GARFIELD: A Robust Supply Chain Management Agent

Bhalchandra Agashe and Vineet Gorhe
Department of Computer Sciences
The University of Texas at Austin
{bsagashe, vmgorhe}@cs.utexas.edu

7 December 2006

Abstract

We present and evaluate the design of Garfield, our Supply Chain management agent, which has been designed to work in the TAC SCM test bed. In this report, we provide a brief introduction to the SCM problem and the TAC SCM test bed and follow it up with a detailed design overview of our agent. We then evaluate our agent in several experimental settings and provide detailed analysis of the results.

1 Introduction

The TAC Supply Chain Management Scenario comprises of games in which six agents act as computer manufacturers in a simulated economy managed by the game server. The challenges in the game include

- Procuring components at economical rates and in a timely manner
- Participating and winning in the first price auction held for customer orders
- Timely production and delivery of computers.

Full details of the game specifications are available in the official specification document[1].

2 Design Overview

Fig. 1 illustrates the high level design of our agent. The functionality of the agent is divided into two parts, the supply side and the demand side. The Supply Manager handles the supply side and is responsible for getting components delivered in a timely manner at economic costs. The Demand Manager handles the demand side and is responsible for producing and delivering computers to customers.

The main objective of our agent is robustness in face of different market conditions and mix of competing agents. Our agent places equal importance to being steady in face of rapidly varying prices as well as maintaining its risk-taking ability. We try to achieve these two objectives by having three sub-strategies and shifting in favor of one depending on current market situation. The most conservative of the three strategies is the *Undercut* strategy which has maximum ratio of offer acceptance but minimum returns while the most risky is the *Overbid* strategy which bids on extra RFQs on basis of offer acceptance ratio of our agent. In between is the *Minority* strategy which tries to intelligently select RFQs which might have very few or almost no bidders and tries to guarantee acceptance at a very high bid price. In addition our agent tries to predict customer demand using a fuzzy approach to help procure the supplies at cheap prices in advance. On the supply side, our agent computes component costs by factoring the predictions as well as the transaction history with the supplier. Thus the supply manager adapts in a limited fashion to the ordering patterns of other agents over the duration of the game.

