

PackaTAC: A Conservative Trading Agent

Erik Dahlgren

2003

Thesis for a diploma in computer science, 20 credit points, Department of Computer Science, Faculty of Science, Lund University, Sweden, in cooperation with Department of Computer Science, North Carolina State University, USA

PackaTAC: A Conservative Trading Agent

Erik Dahlgren

Abstract

The Trading Agent Competition is an event fostering research in automating trading strategies. 2003 was the fourth time the event was held, although it was the first year for the new Supply Chain Management game. The game was designed to offer new challenges after TAC Classic, a game that involves a travel-shopping scenario, had been studied for a few years. TAC-SCM presents a challenging manufacturing scenario where agents compete for customer demand and supplies needed to produce the demanded products. The entry of North Carolina State University, PackaTAC, is a relatively simple agent that plays a conservative, low-risk strategy. Nevertheless, it performed quite well in the 2003 competition held at the International Joint Conference on Artificial Intelligence in Acapulco.

Contents

1	Introduction	7
1.1	Background	7
1.2	The Game	7
1.3	Key Problems	10
2	Strategies	12
2.1	Specialist Strategy	13
2.2	Generalist Strategy	14
2.3	Evaluation	15
3	Supplies	18
3.1	Processing by the supplier	19
3.2	Bid-order Strategy	20
3.3	Small-order Strategy	20
3.4	PackaTAC	21
3.5	Supplier Modeling	22
3.6	Accepting Offers	23
4	Scheduling	26
4.1	Simple Heuristic	26
4.2	Integer Programming	27
4.3	Better Greedy Algorithm	28
4.4	Shipping	31

5 Bidding	33
5.1 Bidding Simulation	33
5.2 Bidding Models	35
6 Results	42
6.1 Qualifying and Seeding Rounds	42
6.2 Finals	42
7 Other Agents	45
7.1 HarTAC, whitebear, PSUTAC, Tac-o-matic, Socrates, and jackaroo	45
7.2 RedAgent	46
7.3 deepmaize	47
7.4 Botticelli and TacTex	48
7.5 PackaTAC	50
8 Possible Improvements	51
8.1 Supplies	51
8.2 Ranking and Bidding	53
8.3 Profit per Product	54
9 Conclusion	55

List of Figures

1	Schematic of the game	8
2	Schematic of supplier negotiation.	19
3	An example of agents competing for supplies.	24
4	Time consumption of integer programme	29
5	Improvement of integer programme over greedy algorithm	30
6	<i>Expected profit – penalty</i> as a function of number of orders won	34
7	Revenue as a function of cycles needed to complete all orders	35
8	Curve showing probability of winning as a function of profit	36
9	Typical graph showing the pattern of ordering supplies	45
10	Curve showing the component holdings	46
11	The pattern of supply ordering for RedAgent	47
12	Curve showing component holdings for RedAgent	47
13	Graph showing the probing of the suppliers by deepmaize	48
14	Strategy for component holding for deepmaize	49
15	Strategy for component holding for the final round	49
16	Typical graph for component holding for Botticelli and TacTex	50
17	Curve showing the ordering strategy for Botticelli	51
18	Curve showing the holding strategy for PackaTAC	52
19	Results in the seeding and final rounds	57

List of Tables

1	The components available in the game.	9
2	The different types of computers that can be built.	10
3	The potential for profit for each type of computer	16
4	Results determining how many types of PCs to produce	17

1 Introduction

1.1 Background

Effective supply chain management is essential to a manufacturing company in order to effectively adapt to changing markets and demands. Today's supply chains are mostly static, and are based on long term relationships between trading partners. As the market changes, more dynamic solutions have the potential to find better matches between producers and consumers. However, due to the complexity in supply chain relationships, and the difficulty in effectively supporting dynamic trading practices, these solutions are very hard to adopt.

The TAC Supply Chain Management game was designed to provide a setting where some of the real world challenges regarding planning and coordination in a dynamic supply chain can be explored. This involves distributed, decentralized planning and coordination over the whole supply chain, based on self interest. At the same time the agents are bidding for supplies and consumer orders as well as planning production and shipment of the end product.

1.2 The Game

TAC SCM is a contest in which six agents compete for customer orders, while at the same time compete to acquire the needed supplies, Figure 1. The agents also have to plan their production and shipping schedules. Each game in the competition lasts for 220 days, making any impact of the startup and end phase fairly small on the overall game. See [1] for the game specifications.

The agents assemble PCs with CPUs, motherboards, memory and hard drives, for a

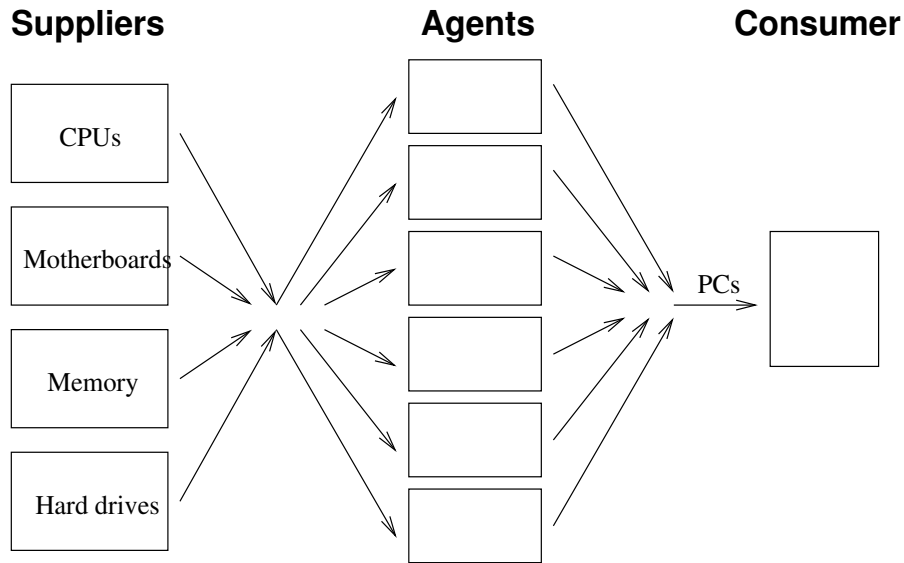


Figure 1: Schematic of the game.

total of 16 available PC types. The customers issue orders every day that specify which PC type they want, the latest day they want the PCs delivered, and the maximum price they are willing to pay for the order, called the customer's reserve price. There is also a penalty specified that the agents have to pay per day if the order is delivered after the agreed upon due date. If the order has not been delivered five days after the due date, the customer cancels it.

In order to build the computers the agents need to acquire supplies. There are four different components including CPU, motherboard, memory, and hard drive. Each component, except the CPU, comes in two different specifications, one high end and one low end, Table 1. The CPU comes in four unique products but with two different brands. Both CPUs of one brand can only be assembled together with a motherboard of the right type. Each component, except the CPUs, is manufactured by two different suppliers. The CPUs

Component	Base Price	Supplier	Name
100	1000	Pintel	Pintel CPU 2.0 GHz
101	1500	Pintel	Pintel CPU 5.0 GHz
110	1000	IMD	IMD CPU 2.0 GHz
111	1500	IMD	IMD CPU 5.0 GHz
200	250	Basus, Macrostar	Pintel Motherboard
210	250	Basus, Macrostar	IMD Motherboard
300	100	Mec, Queenmax	Memory 1 GB
301	200	Mec Queenmax	Memory 2 GB
400	300	Watergate, Mintor	Hard disk 300 GB
401	400	Watergate, Mintor	Hard disk 500 GB

Table 1: The components available in the game.

are manufactured by two suppliers, each making two of the four CPUs, Table 2. This gives a total of ten different components, eight suppliers, and 16 possible ways to assemble a PC. In general, the consumer is willing to pay more for a PC with higher grade components.

The suppliers are manufacturing their components with a factory whose capacity varies according to a mean-reverting random walk. However, when responding to orders from agents, they base on the mean capacity of the factory. Thus, the suppliers may produce either more or less than they had planned, and an agent may have to wait to get the supplies they ordered. If a delivery is pushed back due to low production, it is given priority over all future orders until it is filled.

One day after the agents issued orders for supplies, the suppliers respond with counter offers. The counter offer contains one or two offers, an *earliest complete* offer, and maybe also a *partial* offer. The earliest complete offer specifies the earliest day the supplier have the whole requested amount available, while the partial offer specifies the amount the supplier has available on the requested due date. These offers are valid for one day, so the agents need choose which one, if any, to accept, and send the acceptance message the

SKU	Components	Cycles
1	100,200,300,400	4
2	100,200,300,401	5
3	100,200,301,400	5
4	100,200,301,401	6
5	101,200,300,400	5
6	101,200,300,401	6
7	101,200,301,400	6
8	101,200,301,401	7
9	110,210,300,400	4
10	110,210,300,401	5
11	110,210,301,400	5
12	110,210,301,401	6
13	111,210,300,400	5
14	111,210,300,401	6
15	111,210,301,400	6
16	111,210,301,401	7

Table 2: The different types of computers that can be built.

same day the offer was received.

