An Analysis of RedAgent's Market-based Appoach to Supply Chain Management

Erik Zawadzki

December 4, 2005

Abstract

McGill's RedAgent, the winner of the 2003 Trading Agent Competition Supply Chain Management (TAC SCM) scenario is examined. Supply chain management (SCM), the TAC SCM game and RedAgent's unique market-based approach are introduced and described. RedAgent's internal PC allocation market is analyzed, drawing comparisons between it and the class of scheduling problems. The paper concludes with some remarks about the market-based approach in general, and RedAgent in particular.

1 Introduction

1.1 Motivation

Supply Chain Management (SCM) is an important economic task that has far-reaching effects in industry. Supply chains (also called logistical or supply networks) span the multi-stage process of converting raw materials into a finished product for end users, usually with many manufacturing phases. Supply chains typically include suppliers, manufacturers, warehouses, transportation companies, and retailers.

At present, supply chains are predominantly static and based on traditional relationships between the links of the supply chain. This means that businesses within a supply chain are unable to react quickly to opportunities, such as new customers, under existing static SCM policies. Businesses are often slow to react to detrimental events, such as resource shortages or sudden drops in consumer interest, making the supply chain very brittle. These problems are usually amplified across the chain: if one link fails to meet its quota, it creates a cascading shortage through the remaining chain. Both problems lead to suboptimal profits for the entire supply network.

1.2 Dynamic SCM and the TAC SCM scenario

The TAC SCM game was introduced in 2003 by the University of Minnesota, the Swedish Institute of Computer Science (SICS), and Carnegie-Mellon University

to encourage research into dynamic SCM. Dynamic SCM research promises more adaptive supply chain practices, which would in turn lead to more efficient economies and less wasted materials[2]. Ideally, dynamic SCM policies would allow industries to quickly capitalize on opportunities, and minimize losses when shortages do occur.

The game provides infrastructure for the design and analysis of autonomous SCM agents. Each agent governs the supply chain decisions for a single personal computer (PC) manufacturer. The scenario involves six of these agents in competition. These decisions include making policies for supply contract negotiation, assembly line management, shipping, and bidding for customer orders. Due to the stochastic nature of the TAC SCM setting[2], the vast strategy-space, and the combinatorial value of components, it is difficult to run traditional decision algorithms and planners.

This game spans one 'link' of a larger supply chain. For example, the agents in this game require CPUs manufactured by either Pintel or IMD. These two manufacturers also need to secure raw materials to fabricate their CPUs, and sell their CPUs to customers (which in this case includes our manufacturers). However, in order to reduce complexity, the component suppliers and customers have strategies that are specified in advance[2].

2 RedAgent

There have been a number of attempts to understand SCM as a set of interconnected subproblems, though there are varying opinions on the best way to characterize the game. Sadeh and Arunachalam[4] propose three subproblems: customer bidding, component procurement, and scheduling production. Benisch *et al.* [1] proposed a similar decomposition of the game, breaking down the game into four subproblems: allocation, customer bidding, scheduling production, and component procurement. The allocation subproblem is matching an already manufactured PC to an order. Allocation is a subproblem in one analysis and not in the other because of the difference in production philosophy. In the Sadeh analysis PCs were built for specific orders making allocation trivial. In the Benisch analysis PCs were built and later matched to orders, making allocation an interesting problem. These strategies are known as *build-to-order* and *buy-to-build*[3].

RedAgent uses a *buy-to-build* strategy, which means that it will keep a large inventory of fully manufactured PCs in storage, waiting to match them with orders. RedAgent is therefore much more responsive to orders than a *build-to-order* agent. In the 2003-SCM game, inventory storage costs were non-existent. However, later versions of the game added a per day cost of storing inventory, which makes the *buy-to-build* strategy less effective than it was in 2003-SCM[2].

RedAgent has five types of subagents, and each roughly corresponds to one of the four subproblems suggested by the Benisch analysis (two of the agent types work on the same subproblem). These agents interact in internal markets, and the result of this internal behavior drives the external behavior of RedAgent.

One of the important themes in most subproblem analyses is that all these subproblems are strongly interconnected. For instance, it is impossible for the agent to come up with a reasonable set of component request for quotes (RFQ) if the module that is responsible for component ordering is divorced from the module that bids on customer orders. Therefore, if an agent uses a modular design based around these subproblems then there needs to be a communication mechanism between these modules. In RedAgent's case, the markets act as a medium for this information exchange.

One substantial assumption this model makes is that the subagents are non-competitive, which makes sense as this market is entirely internal. This is an important difference between these auction-like mechanisms and actual auctions. Similar behaviour would occur if multiple divisions within the same company were bidding for resources. In these auction-like mechanisms, the agents are not utility maximizers as they will never, for example, bid strategically for personal gain at the expense of the company.

2.1 Order Agents

The Order Agents (OA) in RedAgent correspond to the allocation subproblem. The responsibility of the OA is to obtain computers from the RedAgent's internal PC market and then ship them to the customer. There is a single OA for each order that RedAgent has won. The OA bids for computers on the PC market, and the bids that the OA place play an important role in establishing the valuations of PCs for the agent as a whole. The bid price is calculated as:

$$b(p, c, q, d) = min(p, c) + (p - min(p, c))D_{sale}^{3-d} + q\sum_{i=max(d,-1)}^{3} D_{pen}^{i-d}$$

[3]

where p is the per unit price, c is the total cost of the components needed to assemble the unit (based on the average prices paid for the all components of a particular type in stock), q is the daily penalty for the order, and d is the relative date (hence $d = d_{order} - d_{current}$). This bidding formula is roughly: the base price plus the future discounted expected profit (discounted by a constant factor of D_{sale}), less the future discounted late penalties (discounted by D_{pen}. Note that q is a negative number in the above formula). It is important that D_{sale} is near one, and D_{pen} is smaller than D_{sale}. In RedAgent-2003, $D_{pen} = 0.7$ and $D_{sale} = 0.9$ [3]. While this may appear somewhat unusual since one might expect that the discounting factors would be tied to some index of the market variability, it turns out that the precise values of these parameters are unimportant to RedAgent's behaviour because of the robust market mechanism.

