to adapt to even this environment and finish with a positive score. RedAgent's aggressive day-0 ordering of components and its great adaptability is simply amazing; I believe there was no luck involved when it won TAC SCM 03.

5 DISCUSSION

These experiments were done by examining how well different simple agents plays against a more advanced one so one could argue that selection of simple agent types has more importance than the results. My selection of the simply types of agents is purely based on my experience with games in general. The simple types were built to represent the 4 largest styles of play; Aggressive(Agent Aggressive), Tight(Agent Calm), Loose / playing badly(Agent Random) and just trying to ruin the fun for everyone else(Agent Saboteur). This way the number of different situations that can occur with different combinations of these agents are maximized. I therefore think my choices of these types of simple strategies are good ones.

Unfortunately, there are very few literary studies made regarding TAC SCM, which is why there's no part in the thesis about this. The TAC SCM is a very new game (only about 1½ years old) so there hasn't been made any large studies about it yet.

One could argue that only running each scenario 10 times will open up possibilities for dirty values having more importance than they should. I disagree with this because while the TAC SCM game has some degree of randomness the agents have not (with the exception of Agent Random, which still only has a small degree of randomness built in). Therefore each game of a specific scenario will play out about the same way every time.

6 CONCLUSION

The conclusion I draw from my experiments is that the TAC SCM game, while fun and entertaining, is nowhere near the real world in terms of complexity. I base this on a number of articles ([4], [9]) and books ([7], [5]) I've read and the knowledge that I've gained building these agents and watching them play. As Chopra and Meindl say it in their book [7] "A supply chain consists of all stages involved, directly or indirectly, in filling a customer request. The supply chain not only includes the manufacturer and suppliers, but also transporters, warehouses, retailers, and customers themselves." There are just too many factors from the real world missing in the TAC SCM game, to make it relevant for companies in the real world, a few of them are; Logistics, new products hitting the market, 3rd party knockoffs, product quality and industrial espionage.

It could be an interesting experiment to try real world strategies for supply chain management in the TAC SCM game, but the game is nowhere near advanced enough to model the real world in a relevant way. It is simply a sandbox for people who want to experiment with different types of agents.

To make the TAC SCM game more interesting a number of things could be added. The most interesting one would be the ability to see some of the moves the opponents make, since that would open up many possibilities for different strategies interesting to experiment with in the research of agents, and at the same time bring the game closer to reality. Another thing that would be interesting to include in the TAC SCM game is packing and shipping costs of products from manufacturers and to customers. Keeping these costs down is a serious problem for many companies in the real world ([9], [4]) and having agents research this area would probably provide helpful information on how to deal with such problems.

Unfortunately, the TAC SCM game does not supply any information regarding the choices the opponents make, so creating models of them is impossible at this time.

However, the factors that are presently implemented in the game seem to be working very well. The TAC SCM game is advanced enough to resist being totally dominated by simple strategies, and the most important ability of a TAC SCM agent is adaptability, to be able to handle every situation without making more than a few wrong moves. There always seems to be some possible way of countering simple strategies like Agent Saboteurs that are designed to beat the game instead of using a profitable strategy, or Agent Aggressive that relies on the fact that there are no big changes to the game environment.

6.1 Future work

There are two paths within this subject that would be interesting to explore. The first is how well strategies that are used in real life would do in games against agents designed specifically for the TAC SCM game; the second just how well agents would do in the real world. Although there has been some research made on how well supply chain management agents would work in the real world [3], I have not seen any comparisons between TAC SCM agents and pure real world strategies, in a TAC SCM game.

In my experiments I did not look at all at supply chain strategies that are used in real life, only how I could beat the game using simple strategies derived from reading about more advanced agents and personal experience. It would, however, be interesting to merge different advanced successful strategies like agent "shimon"'s [1]

variant of the pull strategy with “Deep Maize”’s [6] equilibrium analysis in the game environment.

REFERENCES

[1] Shimon Whiteson and Saurabh Amin. An Autonomous Agent for Supply Chain Management. University of Texas at Austin, Dept. of CS, 2001.

[2] Michael Wooldridge, An Introduction to MultiAgent Systems, John Wiley & Sons Ltd., ISBN: 0-471-49691-X

[3] Ravi Kalakota, Jan Stallaert and Andrew B. Whinston. Implementing Real-time Supply Chain Optimization Systems. Technical report, University of Texas at Austin. Dept. of MSIS, 22/10/1996

[4] Robert Malone, Harnessing the Power of Collaboration, Inbound Logistics, issue July 2003

[5] Martin Christopher, Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service, Prentice Hall Inc. ISBN: 0273630490

[6] Joshua Estelle, Yevgeniy Vorobeychik, Michael P. Wellman, Satinder Singh, Christopher Kiekintveld and Vishal Soni. Strategic Interactions in a Supply Chain Game. University of Michigan, AI Laboratory, 02/04/2004

[7] Chopra Sunil, Peter Meindl, Supply Chain Management: Strategy, Planning, and Operations, Prentice Hall Inc., ISBN: 013101028X

[8] Raghu Arunachalam, Norman Sadeh , Joakim Eriksson, Niclas Finne and Sverker Janson. The Supply Chain Management Game for the Trading Agent Competition 2004. CMU-CS-04-107.

[9] Melissa Campanelli, Who Pays to Get it There?, Entrepreneur magazine, issue Feb 2002.