1.3 Key Problems

There are a number of key problems that need to be addressed by an agent to be successful.

Each simulated day, the agent must decide:

- Which customer orders to bid on and how much to bid.
- Which components to order, what quantity to order, which day the order is due, which supplier to order from, and which counter offers to accept.
- Which PCs to build in the factory.
- Which PCs to ship to customers.

These decisions must be made in a timely manner because each day lasts for 15 seconds. Also, there is very minimal information available to base the decisions on. These problems and our solutions to them are covered in more detail in the rest of the thesis.

2 Strategies

The three key decision to make are what to bid on, what to order, and what to produce. The last one is far easier than the other two, because once the agent knows what resources it has and the computers it needs, even a very simple approach will produce something useful. However, if the bidding on orders or the ordering of supplies is too simple, the agent might find itself either without customer orders, or without supplies. There are several basic strategies to choose from.

One approach is to hold a certain level of inventory at all times. If a sufficient quantity of each supply is on hand then when an agent wins an order, the computers can be produced. This gives the agent the flexibility to build many types of computers. However, the factory has only 2000 factory cycles per day, which limits the number of orders that can be satisfied per day.

Another strategy is to hold a minimum level of built computers in inventory. The computers can then be shipped as they are needed. An agent with such a strategy can tolerate fluctuations in demand. When demand is low, the agent can build many computers. Later, when demand is higher, it can handle many more orders than in the previous strategy. This is because the computers are already produced, and there is not limit in how many computers can be shipped during one day.

A thirds approach is for the agent to order supplies when it win orders. This way there are no unallocated supplies on hand. An agent with this strategy will not be in debt if it does not win many orders. In contrast, if using the first two strategies, an agent must buy supplies without knowing if they are going to be needed. If a game has very low demand

for a large portion of the time, the agent may not get out of debt. By ordering supplies only when they are needed this can never happen. On the other hand, if the agent wins a customer order but can not get supplies in time, it will accumulate penalties instead.

2.1 Specialist Strategy

Some of the problems in the above strategies can be alleviated by choosing to specialize in only a few products. Under this strategy, the agent would pick a smaller subset of the 16 types of PCs and produce only computers in this set. The set could be determined in different ways: fixed from the beginning of the game, based on what orders the agent wins and what is perceived as the more profitable PCs, or based on perceived supplier performance and cost. The set is then evaluated and changed at intervals throughout the game. These intervals could, for example, be when the market report is issued every 20 days.

The main advantage of specialization is in reducing the number of suppliers and supplies. The subset can be chosen in a way to minimize the number of different parts, thus making the agent dependent on fewer suppliers. The choice could then be based on which suppliers are producing over the nominal value of 500 units per day, and which are under-producing. One can then specialize in easy-to-get parts. On the other hand, the subset could be based on that number of cycles it takes to produce each PC, thus maximizing factory output, or minimizing it by building high end PCs, which in general have a higher profit. For instance one could choose the set with SKU numbers $\{1, 2, 9, 10\}$ which would require parts $\{1, 3, 5, 6, 7, 9, 10\}$ and have an average of 4.5 production cycles per computer. In comparison, the set $\{1, 2, 5, 6\}$ would only use parts $\{1, 2, 5, 7, 9\}$ but instead

require 5 cycles for the average computer.

Another advantage of specializing is to keep some inventory of supplies and possibly pre-produce some PCs if there are otherwise unused cycles left some days. Both of these options are possible without specialization as well, but are less efficient and may induce a higher cost. Because the set of parts needed is smaller than if not specializing, it is possible to have unused supplies kept in inventory. While there is no cost for storage, there is a loss in interest on the money tied up in inventory for the days the agent holds it. If large quantities are unused, interest may be payed on debt from buying the supplies, instead of earned on profit. Without specialization it is also less efficient to pre-produce, since it is harder to decide on which PC to produce without an order. The agent may produce PCs which end up not being sold for a long time.

The whole idea behind this kind of specialist strategy is to divide the demand among the manufacturers. Each agent would then have a monopoly on a subset of the PCs, which enables them to avoid competing and extract maximal profit from the consumer.

2.2 Generalist Strategy

We characterize an agent who choose to build a larger subset, or the whole set, of PCs as a generalist. While the specialization subset would be fairly small, the generalization one would be fairly large. The advantages of being a generalist are obvious in low demand situations. When there are few Requests For Quotes (RFQs) from the customer, there are few opportunities for a specialist to bid on PCs. A generalizing agent would run a lower risk of finding itself in that kind of a situation because it can bid on more types of computers. Furthermore, the success of a specializing agent depends on successful coordination with

other agents. A generalizing agent has a greater ability to adapt to fluctuations in demand. The specializing agent is more dependent on producing the whole subset of computers, and thus may find itself having to either try to under-bid competitors for the same type, or having to choose a new subset. These problems inhibit the advantages of specializing and may lead to price wars and very low profits or possible losses to maintain market share while punish competitors who try to take over or break into your market share.

2.3 Evaluation

In our initial analysis there seems to be enough demand most of the time to support specialization. Per day there is roughly 80 to 6400 computers ordered. On average that leads to 3240 computers per day. The agents have a production ceiling of 2000 cycles per agent per day. With four to seven cycles required per computer, all the agents together can produce between 1700 and 3000 computers per day. Thus, all agents should be able to specialize with their own non-overlapping market share on the consumer side. Each manufacturer would then be able to extract maximal profit[4]. Although, they would still be competing for supplies.

However, if everyone specializes, some subsets will be more profitable than others. What this subset looks like depends on the availability of supplies, but in general the computers with the memory and/or hard-drive upgrade carry less profit. Table 3 shows the potential for profit for each type of computer, assuming average reserve price, and factoring in the number of cycles needed. In the table, *Max num* is the maximum number of computers of that type that can be built in one day, and *profit* is the potential for profit, calculated by $Max\ num * Base\ Price$. Only half of the computer types are shown, because

SKU	Cycles	Max num	Base price	Profit
1	4	500	1650	825,000
2	5	400	1750	700,000
3	5	400	1750	700,000
4	6	333	1850	616,000
5	5	400	2150	860,000
6	6	333	2250	749,250
7	6	333	2250	749,250
8	7	285	2350	669,750

Table 3: The potential for profit for each type of computer, assuming average reserve price, and factoring in the number of assembly cycles of one day. Note that only half of available PCs are shown, because the data is the same, other than for SKU-numbers and components required (IMD brand CPUs instead of Pintel).

the numbers are the same for the other half. The only difference is in the SKU-number, add eight to the ones shown, and requiring the IMD CPUs instead of Pintel.

The computers with the most profit potential are types 5 and 13, so all agents would try to capture the market for these two types. Whoever is left with a less desirable subset is perhaps less likely to stick to it. Such an agent would probably find it more profitable to run an optimizing strategy and “steal” some orders from each of the other agents; picking orders where the supplies are easy to get.

The strategic decision is analogous to the prisoners dilemma. The group as a whole is likely to do better if everyone cooperates instead of competing, but an individual agent will do better if everyone cooperates except themselves. This, along with the difficulty of punishing an agent that tries to take over “our” market share and the fact that the customer does not have any manufacturer allegiance, makes generalization the strategy of choice. However, this also provides a setting in which price wars may be common[5].

We also ran a simulation with a genetic algorithm to determine which strategy was favorable. The algorithm was run with a population size of 50 over 20 generations. Each

Number of PCs	1st Generation		10th Generation		20th Generation	
	Num	Score	Num	Score	Num	Score
1 - 4	12	6313	6	1546	2	650
5 - 8	22	16482	25	2713	27	1239
9 - 12	15	25274	18	3940	21	1784
13 - 16	1	32070	1	5592	0	-

Table 4: Results of genetic algorithm to determine how many types of PCs to produce.

individual contained 16 bits, indicating which PCs it was allowed to bid on, and produce. The consumer and suppliers were scaled up to support this large number of agents. All the individuals were created randomly from the beginning. The agents were extremely simple, bidding a random permutation of the base price, without any learning or adaptation. Also, they did not bid on RFQs if previous RFQs already bid on amounted to 2000 cycles. Since the agent were completely randomized, the set of computers some of them produced might have been completely illogical. This together with the lack of learning may have lead to a very artificial setting that is somewhat unrealistic.

In Table 4 *Number of PCs* is the maximum types of PCs an individual is producing. The *num* and *score* fields are the number of individuals in the group and the average score of the group respectively.

This experiment still showed a clear increase in revenue when producing more types of computers. However, it also clearly shows that as more agents produce more types of computers, the revenue earned by one single agent goes down. This exemplifies the prisoners dilemma. The lack of agents producing 13 - 16 different computers may be explained by the artificial setting. Because we could not count of the other agents cooperating, we decided to make PackaTAC a generalizing agent.

3 Supplies

One of the more difficult problems in the game is to make sure that supplies are available at the time they are needed for production. This is a hard problem because the suppliers make promises based on a nominal production level of 500 units per day. However, the actual production level is determined by a random walk. Thus, when the supplier produces less than what it planned, the agents have to wait longer for the supplies. The server generates a market report every 20 days that provides information about the different components, such as the amounts produced and the individual prices. However it does not specify whether each supplier is under- or over-producing. In order to make better ordering decisions it is necessary to develop a model of the price and delivery time for each of the suppliers.