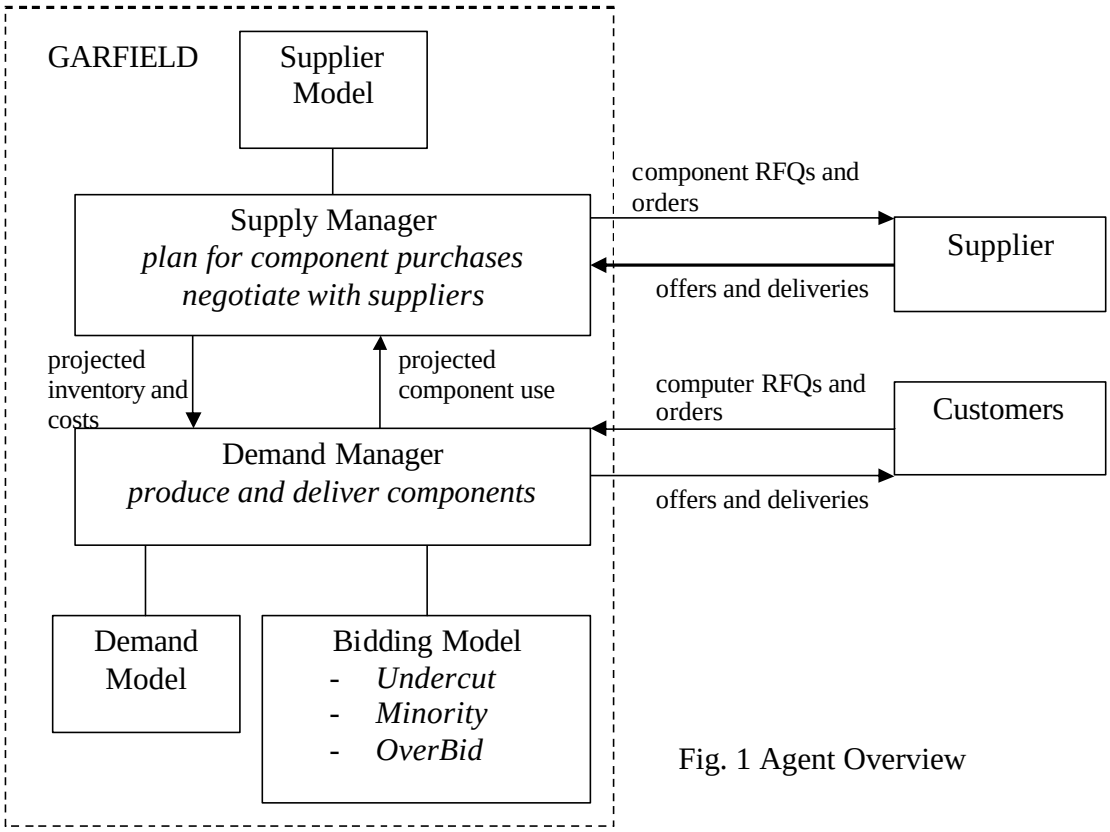

Fig. 1 Agent Overview

3 Supply Manager

3.1 Supplier Model

The supplier model is at the heart of the supply manager. We maintain a supplier model for each of the ten suppliers. Each model tries to forecast the available production capacity of the corresponding supplier.

3.1.1 Prediction

The TAC SCM framework allows 5 RFQs to be sent per component per supplier every day. These RFQs can contain the actual quantity needed or they could be zero quantity price probes. The supplier's responses to these RFQs are used to estimate the available capacity with each supplier. This information is then used to predict the prices at which the supplier would be willing to accept the complete quantity RFQ.

Our agent tries to model the suppliers up to 20 days into the future. Hence, we need to spread out the available 5 RFQs (per component, per supplier), over the period of next 20 days.

We achieve this by dividing the next 20 days into 5 intervals, viz. 3 to 5, 6 to 8, 9 to 11, 12 to 14 and 15 to 20. This ensures that we get information about the supplier's committed capacities spread over the next 20 day interval.

Given these values we interpolate to find the available capacities for all 20 days. This makes it possible to predict component prices each day, over the 20 day interval.

3.1.2 Adaptation

As the game progresses, we try to estimate the ordering patterns of other agents. The summary of the ordering patterns of other agents reflects in the component prices over the 20 day interval that we consider.

If the supplier responds to our RFQ with a reduced quantity offer but not an alternate one, then it is a clear indication that we were asking the components at too low a price, in the given interval. The difference in the quantities requested and offered is used to estimate how wrong the price prediction was. Thus at the start of each day we boost the component prices such that probability of getting the complete order increases. Since we have defined 5 intervals, we do this for each interval. This information is then incorporated into the price prediction mechanism to predict a more accurate price over each of those 5 intervals.

The motivation behind this approach is as follows. If the mix of competing agents prefers to order components up to 15 days in advance, then the supplier is likely to make partial offers during those days. Hence in order to get the complete quantity, the reserve price needs to be hiked in that interval. The supplier is likely to make complete offers for 15 to 20 days into the future. Hence, there is a possibility of getting these components at a cheaper price between days 15 and 20. Our agent explores that possibility as well.

3.2 Supply Manager Functionality

This module has the responsibility of maintaining component inventory levels so that existing orders can be manufactured and delivered on schedule. In addition, the Demand Manager needs an assurance about the component inventories, for deciding the feasibility of bidding on (considering) customer RFQs. It is the Supply Manager's task to provide this assurance. Thus, the supply manager performs the following distinct operations:

- Procures components needed for customer orders that have been won.
- Procures components in anticipation of customer orders that would be won in the near future.

On any given day, the demand manager has information of the components that would be used in the next 10 days. This includes orders that were won and are due in the next 10 days as well as RFQs that our agent has bid on which are due in the next 10 days. This information is weighted using fuzzy customer demand predictions and is passed to the supply manager. The supply manager performs linear regression on this information to arrive at the projected components usage for the next 20 days. The predictions for days 11 to 20 are likely to be incorrect since linear regression may not fit the demand correctly. Since the error in prediction tends to increase with number of days we try to reduce dependence on predictions far in the future. The projection for the day 11 is more likely to be correct than the one for day 20 hence we shade down these projections linearly from 1.0 to 0.5. In other words, we assume that the projection for day 11 is correct and that for day 20 is just half as likely to be correct.

The supply manager uses this information to make the following decisions:

How much to order?
It orders components in quantities that make sure that a threshold inventory level is always maintained.