The process of ordering supplies is outlined in Figure 2. First, the agent sends a request for some quantity of a specific product to one of the suppliers producing this product. On the next day the supplier responds with one or two offers. This includes one of a *partial* order on the requested day, and an *earliest complete* offer, or both. The partial offer specifies how much the supplier can deliver on the specified due date, while the earliest complete is when the supplier can deliver the whole amount. Note, that the supplier may offer to deliver after the game is over. Both the offers are valid for one day. The agent decides which offer(s) to accept, if any, and respond with with an acceptance message for the chosen offer(s). There are a number of decisions to make when ordering supplies, including which supplier to order from, how much to order, and on which day to request delivery. Also, which offer(s) to accept.

One important factor is the price of supplies. It ranges between $[0.5 * basePrice; 1 *$

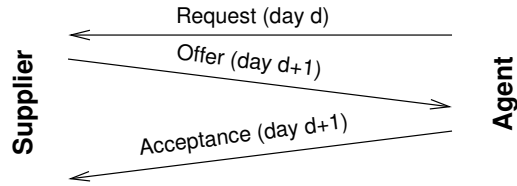


Figure 2: Schematic of negotiation for supplies. This process spans two days.

$basePrice]$ with occasional higher spikes. The price depends on demand, where high demand leads to high prices. On the first day, because there is no previous demand, the price is $0.5 * basePrice^1$.

3.1 Processing by the supplier

The sequence in which the orders are processed by the suppliers is based on a *order ratio* for each supplier.

$$Order\ Ratio = \frac{Quantity\ Purchased}{Quantity\ Requested}$$

Quantity purchased is the total quantity the agent has purchased from the supplier. *Quantity requested* is the sum of quantities in all the requests issued by the agent. This ratio is one at the beginning of the game. At the end of each day the supplier processes the requests, in a weighted random order, where the order ratio is the weight. This discourages agents from requesting large quantities without intent to order, just to block other agents. The supplier will consider the first order each agent before it processes the second order from any agent. This continues until all orders have been processed.

¹Because this is the lowest the prices are ever going to be, some strategies included ordering all the supplies needed for the whole game on the first day.

3.2 Bid-order Strategy

One strategy was to buy most or all of the supplies for the whole game on the first day. The benefit of doing this is that the price of supplies is never lower than what it is on the first day, so the agent keeps costs down. In some games, the price stayed considerably higher than the first day low, and by using this strategy an agent would not have to buy much at the high price. This is similar to the strategy in the TAC Classic game where agents bought flight tickets early to get low prices [2][3]. As a side effect of this, other agents would be “locked out” from the suppliers while these big orders were filled.

However, a downside to this strategy is that in games with very low demand the agents did not sell as many PCs as they planned. Frequently in these games, agents with this strategy incurred large negative scores, because they either ended up with large quantities of unused inventory or competed away profits. Thus, it is a high risk strategy.

3.3 Small-order Strategy

Another approach is to order very little or nothing on the first day. Instead, an agent can attempt to hold small quantities in inventory throughout the game, and reorder when needed. Such an agent is likely to pay more for the supplies, but has great potential to adapt to changes in customer demand. Also, it might be hard for such an agent to obtain supplies when they are needed. If the agent needs supplies, but the suppliers are busy producing the large orders for the big-order agents, it may find itself without some supplies for an extended period of time, and unable to make a profit. If the agent also has won orders for PCs that now cannot be produced, it has to pay penalties on top of the lack of profit. Thus,

there is a trade off between adaptability to demand and dependence on the suppliers on the one hand, and price paid for supplies on the other.

3.4 PackaTAC

We observed agents that played the high risk strategy, and saw that they had big problems in low demand games, and ended up with very bad scores. Because of this, we decided to not use the big-order strategy for PackaTAC. Instead, a variation of the low risk strategy was used. On the first day PackaTAC would order a *pre-order quantity* for each component up to six times, with requested delivery dates spread throughout the first half of the game. In the finals, the pre-order quantity was set to 1,800. In contrast, some of the agents that played the high risk strategy ordered more than 30,000 units of each component on the first day.

After this initial ordering, the agent tried to maintain a *target* inventory level. This was set to 1500, which, at the least, would last for nine days of production. Whenever the inventory for any supply went below the target level PackaTAC would reorder that supply. However, to prevent PackaTAC from over ordering in the case where it was low on supplies, but the suppliers were busy with other orders, there was a *back order limit*. This is the maximum amount that is allowed to be in orders not yet delivered. This prevented the agent from ordering too much when supplies were not available.

Also, as part of our conservative approach, PackaTAC did not count on promised deliveries when bidding on orders. If an order could not be produced with current inventory levels, the agent would not bid on it, even if the missing supplies were supposed to arrive the next day.

All the requested delivery dates for the initial large orders were in the first half of the game, however it was evident during the final rounds that this date frequently got pushed back by the supplier until the last few days of the games. When this occurred, the agent performed very poorly since it was often out of one type of component, and thus did not have the capability to build any computers. Due to a design flaw, it also frequently failed to issue smaller orders in the mean time. The back order limit was designed to take into account these large initial orders so they would not fill up the limit. However, the logic did not properly handle delivery dates that strayed from the original plan. If most of the initial orders had a due date that differed from the plan they would fill up the back order limit, and the agent would not order more supplies even if it was out. As the game carried on, the requests for supplies went down, and other agents performed extremely well while PackaTAC was stuck waiting. There was a quick attempt to alleviate this problem right before the final round, but it did not seem to have any significant impact on the performance of the agent.

3.5 Supplier Modeling

During the development phases, we attempted to model the suppliers by keeping track of which due dates were offered versus the date an order was actually shipped. The agent would also look at the market report issued every 20 days and make any required adjustments. In our tests, the model performed poorly. The six agents in the simulation had a completely different view of how the suppliers were doing, and yet none of the models were even remotely close to the suppliers true behavior. Some adjustments were made, but they did not have any noteworthy effect on the result. Because of the poor outcome, the

modeling technique was not included in the final agent.

In order to get a better picture of the suppliers, the agent would most likely have to *probe* them almost every day, by sending repeated orders. Every agent is allowed to send 10 requests to each supplier per day. It seems that most of these allowed requests should be used in order to inform a good model.

By sending all 10 requests to the suppliers every day and keeping track of the promised delivery date and the actual delivery date, an agent can keep track of how much orders get delayed, if at all. This may inform the agent about how far the suppliers are producing under nominal capacity. By issuing requests with due dates earlier than when previous orders are supposed to be delivered an agent may see if a supplier is producing over nominal capacity. If such a request does get accepted, the supplier must be over producing.

For example, the designers of the agent *deepmaize* said at the conference that their agent had a fairly good model of the suppliers. When looking at the game data afterward it turned out they used all 10 of their allowed requests daily for every supplier.

3.6 Accepting Offers

One part of ordering supplies is to decide which counter offer to accept when the supplier responds with more than one. If the supplier cannot complete the order on the requested due date, it may respond with two different offers, each relaxing one parameter: quantity or due date. The first offer will specify the amount available on the requested due date, and the latter will specify when the whole request would be available. It is a difficult decision to select which offer to accept.

Consider an agent with two orders for PCs, *A* and *B*, each for five units, having part

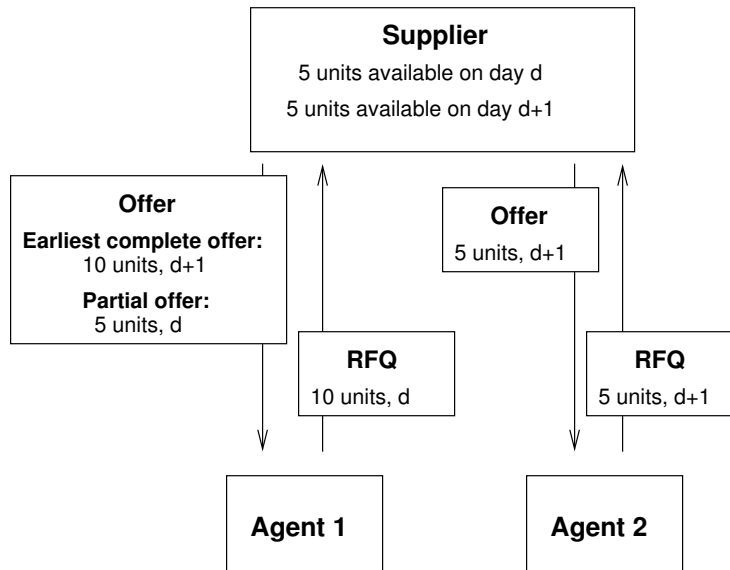


Figure 3: An example of two agents competing for components from one supplier.

x in common. Let order A be due on day $d + 1$ and order B be due on $d + 2$. Then order A need to be build on day d , and B built on day $d + 1$. Furthermore let the supplier of part x have five units free on each of day d and $d + 1$. The agent now has two options. Either it can issue one order for 10 units, or it can issue two orders, each for five units, with different due dates.

If the agent orders all 10 units for day d , as illustrated in Figure 3, the supplier will respond with two offers, one to deliver five units for day d , and one to deliver 10 units for day $d + 1$. If the agent chooses the second offer, order A will be late. If it chooses the former, the agent does not receive enough supplies to complete both the orders. The agent will have to wait a day to reorder the other 5 needed units, during which time another agent may be evaluated by the supplier and take the left-over capacity. This would result in order B being late.

Alternatively the agent can issue two orders, each for five units. The first order for five units due on day d , and the second also for five units but for day $d + 1$. The agent has the possibility of getting all the supplies and completing both orders. However, because the supplier processes the first request from all other agents before processing the second, it is possible that other agents will submit an order that will be processed between the two orders from this original agent, and take some of the supplies. Thus, there is some risk in submitting multiple offers. The suppliers also keep track of how much an agent requests versus how much it accepts. If it accepts a lot less than it requests it is less likely that its requests will be picked first on future days. In essence, establishing a bad reputation. This game feature is to punish agents to keep them from requesting large quantities, without the intent to order, just to block other agents.

The strategy PackaTAC used was to pick the partial offer if the inventory levels were low, implying the supplies were needed sooner. If the inventory levels were higher, the agent could afford to wait and get the complete order. PackaTAC also always accepted the full offer for the large initial bids unless they were too close to the end. It never accepted the partial offer for these orders. The agent also made sure not to accept offers too close to the end to make sure there was time left to get rid of the supplies.

4 Scheduling

One of the main challenges an agent faces in the TAC game is scheduling. The scheduling problem is how to allocate supply resources and factory time — what to produce, and when to produce it. As a baseline, a simple greedy algorithm was developed. This algorithm schedules production earliest deadline first, with random tie-breaking. This resulted in a reference point for the other algorithms.

4.1 Simple Heuristic

Early in the design we tested a simple heuristic algorithm. It schedules based on priority.

$$Priority = (Price - Cost) + Penalty + (12 - Due Date) * S$$

Price is the price the agent bid and the customer agreed on.

Cost is the average cost so far in the game for the supplies needed to produce this order.

Penalty is the penalty incurred, if any, if the order is not produced today, and would therefore be late.

$12 - Due Date$ is used to give higher priorities to order that are due soon, so they don't become late.

S is a scaling factor. This is needed since *Profit* and *Penalty* are dollar amounts that may be as high as 3380–125% of the most expensive computer for the reserve

price, and an additional 15% for penalty, whereas $12 - DueDate$ is in the interval $[0; 9]$ and may become insignificant without scaling.

By using genetic algorithms with a random set of supplies and orders the value of the scaling factor was determined to be roughly between 100 and 700. The result was very inconclusive within that range.

The result of this scheduler was not very favorable. It was regularly about 20% better than the greedy scheduler, but just as frequently it performed almost 20% worse. Furthermore, all values in the determined range of S performed equally bad over a random set of resources. How well the heuristic algorithm performed seemed less dependent on the value of S and more dependent on the set of resources.

4.2 Integer Programming

In an effort to improve the algorithm, we implemented an algorithm based on integer programming. *Lp_solve*² was used to solve the programme. The constraints are

$$\max \sum_i x_i V_i$$

$$\sum_i x_i L_i \leq 2000$$

$$\sum_{i,j} x_i y_j Q_i \leq S_j$$

$$x_i, y_{i,j} = \{0; 1\}$$

²ftp://ftp.es.ele.tue.nl/pub/lp_solve

where x_i indicates whether the order is to be produced, V_i is the value of producing the order, which is $Price + Penalty$ if the order will be late if is not produced today, and $Price$ otherwise. L_i is the number of cycles needed to complete the order, Q_j is the quantity of the order, $y_{i,j}$ indicates if the order uses part j , and S_j is the quantity in inventory for part j .

The improvement of using the integer programme were up to 40% better than the greedy scheduler. Unfortunately, it takes too long to run. Each day in the game is only about 14 seconds long, and the straightforward integer programme frequently takes about 10 minutes to solve!

Pruning

In order to make the integer programme solvable in the few seconds a day lasts, some solutions need to be pruned. One pruning method is to consider only orders due in the next n days. Then vary n so the resulting programme is solvable within the time limit, and does not degrade the result too much. Repeated testing showed that the cut off point was around $n = 3$, Figure 4. Above that the programme frequently would not be solved in time. However, when $n = 3$, the performance is then about the same, or lower, than the greedy algorithm. The result is shown in Figure 5.

4.3 Better Greedy Algorithm

One of the problems that turned up in the qualifying rounds was that once the agent got a sufficient number of orders that were late, it would fall into a domino effect, and produce all future orders late. This happened when there were enough late orders to use up most of

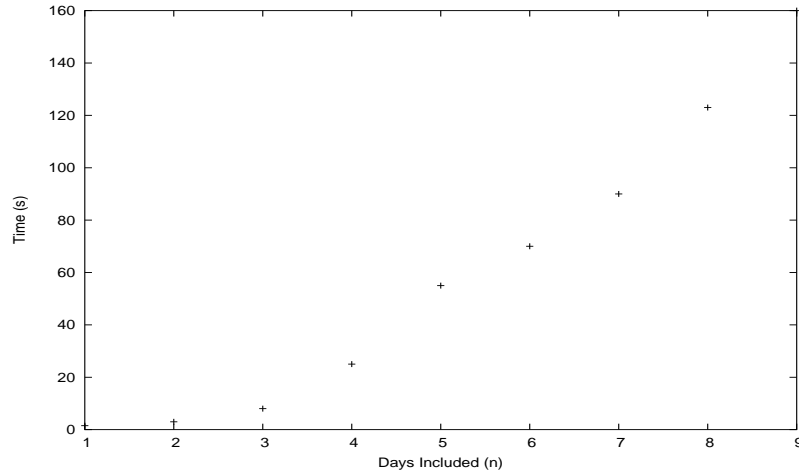


Figure 4: Time consumption of integer programme for different values of n .

the factory capacity for a day or more. By the time the next set of orders were produced they were late enough to cause the next orders also to be late and so on. The following illustrates the point

1. Let A, B, C, D denote sets of orders where each set takes one day worth of factory cycles to produce.
2. Let set A be shipped on day d , and B, C, D be shipped on day $d+1, d+2, d+3$, respectively.
3. Assume that all factory capacity up until day d is fully used.
4. A set of orders can be shipped, at the earliest, the day after it is produced.

Under these assumptions, if on day d set A is produced it will be delivered late. Also, now it is impossible to produce B and deliver it on time. If on day $d+1, B$ is produced, C

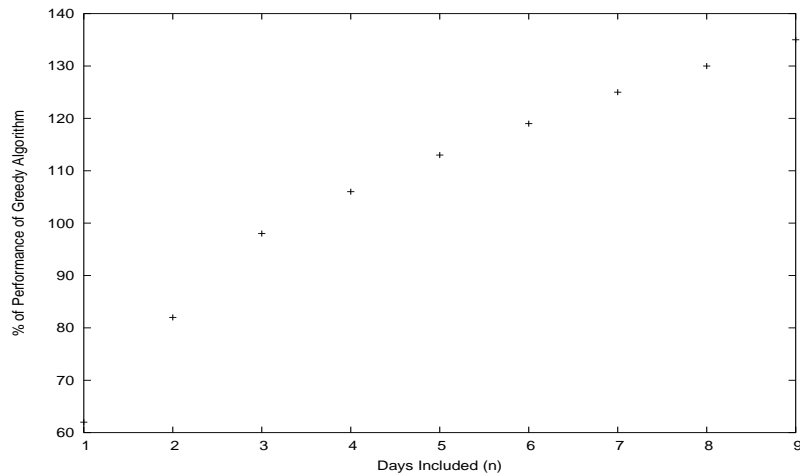


Figure 5: Improvement of integer programme over greedy algorithm for different values of n . For each n the result is the average of 100 simulations

and D will be late and incur penalties.

If, instead of the above actions, set A was not produced, B could have been produced on day d and shipped on time. Further, on day $d+1$, C would have been produced and later shipped on time. Continuing this way future orders can be produced and shipped on time.

In the first strategy in the example future orders would always be late, resulting in multiple penalties. The second strategy would incur the full five day penalty for set A , but future orders could be completed in time, and, over time, often results in lower total penalties.

This feature was never explicitly encoded in the integer programme, although it may explain why short horizons did not do well. Since the integer programme scheduled only for the current day and never looked at schedules for future days, it is unlikely that this

would be reflected in the result.

We enhanced the greedy algorithm to favor on-time orders over late ones. This, along with the fact that the algorithm can easily run in about 2 seconds, resulted in the greedy algorithm being the scheduler of choice. Also, because the agent plays a conservative strategy that is designed to avoid bidding on more orders than it can handle, the agent rarely needs to choose which orders to produce on any given day — generally, if an order needs to be scheduled for production today, it can be. The greedy algorithm is as follow:

1. Sort all orders on their due date, setting aside orders that are late.
2. For orders with the same due date, sort on profit and current inventory.
3. Until the schedule is full, schedule orders that need to be produced in two days.
4. If there is extra capacity, schedule orders that are already late.
5. If still extra capacity, schedule orders due after $d + 2$.