When to order?
The components which are ordered should arrive on or before the day when inventory drops below the buffer level. The supply manager follows the policy of placing only one order in the interval. To arrive at this exact date, the supply manager does the following:

- Decides in which of the 5 predefined intervals does the day of projected inventory shortfall lies.
- Aggregates this information for the interval (i.e. finds the total shortfall amount in the interval).
- Calculates the cost of ownership of those components for every day in that interval up to the first day of shortfall.

Cost of ownership includes the component costs and the inventory holding costs. Orders are placed for the date where cost of ownership is the least.
e.g.: If the projected inventory shortfall is 8 days into the future, then the supply manager calculates the component ordering costs for days 6, 7 and 8. It also computes the storage costs for each of these orders. If the components are ordered due 6 days, then they need to be stored for 2 days before they actually fill in the shortfall. Similarly, components which are ordered due 7 days incur storage costs for 1 day. Cost of ownership is defined as the sum of these component costs and storage costs. If it turns out that the cost of ownership is the same, then we order it for the earliest date to minimize the impact of late component deliveries.

Where to order?
Each component (except the CPUs) has two suppliers. The supply manager uses the corresponding supplier models to predict the component costs and orders from the one that promises to be the cheapest source. It sends an RFQ for the entire order at the predicted price to that supplier.

3.3 Zero Quantity RFQs

If it turns out that the inventory level for a component does not drop below the threshold in a given interval then the supply manager sends a zero quantity RFQ due first day in that interval to both suppliers (just one supplier in case of CPUs).

If it turns out that the inventory level for a component does drop below the threshold in a given interval then after sending an appropriate quantity RFQ to the best supplier, the supply manager sends a zero quantity RFQ to the other supplier of that component (none in case of CPUs) due first day in the interval.

These zero quantity RFQs facilitate the supplier model to accurately predict component prices.

3.4 Specifics

The supply manager attempts to maintain an inventory threshold level of 800 for each component (except for CPUs for which it is 400) over the first 210 days. It then reduces the threshold linearly to 0 over the last 10 days, in an attempt to end up with zero inventories.

3.5 Fuzzy Customer Demand Predictor

If the customer demand can be predicted, then the requisite components can be ordered in advance. This would result in the Demand Manager bidding more on RFQs concerning computer types which are in high demand. We capture the customer demand by measuring the number of RFQs for each computer type arriving on each day. We use similar information for the past 10 days and perform linear regression to predict Customer Demand over the

next 10 days in each of the three segments. This information is used to procure more components where the predicted customer demand is more.

4 Demand Manager

The entire strategy of demand manager revolves around selection of RFQs to bid upon. We found through our experiments with other agents that selection of correct RFQs was the single-most influencing factor in profit-making ability of an agent. There are two general ways in which an agent could bid: 1) Try to bid on a majority of RFQs and track the fraction of offers getting accepted. In this case an agent can try to bid a number in inverse proportion to the average of fraction of offers that get accepted historically. So, if the average fraction is 0.5 the agent should try to bid at least double the offers it expects to receive orders for. 2) An agent could try to select RFQs using some selection criteria and bid only on those offers. We try to integrate both these approaches and come up with a unique solution to the RFQ selection problem.

It is not only important to select the right RFQs to bid upon but it is also important to bid at the correct price since that determines the probability of offers getting accepted. We use the fraction of offers accepted yesterday for a particular type of computer to determine the correct price to bid on a RFQ.

4.1 Selection of RFQs and pricing

The basic assumption our agent makes is that every other agent in the competition tries to sort the RFQs in descending order of profit expected from that RFQ (although we later demonstrate that different sorting orders of RFQs do not really affect us [Experiments 6,7,8]). Profit here is calculated as the difference between yesterday's lowest winning price for a computer type and the computer cost for that type. Our agent sorts the RFQs in descending order of profit and divides this list in three parts. We use different bidding strategies for each part. Our agent also reserves available cycles up to 10 days for today's RFQs. We divide these cycles in two parts for bidding on first two parts of the RFQ list.

a) *Undercut strategy:* In the first part, we keep bidding on RFQs in descending order of profit till we run out of cycles reserved for this part or none of the offers can be fulfilled using current component inventory. To avoid bidding on offers with very less profit margin we restrict bidding to RFQs which guarantee at least 5 percent profits (of the computer cost). We bid on all RFQs which can be fulfilled using factory production and/or computer inventory.