This algorithm biases the agent in such a way that it builds today's orders, then yesterday's orders, and then future orders. Note that this factory schedule is for tomorrow.

4.4 Shipping

There is no limit on how many computers can be shipped on one day. This makes it fairly easy to decide which orders to ship. Since there is no reward for shipping an order early and no costs associated with storage, the agent might as well wait to deliver a completed order until its due date. That way, if it is possible to use the completed units for another

earlier order and still be able to build more to make up for the loss, the agent will do so. Thus the strategy is to keep all completed orders in inventory until the day before their due date, and then ship as much as possible. If there are completed PCs in inventory after that, ship any overdue orders.

At the end of the game this is done slightly different. The customer requests PCs with due dates after the game is over. The agent can then choose to either ignore these extra orders, or bid on them and make sure they are produced and shipped before the last day, day 219. Because these orders have to be produced early, and some agents decided to ignore them, there was less competition for these last orders, which led to higher profit margins in the end game.

5 Bidding

Bidding is an important part of the game. An agent that does not have a good bidding strategy is likely to find itself either not winning as many orders as it can handle, or bidding lower than necessary, and maybe also winning too many orders. All of these scenarios will lead to a lower score than if the agent won just the number of orders it can handle.

5.1 Bidding Simulation

Early in the design we used a game simulator to study issues related to bidding. This included how many orders to bid on as well as how much the penalties would hurt the score. Both of these issues were researched in a game simulator. The simulator was designed to capture general patterns of the revenue. It was not designed to capture the competitive aspect of a game. The observations were based on 1000 simulations with a set of 200 randomly generated orders.

We assumed a constant profit margin, which is not necessarily true during the competition, but probably had little effect on the shape of the results. The revenue was measured as the score by the simulator. In the simulation, penalties were subtracted from the score if the agent got orders for more than 2000 cycles.

Number of Orders

There are 2000 assembly cycles per day, the average computer takes 5.5 cycles to assemble, and the average order is for 10 computers. This suggest that, on average, an agent can complete about 36 orders per day, if no preference is made with respect to order size or

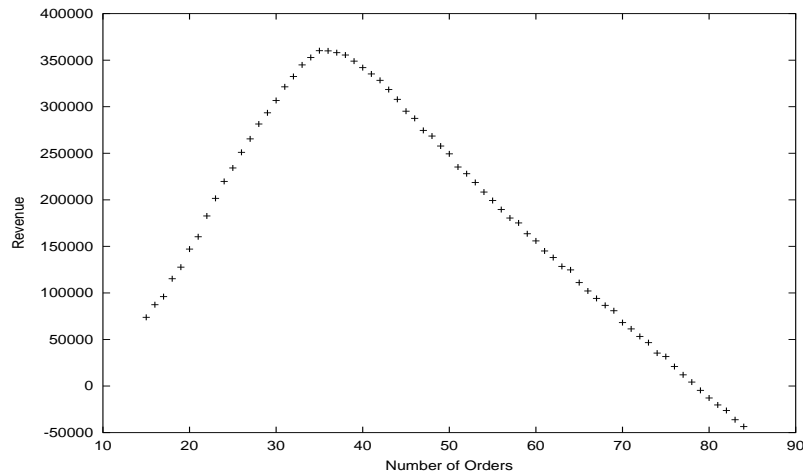


Figure 6: Revenue as a function of number of orders won on a single day.

type.

Figure 6 shows a graph of *expected revenue – penalty* as a function of number of orders won. The number of orders was varied between 15 and 85, with one step increments. This shows that targeting the average number of orders does in fact give a good result. The optimal number of orders in the simulation was 35. The analysis was intended to inform the probability bidding described in Section 5.2.

Penalty Relevance

One of the questions with regard to bidding was whether or not it would be better to attempt to get orders totaling slightly more than 2000 cycles per day in order to keep the factory full. The downside to such an approach is if too many orders are won and the penalties add up to be too significant to ignore.

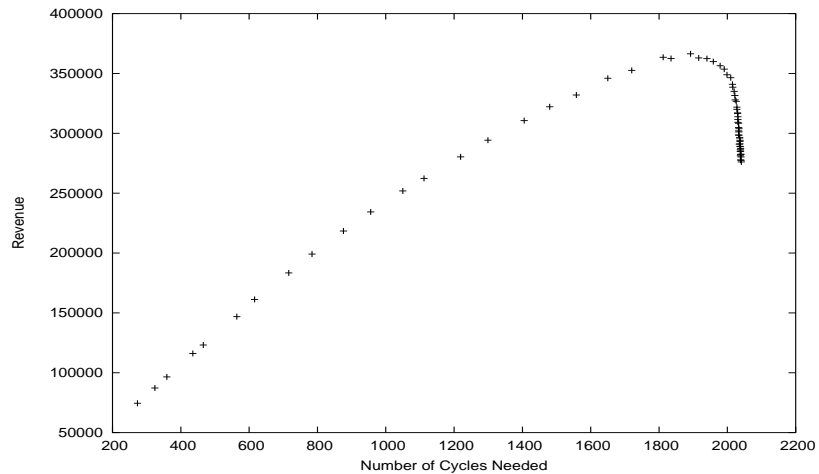


Figure 7: Revenue as a function of cycles needed to complete all orders won on one day.

Figure 7 curve shows a steep downward slope once the orders need more than the 2000 available cycles per day. This implies that overshooting by even a small amount had a significant negative effect on our profit. We designed the agent with a conservative strategy in order to stay out of this situation.

5.2 Bidding Models

We considered two different approaches to bidding on customer orders: *targeting expected capacity* and *profit margin*. *Targeting expected capacity* bids on all orders, but chooses a bid price in such a way that expected quantity won equals the factory capacity. *Profit margin* assumes all bids will win, and base the bid value on the price reports stating the actual prices for the previous day and a perceived profit margin as well as our cost of

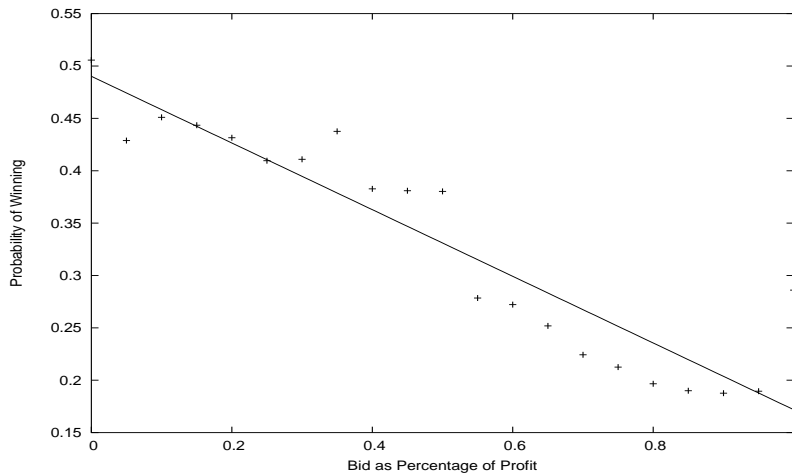


Figure 8: Average curve over 40 games showing the probability of winning as a function of profit, including a linear approximation.

supplies.

Targeting Expected Capacity

This method is based on the assumption that we have a function that maps a possible bid value to a probability, $f : b \rightarrow pr$, that indicates how likely the bid is to win. By using this, the agent will bid on more orders than it has capacity for, but in such a way that it expects to win orders totaling the unused capacity. The mapping function is computed by counting all the bids within certain ranges (buckets) and comparing them to how many of those bids were winning bids. This generates a curve expressing what the probability that a bid in that bucket will win.

Based on these probabilities the agent bids on a combination of Requests For Quotes

(RFQs) for PCs, that will, on average, yield full factory utilization. Naturally the performance of this strategy depends on the correctness of the probability curves. The goal was to start with an average curve and then periodically evaluate the state of the game and fit it to one of many predetermined curves. This curve would then be used until the next evaluation.

The curves would be pre-computed based on history, and only for games in which PackaTAC was playing. By calculating the probabilities from a sufficient number of games, including all agents, the agent would have information describing how its opponents play in different game scenarios. It would be desirable to have data describing these probabilities over the course of the games. Thus it would know how the opponents react and could take appropriate action in time to ensure its well being, and possibly “sabotage” the opponents actions.

This method was used in the seeding rounds, and turned out not to be very effective. First of all, we did not have enough time to implement it completely. The implemented version had a hard-coded curve that did not change over the course of the game. This curve was the average curve of about 40 selected games. It turned out not to be a representative curve in almost any of the games the agent played.

There were also disadvantages to this strategy related to the fact that this was the first time the competition was being held. Due to constant development and modifications by all contestants, performance in previous rounds did not predict how agents would perform in future rounds. Agents showed drastic differences in strategy and performance between rounds. Also, there was not much historical data available since the game is new.