   We put the bid price of RFQ as yesterday's lowest price – *k* (where *k* is varied from 1 to 10 depending on fraction of offers being accepted). Thus this strategy tries to win offers by undercutting all other agents and trying to bring down the prices of computers.

b) *Minority strategy:* The second part of our bidding strategy can be viewed as a *minority* strategy where our agent tries to bid on those RFQs which have very few or no bidders (so our agent will gain the most if it is in the *minority* rather than majority). We determine such RFQs by going in reverse order of the RFQ list starting from the last one. The reasoning behind this is since these RFQs lie at the bottom of the RFQ list (and since all agents sort RFQs based on some function of profit) almost no agent will try to bid at the later part of RFQ list. To avoid bidding on RFQs with extremely low margin of profit we restrict bidding to only those RFQs which have a profit margin (here, profit is our bid price minus the computer cost) at least equal to the lowest profit offer we bid on in the first part (*Undercut strategy*).

The bid price we put on these RFQs is a function of yesterday's winning high price for this type of computer. We multiply the winning high price by a factor which is dependent on fraction of offers of this type accepted. If the fraction of offers getting accepted is very high then we calculate the bid price as a factor of the reserve price of RFQ instead of high price in hope of larger gains. This strategy tries to drive the prices of computers up by bidding high prices. For our winning offers we increase the average winning price of that computer type and hence balance our previous strategy of bringing the prices down.

c) *Overbid strategy:* Due to our restriction criteria of not bidding on offers below a certain threshold of profit in previous parts it may happen that we bid on very few offers. We try to counter-act this by bidding on extra offers. We decrease the threshold of minimum profit expected per RFQ and bid at a price similar to the *minority* strategy. The number of such extra bids depends on the fraction of offers of *minority* strategy getting accepted (we assume that most of our *undercut* offers will get accepted). We increase the extra bids if very few offers in *minority* strategy are getting accepted.

4.1.1 Interaction between Undercut and Minority strategy:

The *undercut* strategy and *minority* strategy tend to complement each other. Since, our agent tends to be the only bidder in a substantial number of RFQs in *minority* strategy we affect the lowest, average, and highest winning price of that type of computer. Generally, our bids in *minority* strategy tend to increase the current prices. Thus, some types of computers bid on in *minority* strategy will show up in the *undercut* strategy as most profitable over the next few days. And some offers being currently undercut will lie in the bottom part of RFQ list so that the *minority* strategy picks them up. Thus, our strategy aims at finding the correct balance between these two types of bidding strategies with the over-bidding strategy being used as a way to increase the offer acceptance probability of the *minority* strategy.

4.2 Adaptation

Our agent maintains fraction of offers getting accepted for the undercut and the *minority* strategy. We also store the profit made on offers getting accepted. During the course of the game our agent tries to increase/decrease the number of cycles available for each strategy based on which strategy gets the most profit per RFQ.

The bid prices for all strategies are adaptive and try to get maximum number of offers accepted by varying the prices as a function of offers getting accepted.

4.2.1 Price-war mode

Depending on the current product inventory level, our agent enters a price war mode if the number of computers per type is above a certain threshold. In the price war mode, our agent bids on only those offers which can be fulfilled directly from inventory for the *undercut* strategy. Other strategies continue working normally. Also, in price war mode no extra computers get produced out of remaining factory cycles.

4.3 RFQs by due-date

In all the three bidding strategies, our agent maintains the quantity of computers promised (through a particular RFQ) on a particular due-date. We try to cap the maximum computers that can be promised as an increasing

function of the RFQ due-date. Effectively, our agent reserves maximum quantity of computers for RFQs with farther due-dates which give our agent more time to plan production and maintain supply. Currently, our agent does not bid on 3-day RFQs (though, they tend to be most profitable) due to their inherent nature of upsetting production plans.

4.4 Extra computers

Our agent uses extra factory cycles to produce computers. If the current product inventory is above a certain threshold then all the RFQs which can be fulfilled directly through the inventory are bid upon.

5 Experimental Evaluations

To evaluate the performance of our agent in a systematic fashion, we ran a series of controlled experiments.

5.1 Experiment 1

Objective: To test the effectiveness of our Supply Manager strategy.

Description: We ran our agent against another version of our agent which uses the supply manager provided with the starter agent.