Profit Margin

Our second bidding strategy is based on the assumption that all bids will win. The agent kept track of its available capacity for each day via the planning horizon. The available capacity on a day limited the number of orders bid on. No all bids will win, but because the planning horizon was far enough in the future, each day would be considered many times, and eventually fill up.

The planning horizon is far enough into the future that each day is considered 10 times. How quickly the free capacity for a day fills up is an indication of if we are getting as much profit as we can. We want the free capacity for a day to just be filled up by the last day it is considered. If it is not filled up in the end, factory capacity is wasted, if it fills up too quickly, we are not getting as much profit as we can. The *profit margin* is adjusted to make sure we win orders at just the right rate. The profit margin is the only adaptive factor part of the agent.

As part of the conservative strategy, the supplies and capacity needed for any order on which the agent bid were removed from consideration so the agent does not win more orders than it can build. If an order was not won, the supplies and capacity reserved for that order could be put toward another order the next day. There is also a *ranking function* that will rank the orders to indicate in which order the agent will bid on them. Orders using parts with high inventory levels will be ranked higher.

The price for an order was calculated with upper and lower bounds as follows:

$$price = 5 + profit\ margin * (reserve\ price - our\ cost) *$$

$$(num\ orders - 80)/240$$

$$upper\ bound = reserve\ price - 1$$

$$soft\ lower\ bound = previous\ day\ prices + random(30) - 10$$

$$hard\ lower\ bound = our\ cost$$

where

reserve price is the reserve price for the order as set by the customer.

our cost is the most recent price the agent paid for the supplies needed for this type of computer.

num orders is the number of RFQs on the current day, which is used to indicate if it is currently a high or low demand situation. In high demand situations the agent can expect to be able to extract more profit.

previous day prices is the calculated previous day price for this type (see below)

The soft lower bound has an upward drifting bias to make sure that if PackaTAC had placed the lowest bid on the previous day, prices would drift upwards. Also, the agent should not bid below this bound unless there are specific reasons, such as when it has more than the tolerated amount of some supplies, or at the end of the game when it needs to unload as much as possible to not be stuck with unused inventory.

The term *previous day prices* was calculated based on the high and low prices for each

type of computer the previous day.

$$\textit{previous day price} = \textit{low} * (1 - \textit{demand}) + \textit{high} * \textit{demand}$$

$$\textit{demand} = (\textit{num computers} - 800) / 1600$$

where

low is the lowest price for the previous day, as reported by the server.

high is the highest price, as reported by the server.

num computers is total number of computers ordered

This function was based on the assumption that if there is high demand there is less competition for the orders, and the agent can squeeze more profit out of the orders. On the other hand, if there is low demand it is necessary to bid lower to win orders.

This approach was used toward the end of the seeding rounds as well as the final rounds, and seemed to work reasonably well. Afterward, one issue was however found. The agent was designed to unload inventory at a really low price at the end of a game. The bid price was allowed to be below both the soft and the hard lower bounds since any cost is sunk cost at this time. However, this low-balling code had a bug that caused the prices to be too low, sometimes even negative. The unloading technique worked quite well anyway, and the agent usually made quite a bit of money during the last few days.

This strategy can be characterized as a *following* strategy. PackaTAC did not try to undercut other agents. It only lowered the profit margin if it was not winning enough

orders, often due to other agents bidding lower. It did not either try to find out which orders or computers tended to be more profitable in the long run, it only considered current inventory in the bidding scheme.

6 Results

The competition was played in essentially two parts. The first part consisted of a qualifying round and two seeding rounds. The second part entailed the three final rounds played during the IJCAI conference with presentation and demonstrations of the game.

6.1 Qualifying and Seeding Rounds

In the qualifying round PackaTAC finished in 13th place out of 20. This round was used mostly for testing in the real game setting. All agents were being worked on during this round and the behavior sometimes changed radically from game to game. Thus the final score was insignificant.

In the first seeding round PackaTAC finished 19th out of 20. At this point it was possible to observe how well the agents were playing, and make inferences about their overall strategy.

In seeding round number two the agent finished 16th out of 18. During both seeding rounds serious bugs in the agent severely hampered its performance, resulting in a very low score. However, these bugs were eliminated at the end of the second seeding round, and the last few games were significantly better than the previous. Because of these last minute changes the score probably does not show the true effectiveness of the strategy.

6.2 Finals

A table with the complete game results is shown on page 57. The table shows the results from all three final rounds.

Quarter-Final

Because of the low score during the seeding rounds, PackaTAC was seeded quite low — 16th place. In the quarter-finals the top six and the bottom three agents played in one group, and the middle nine played in another. Since PackaTAC was in the first group, to move on it had to beat at least one of the best six agents along with the bottom two agents.

It turned out that the top six agents, which all played strategies very similar to the high risk strategy outlined in Section 3, ended up somewhat butting heads with each other. Also, there were some games with very low demand where these agents incurred very large negative scores. As a result, the conservative low risk strategy payed off, and PackaTAC was the best agent in the quarter final.

Semi-Final and Final

In the semi-final round the top three agents from one group met the middle three from the other group. PackaTAC did not perform as well as in the previous round, and for a while it seemed like it would only finish fourth in its group. This difference in performance from the previous round is most likely due to the mix of strategies employed by the other agents in the group, and the relative small number of games played in each round. PackaTAC can play well against some strategies, but not as well against others, and in this round PackaTAC had to play against more of these harder strategies than before. In the end PackaTAC managed to finish third, and therefore made it to the final round.

During the final round the agent played somewhat differently. The teams were allowed to make changes overnight, between the rounds, and we configured the agent to be willing

to pay a higher price for supplies than in previous rounds. This was done to try to solve the issue of not getting supplies that we observed in the rounds before. It turned out, however, that this change did more harm than good, as we did not get supplies in a more timely fashion, we only paid more.

Another possibility as to why it played differently could be due to how the other agents played. Some of the opponents may have kept track of our actions and successfully tried to sabotage the strategy. This could have been done by keeping track of when supplies were ordered in previous rounds, and then try to block orders on those days. Thus, PackaTAC would have a hard time getting the supplies needed. In the end PackaTAC managed to hold on to fifth place.

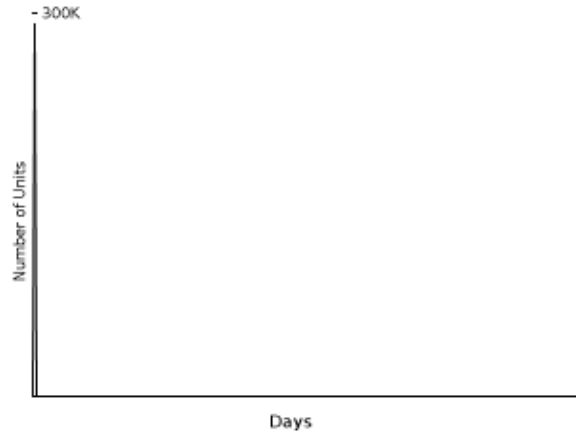


Figure 9: Typical graph showing the pattern of ordering supplies. Note that the Y-axis is not shown in order to not interfere with the graph.

7 Other Agents

Since this was the first year the competition was run, it was a building year. We expect extensive post game analysis to find out what worked well, and what did not. The following are some observations of the agents that did well in the different rounds.

7.1 HarTAC, whitebear, PSUTAC, Tac-o-matic, Socrates, and jackaroo

All these agents played extremely similar to each other. They all used the high risk strategy described in Section 3, where they ordered all the supplies the first day. Many of them ordered 250,000 units, or more, the first day. The differences in their results was mostly due to how well tuned they were — if they ordered just the right amount, or too little, from the beginning.

HarTAC somewhat kept probing the suppliers throughout the game, but ordered next to

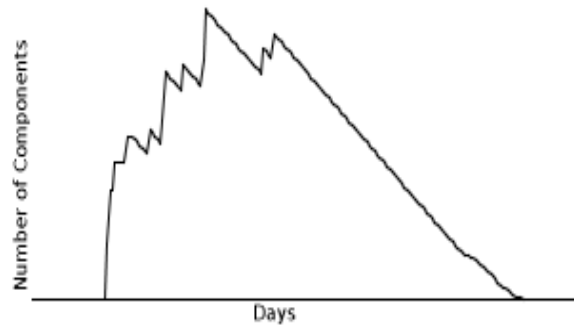


Figure 10: Curve showing the typical component holdings for this group of agents.

nothing besides the first orders. PSUTAC and jackaroo, in a couple of games, issued several other large orders besides the first ones. This probably occurred if the agents realized it was running low and needed to replenish the supplies. It only happened in one or two games, and might have been experimental code.

Whitebear frequently seemed to under-order supplies, and in many games had a tendency to run out before the end of the game. This most often resulted in hurting the agent in games with high demand where the other agents had about 30 more days of playing time. In games with low demand they often did not get as negative of a score as the other high risk agents. It was a conservative version of the high-risk strategy, and seemed to work to their advantage since whitebear was the only one of these agents to go on to the final round. It did not have as much luck in the final round, ending up in last place.