Observation:
We observed that when everything else was kept the same, Garfield's Supply Manager Strategy resulted in higher net profits across games in a statistically significant way (statistically significant at 99 percent confidence, calculated using student's t-test)

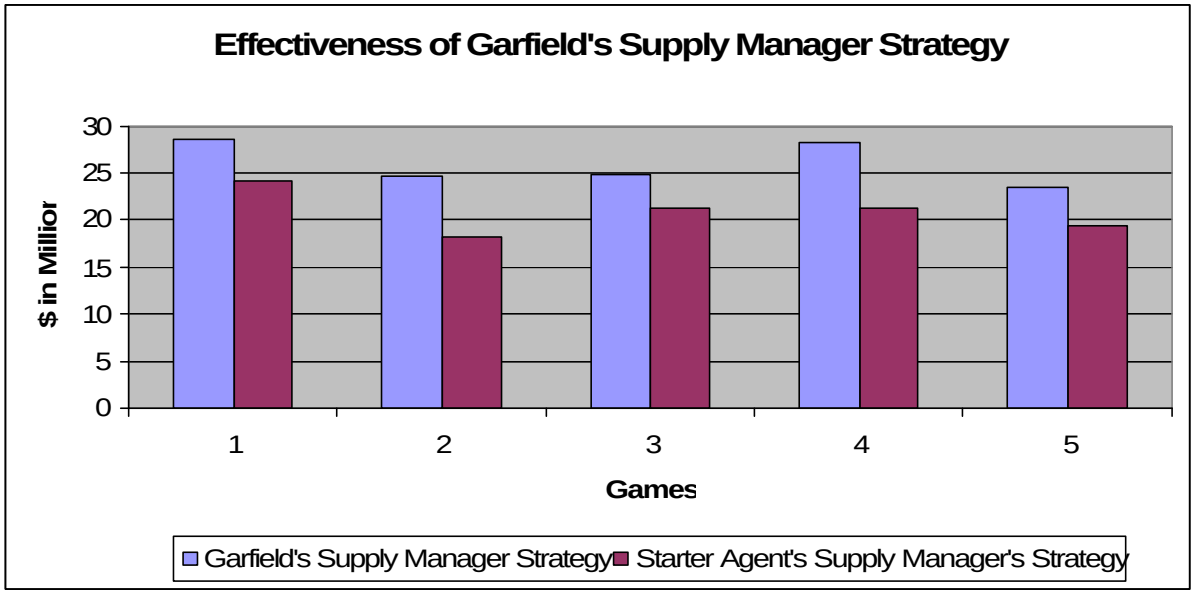

Fig. 2

5.2 Experiment 2

Objective: To test the effectiveness of our Demand Manager strategy.

Description: We ran our agent against another version of our agent which uses the demand manager provided with the starter agent.

Observation:
We observed that when everything else was kept the same, Garfield's Demand Manager Strategy resulted in higher net profits across games in a statistically significant way (statistically significant with 100 percent confidence, calculated using Students t-test).