7.2 RedAgent

This agent played fairly similar to the previous group of agents, but it ordered low amounts of supplies throughout the whole game, not just the first day. Although unlike the previous

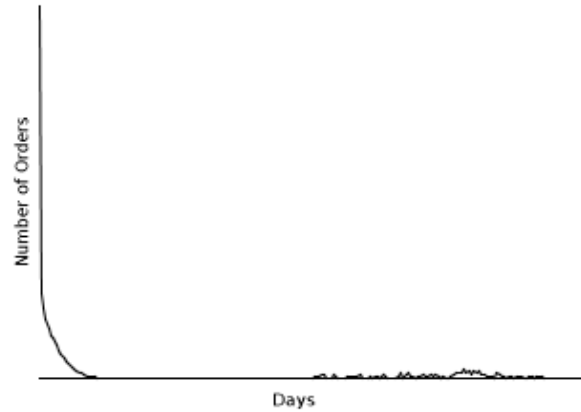


Figure 11: Typical graph showing the pattern of supply ordering for RedAgent.

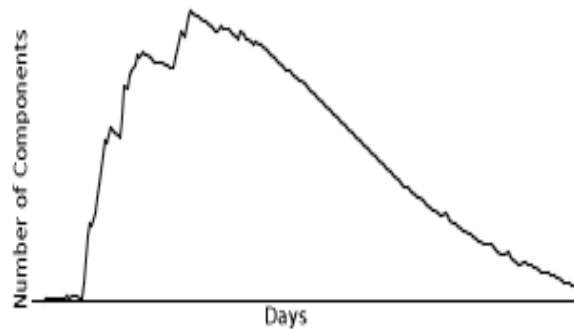


Figure 12: Curve showing component holdings for RedAgent.

group, it did have enough supplies to last the whole game. This was most likely due to the continued ordering, because it had the ability to adapt to changing demand. This agent ended up winning the competition.

7.3 deepmaize

This agent was quite interesting. The designers reported at the conference that on the first day, deepmaize issued a very large request for supplies and declining it, intending to block

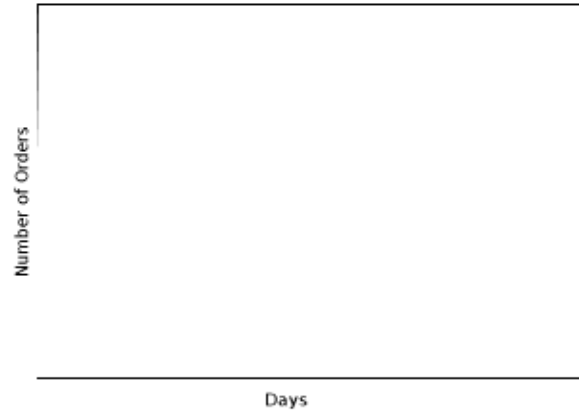


Figure 13: Graph showing the probing of the suppliers by deepmaize. This agent sent all the allowed requests every day, hence the square graph.

the big-order agents. This request was on the order of 2.2 million units. It also ordered the majority of the supplies needed the first day. Unlike other agents it kept probing the suppliers as much as possible throughout the game, as seen in Figure 13. It placed small orders during the whole game, most likely to adapt to change in demand.

Deepmaize somewhat changed its strategy for the final round. Instead of getting all the supplies right away, the orders were more spread out. This is shown in Figure 15. This agent placed second in the final.

7.4 Botticelli and TacTex

The agents, Botticelli and TacTex, played fairly similar to each other. They both seemed to employ strategies that built up their component holding levels during the first half of the game. Then during the second half they ordered very little but sold all their inventory. In the final round, TacTex was third, and Botticelli fourth.

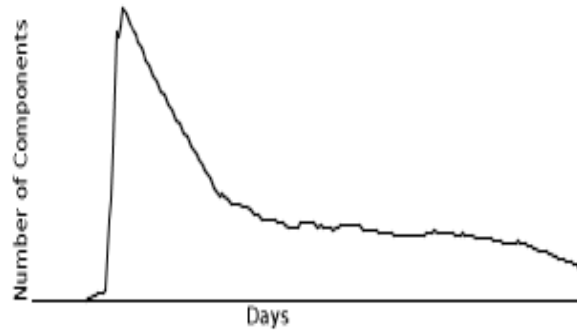


Figure 14: Graph showing strategy for component holding during the quarter and semi finals.

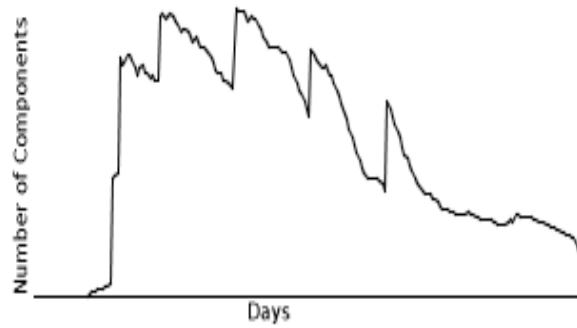


Figure 15: This shows strategy for component holding for the final round. Note the change to a saw-tooth pattern.

While the graph showing component holding levels is very similar to the one for the high risk strategy, Figure 10, these two agents did space out the first-day orders more. This is seen in Figure 16 by the prominent saw tooth pattern in first half of the graph.

Botticelli

Botticelli had moderate sized orders the first day, about 200,000 units, not nearly as large as for the high risk agents. Instead, around day 21 the agent placed new orders for almost as many supplies as the first day, close to 140,000 units. This waiting period most likely

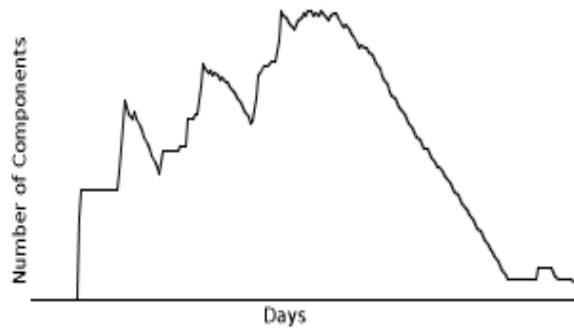


Figure 16: A typical graph for component holding for Botticelli and TacTex. Note that unlike RedAgent the peak is more at the middle of the game, with clear saw-tooth like additions.

allowed the prices to go back down after the initial peak on the first day.

TacTex

The previous graph which shows ordering for Botticelli does not represent the strategy of TacTex. Instead, TacTex had a curve somewhere between deepmaize and the high risk strategy. It relied heavily on the initial bids, but kept ordering smaller amounts throughout the game.

7.5 PackaTAC

The order graph for PackaTAC is similar to the one for RedAgent, except that it is a few magnitudes lower. The initial peak is a lot lower, about one third, while the remaining peaks are slightly higher. This gave great adaptability for low demand games, in which PackaTAC often ended up being the best agent.

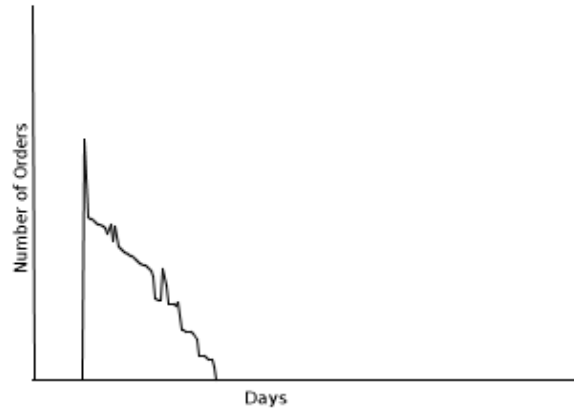


Figure 17: Curve showing the ordering strategy for Botticelli. The second peak is the reordering around day 21. The Y-axis is not shown in order to not interfere with the graph.

8 Possible Improvements

8.1 Supplies

Due to a lack of time, some things did not get implemented or looked into that could have enabled more improvement. For instance, something that would be good to know more about is the state of the suppliers. If the agent knew the level of production and how long the wait would be to get the supplies, better decisions can be made about when to order supplies, how much to order, and which supplier to order from. This knowledge would make it easier for the agent to keep enough inventory on hand to then be able to complete orders at full capacity. Right now that is an issue with the agent. It may run out of supplies, causing utilization to go down, and it does not make money.

This would help in selecting which supplier to order from. Currently a supplier is picked at random. By keeping track of supply and demand for each individual supplier, the

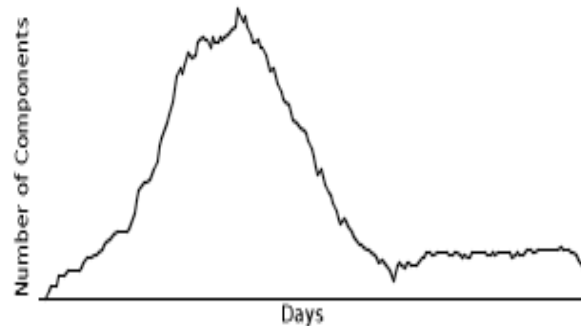


Figure 18: Curve showing the holding strategy for PackaTAC. Note how the target inventory level is maintained at the end of the game. The initial large orders were designed to last to about day 140.

agent could pick the cheaper and faster supplier. This could also lead to insights of how the other agents are doing, based on what and how much they order.

Another problem that would be good to resolve was described in Section 3.4, where the agent's initial large orders were delivered very late. One way to relieve this issue would be to set a limit on how much of a delay our agent is willing accept. Right now there is no such limit, which hurts the agent's performance. Another solution would be to probe the suppliers even when the backorder limit has been reached. If the inventory levels are too low, and the probes show that the products are available, they can be ordered. Part of this solution was implemented right before the final round. A cap on the due date was created to not accept really late offers. Unfortunately the change did not show obvious improvements. A more elaborate scheme, like using probing, described in Section 3.5, would likely have a better performance.

There were also a few bugs and design flaws that were found after the contest was over. For example, when the backorder limit is calculated, it is designed to take into account any

outstanding large initial orders. However, due to a bug they are accounted for on the due date for which they were requested, not the due date that was offered or accepted. This leads to a situation where, if there are any of these orders due after the last planned due date, the backorder limit is reached. Because of this, the agent will not order new supplies when inventory levels are low because it thinks there are plenty of orders coming in. These orders however are far off in the future. The lack of supplies on hand result in inactivity for the agent.

8.2 Ranking and Bidding

In the bidding strategy, the first step is to rank all the RFQs. The most important aspect of the current ranking method is that it drops all RFQs that cannot be completed in time due to lack of supplies or factory capacity. Something that most likely would have showed improvements is to base the ranking on other properties of the RFQs. Unfortunately there was not enough time to study this issue. This included comparing the potential for profit between the RFQs, i.e., calculating the reserve price based on the price of the supplies the agent paid, or penalties, etc.

Also, the agent does not take into consideration if particular types of PCs are in low demand or supply. The initial observation was that there was no noticeable shortage in either demand or supply and thus the issue was not of enough importance to spend considerable time on. However, there might still be less competition in the game for certain types of PCs, maybe even over certain types of orders such as large/small quantity, early/late due date, and high/low penalty, due to biases in our opponents behavior.

8.3 Profit per Product

Another idea that did not get implemented due to time was tracking profit per product. Currently the agent only kept track of profit margin for all the products combined. If this was done per product, the agent could learn to extract more profit on products where there is less competition.

This could be accomplished by comparing how many orders of each type the agent bid on, as well as how many it won. If it won all the orders it bid on, it can raise the profit margin for that product. If it does not win any of the bids, it could lower the profit.

9 Conclusion

TAC-SCM was a very interesting competition, with a lot of challenges. The design approach taken in the development of PackaTAC was to solve the problems in simple rational ways. The resulting agent was a very simple, heuristic agent. The simplicity can be seen in the computing power required by the agent. During the competition two game servers were run, requiring two instances of the agent to be running. Both instances of PackaTAC ran on a single iMac. During testing, six instances were run on the same computer, without any noticeable deficiency in performance for either the agents or the computer. Compare this to deepmaize that utilizes four computers for each instance of the agent. The result in the competition seems to validate the simplistic approach used by PackaTAC.

The agent played a very conservative strategy. It was designed to hold relatively low inventory, while at the same time not bidding on any more orders than it can fulfill with the current inventory and free factory cycles. PackaTAC had a delivery performance far better than any other agent in the final round. It also has a follower strategy, where it does not seek out the most profitable group of computers. Instead, it adapts to the general state as it changes over the course of a game. This conservative strategy seems to be reasonably robust, considering most of the agents that played the high risk strategy to the fullest extent, like PSUTAC and HarTAC, did not make it to the final round.

Because it was the first year this new game was run, there was a lot to learn just to achieve competence. Most of the work in developing PackaTAC ended up being focused on bookkeeping and bug fixing. The other teams were in the same situation, and this resulted in changing behavior and competence for the opposing agents between the different rounds.

Now all teams have a solid base to start from, the possibility of an extensive analysis of the game and the agents, and the time to rework and extend their agents. It will be very interesting to see what the different teams come up with for the competition next year.

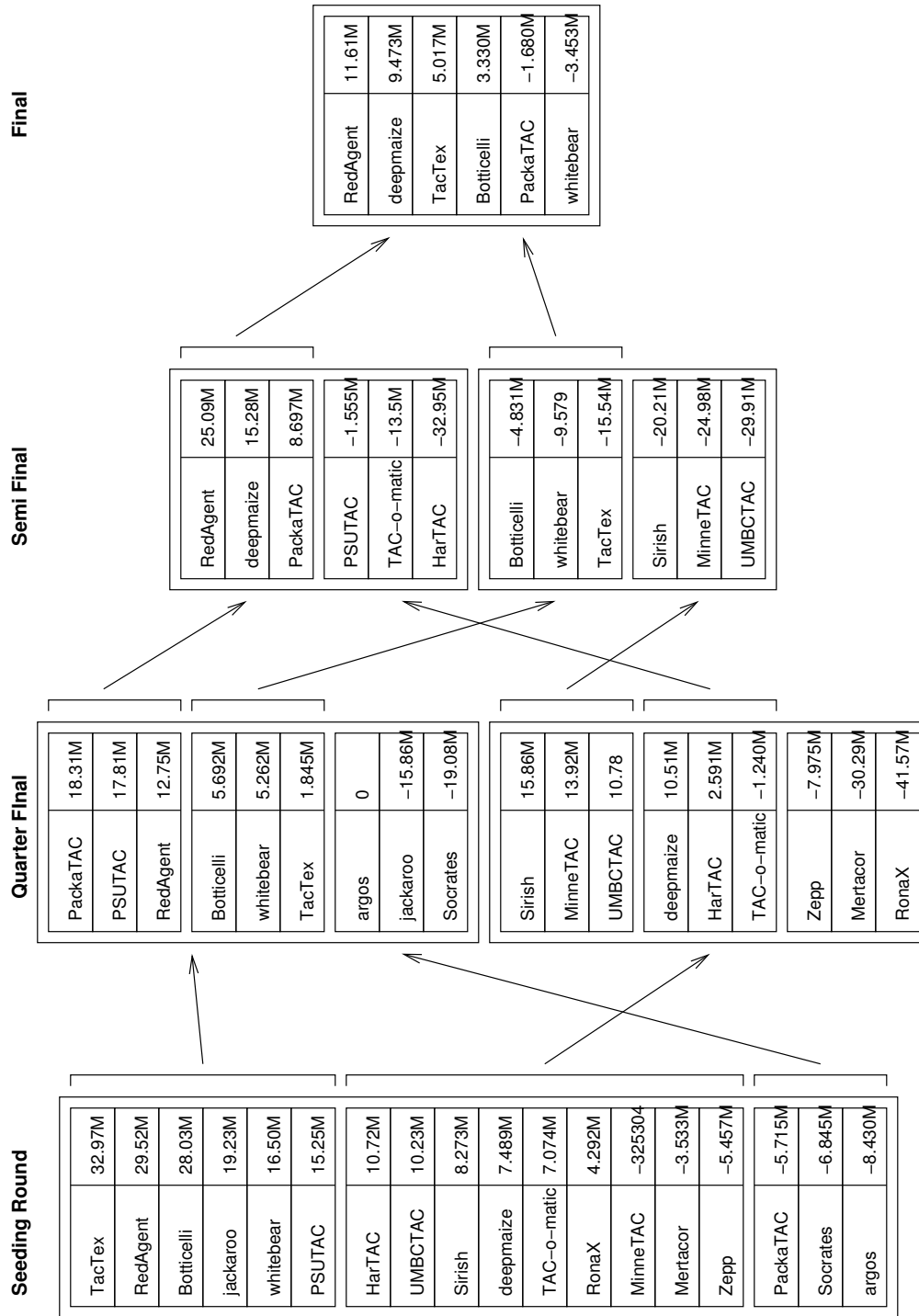


Figure 19: Results in the seeding and final rounds.

References

- [1] Raghu Arunachalam, Joakim Eriksson, Niclas Finne, Sverker Janson, and Norman Sadeh, The TAC Supply Chain Management Game, http://www.sics.se/tac/TAC03_spec.pdf, Draft version 0.62, 13 June 2003
- [2] Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, and Yevgeniy Vorobeychik, Price Prediction in a Trading Agent Competition, <http://ai.eecs.umich.edu/people/wellman/pubs/ppredict02.html>, Technical report, University of Michigan, 2002
- [3] Shih-Fen Cheng, et al., Walverine: A Walrasian Trading Agent, <http://ai.eecs.umich.edu/people/wellman/pubs/walverine02.html>, Technical report, University of Michigan, 2002
- [4] James E. Hanson, and Jeffrey O. Kephart, Spontaneous Specialization in a Free-Market Economy of Agents *Second International Conference on Autonomous Agents*, Minneapolis/St. Paul, May 1998
- [5] Amy R. Greenwald, and Jeffrey O. Kephart, Shopbots and Pricebots, *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, August 1999