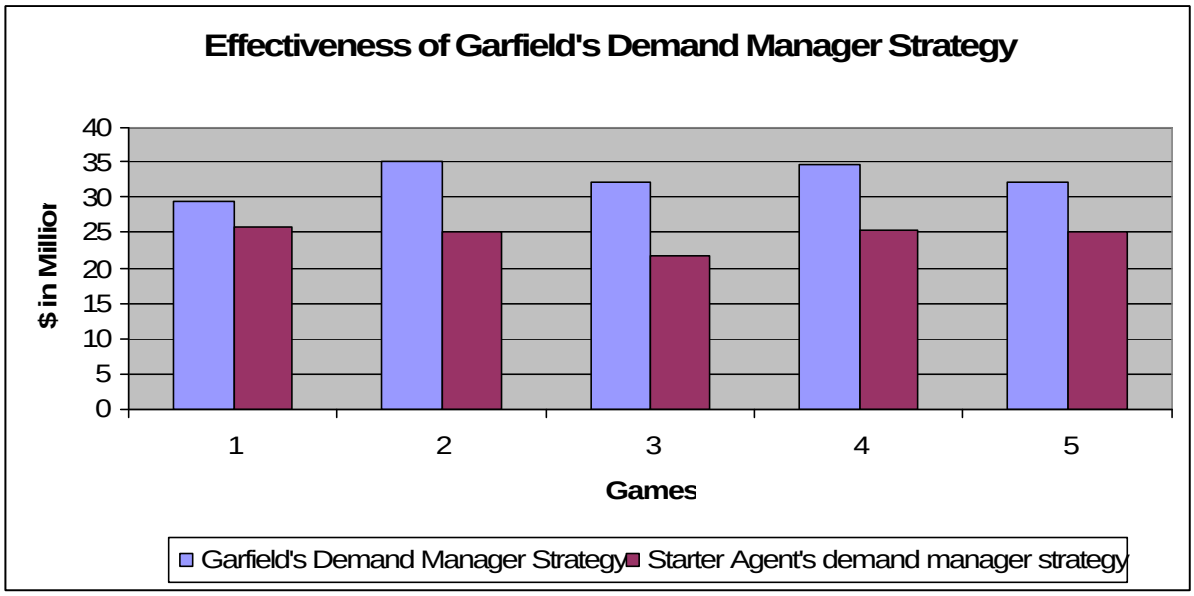

Fig. 3

5.3 Experiment 3

Objective: To test the ability of our agent to compete against its clones.

Description: We ran 6 instances of Garfield against each other in 5 games.

Observation: We observed that all instances of Garfield made a profit. This is in contrast with the case when 6 instances of the starter agent are run against each other (where they end up making huge losses). This collective behavior of the Garfield agent is due to its hybrid bidding strategy. It does not rely only on undercutting and shifts its focus to the *minority* and overbidding strategies when the profits accruing from the undercutting strategy go down. This effectively casts a wider net and each clone of Garfield ends up procuring more orders than it would have without such a hybrid strategy

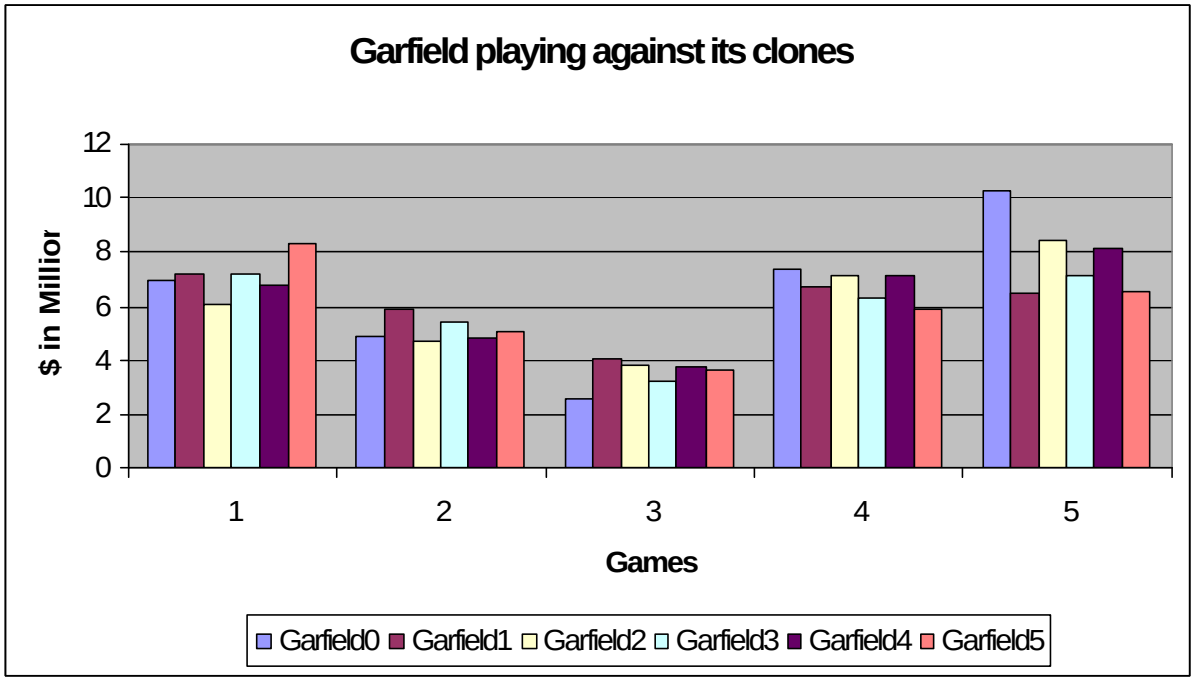

Fig. 4

.

5.4 Experiment 4

Objective: To evaluate Garfield's profit metric against agents who use average price for calculating profits

Description: Garfield calculates potential profit using last day's lowest winning price. In this experiment, we test this approach against starter agents which use last day's average winning price to calculate profit.

Observation: As can be seen from the graph, Garfield's profit metric outperforms starter agents which use average price (from last day) to calculate potential profits

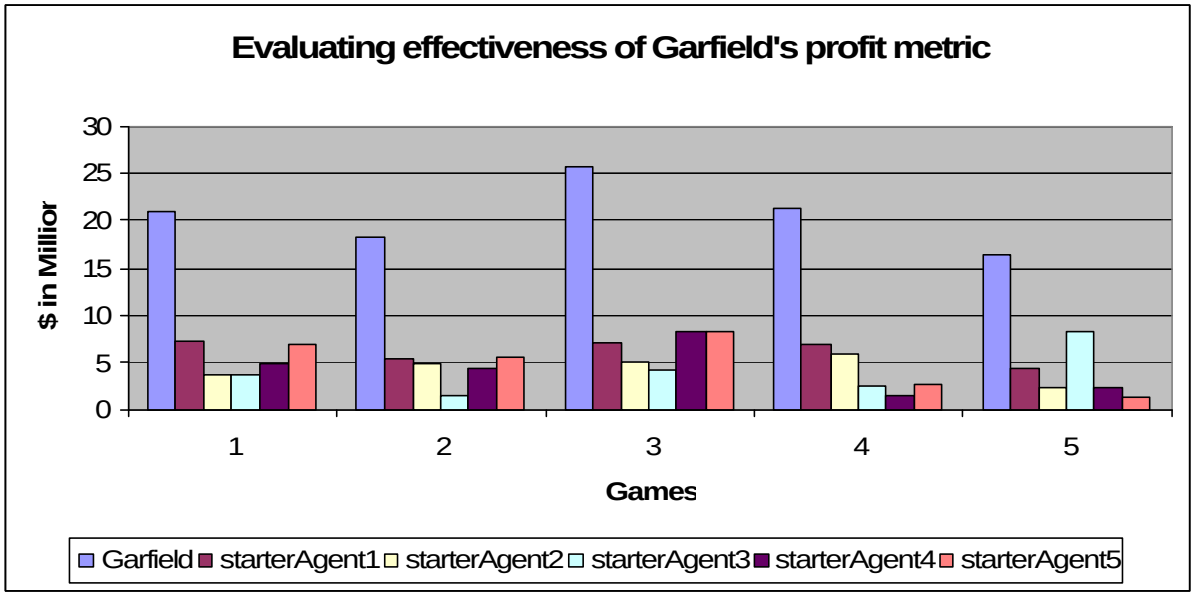

Fig. 5

5.5 Experiment 5

Objective: To evaluate Garfield's profit metric against agents who use high price for calculating profits

Description: Garfield calculates potential profit using last day's lowest winning price. In this experiment, we test this approach against starter agents who use last day's highest winning price to calculate profit.

Observation: For the remaining experiments, we set up a diverse bidding environment by changing the RFQ selection preferences of the starter agents. Thus, the market contains a mix of agents each of which sorts the customer RFQs based on a separate criterion (viz. Profit per Cycle, Order Quantity, Due Date, etc.). This should test the robustness of the agent.

As can be seen from the graph, Garfield's profit metric outperforms starter agents which use high price (from last day) to calculate potential profits

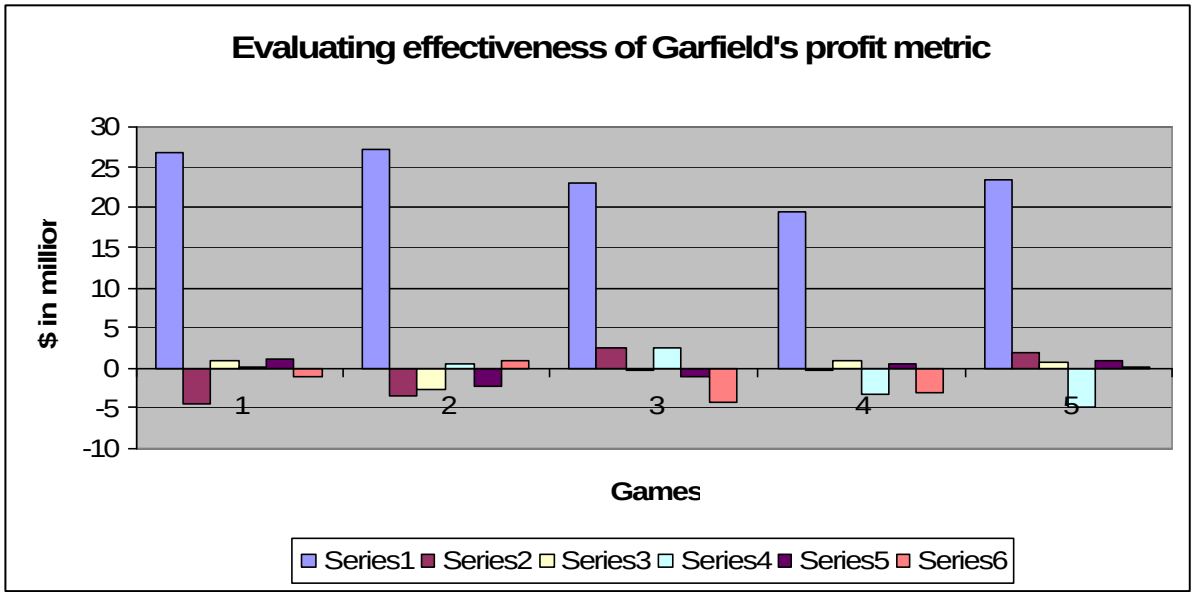

Fig. 6

5.6 Experiment 6

Objective: To evaluate the number of competing bids received for RFQs bid upon by Garfield in each type of bidding strategy.

Observation:

Strategy	Fraction on which no other agent bid (averaged over 5 games)
Undercut	17.26%
Minority	22.8%
Overbid	25.12%

From these experiments we observe that even in a diverse environment, Garfield manages to bid on considerable number of RFQs on which no other agent bid.

5.7 Experiment 7

Objective: To compare average profit per order earned by Garfield across all its bidding strategies

Observation:

Strategy	Average Profit per Order (averaged over 5 games)
Undercut	$365
Minority	$622
Overbid	$472

As hypothesized, the profit per order is maximum for the *minority* strategy.

5.8 Experiment 8

Objective: To compare the fraction of offers accepted in each type of Garfield's bidding strategy

Observation:

Strategy	Fraction of Offers accepted (averaged over 5 games)
Undercut	0.9582
Minority	0.3132
Overbid	0.0102

As hypothesized, majority of bids in *undercut* strategy are accepted in contrast to fewer high profit-margin offers accepted in other strategies.

6 Garfield's Performance in the Class Tournament

Garfield finished 3rd in the class competition comprising of 5 agents developed by class participants. It made an average profit of 4.004 Million over 16 games. It made losses in 3 games with the maximum loss being 6.36 Million.

Garfield was placed behind agents Simplicity and RedBull in terms of average profit. These agents rely on probability of offers being accepted to bid on a high number of RFQs and thus were able to get a larger number of offers. In contrast, Garfield focused on trying to achieve a balance between low-profit and high-profit markets. This turned out to be a relatively conservative strategy as compared to the top two and Garfield ended up making less profits.

The other agents Stormfront and JAgent finished 4th and 5th respectively. Stormfront relied only on undercutting to gain orders. Although it earned a large number of offers it could not make huge profits on them. JAgent seemed to be the most conservative agent in the competition. It focused on minimizing losses rather than trying to get more profits and hence did not bid very aggressively.

7 Conclusions

In this paper, we described Garfield, a Supply Chain Management agent which consists of a hybrid RFQ selection policy, a fuzzy customer demand predictor, and a supply manager which adapts to ordering patterns of other agents to a certain extent. We show that our strategy is robust against various types of agents with different environmental settings. We also introduce a unique hybrid RFQ selection policy which tries to achieve a balance between low-profit and high-profit markets. We perform experiments and analyze our agent extensively to show that such strategy is indeed feasible.

In future, we would like to increase the percentage of total offers bid on by giving more preference to our *overbid* strategy. We would also like to predict offer acceptance probabilities and use it to shift between strategies. Also, we would like to incorporate learning between games to adjust the fraction of resources committed to each bidding strategy.

References

[1] J.Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, S. Janson. The Supply Chain Management Game for the 2006 Trading Agent Competition.

[2] D. Pardoe, P.Stone. TacTex-05: A Champion Supply Chain Management Agent. *In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 06), Boston, MA, July 2006.*

[3] M. He, A. Rogers, X. Luo, and N. Jennings. Designing a successful trading agent for supply chain management. *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1159-1166. Hakodate, Japan, 2006.*

[4] C. Kiekintveld, J. Miller, P. Jordan, and M. Wellman. Controlling a supply chain agent using value-based decomposition. *Seventh ACM Conference on Electronic Commerce, Ann Arbor, MI, 2006*