Once an OA has acquired PCs, it offers to sell to other OAs at utility (sale price less the cost of acquiring the PC). This allows for situations where another OA is suddenly created for high margin order, and desperately needs units of a particular PC configuration(*stock keeping unit* or SKU). This two-sided mechanism means that RedAgent's allocation solution is more sophisticated

than a simple greedy allocation.

2.2 Assembler Agents

An Assembler Agent (AA) is responsible for assembling a SKU. Each OA must deal with a specific AA because every order requires a single SKU and the SKU are not interchangeable. To assemble a PC, an AA must negotiate for required components and production cycles. The main priority of the AA is to acquire enough of these materials so that the inventory of PCs remains within certain bounds. The AA has bands of reserve prices to ensure that its inventory remains within this buffer. In particular, the AA drops the reserve price if it has too many PC, and raises the reserve price if it is running out of stock. The target number for a PC configuration is the expected number of PC of that configuration needed over the next 10 days(calculated as a 20-day moving average). This heuristic buffering behaviour means that the agent is less susceptible to fluctuations in component production.

Since the AA must acquire each component in a separate auction and the components have combinatorial valuations, each AA has significant exposure to risk. For instance, if a AA purchases enough production cycles to make 20 PCs, but in a subsequent auction only receives 15 CPUs, then the AA has over-acquired 5 PCs worth of production cycles, which are heavily sought after[3]. RedAgent avoids this problem through an indirect two-sided auction mechanism for the resource market, which allows excess resources to be sold to the other AAs.

2.3 Resource Agents

There are two types of resource agents: the component agents (CA) and the production agent (PA). The production agent may be seen as the factory owner, and it is the duty of the PA to sell the factory's production cycles to the 16 AAs. As there are a fixed number of production cycles per day the PA does not have to do any acquisition, and only has to deal with the AAs. The PA and the AA together correspond to the production scheduling subproblem.

The CAs do not have the same sort of renewable resource to govern, and must actively acquire components from the component suppliers (such as Pintel). The CAs use a banded pricing scheme to maintain a soft lower bound on the number of components. The lower bound is the projected sales for the next ten days, based on the running average of daily sales to the AAs. Again, if the inventory slips below the threshold, the CA increases the reserve price for the component. If the inventory exceeds the upper bound (which is calculated as the projected sales for the rest of the game), then CA drops the reserve price to encourage sales.

The CA must send out supplier RFQs to restock its component supply, and so corresponds to the component procurement subproblem. The CA projects the component sales to be the running average of daily component sales, discounted by a factor D_{comp}, which is 0.98 in RedAgent-2003. So in d days,

the CA will expect to have sold $\sum_{i=1}^{d} D_{comp}^{i} sale_c$ where $sale_c$ is the running average of the daily sales. If the projected inventory falls below the lower bound on day d_{lower}, then the CA sends out a RFQ to all available suppliers. The RFQ is due on d_{lower} and quantity requested is $min(3000, 200(220 - d_{lower}), sum_{i=d_{lower}}^{220} D_{comp}^{i} sale_c)$, or in other words the expected discounted sales from day d_{lower} to the last day, with a per order and a per day cap.

2.4 Bidder Agents

The Bidding Agent (BA) is tasked with bidding on customer RFQs, and so correspond to the consumer bidding subproblem. The bidder must bid low enough to win the order, but pad the bid enough that RedAgent still makes a profit. The BA learns the approximate valuation of PCs from the internal PC markets (a 4-day moving average over the PC market prices), and then adds a margin for each production cycle used. This margin is tuned based on the success rates of the bids.

3 Market Optimization

At the heart of SCM is a time-constrained, online, stochastic, optimization problem[1]. The internal market structure of RedAgent is just a intuitive model for this optimization problem. Optimization problems are classically phrased as either linear or integer linear programming problems (LP and ILP, respectively). Due to a stochastic environment (the randomized production capacity of the component suppliers is an example of this uncertainty), deterministic programming solutions like LP or ILP are clearly not adequate for optimizing the SCM problem. In contrast to RedAgent's internal market approach, some agents use an explicit stochastic programming (SP) approach, namely Brown University's Botticelli agent[1]. Botticelli was also a finalist in the 2003 SCM game.

There are reasons for interpreting the optimization as an interaction between internal markets. Firstly, the markets offer a very intuitive model for the optimization. This is important for designing and debugging the agent as it is much easier to work on a component when there is an intuitive understanding for how it should be functioning.

Additionally, the markets and subagents provide a very natural way of breaking up the problem for parallelization purposes. Since a simulated day in SCM is only 15 seconds, time is an important factor for optimization algorithms, and so these agents are usually run on distributed systems. Naturally and intuitively parallelizable agents again make for a simple and easily maintainable design.

The internal market design also makes the agent insensitive to the small variations in the formulation of problem[5]. This is one of the reasons why the internal markets are a good model for a stochastic environment such as the SCM game. This may also account for the robustness of the agent in the face of variations in its discounting factors (such as D_{pen}. D_{sale}, and D_{comp}).

3.1 PC Market

The interaction between the OA and the AA in the PC market, which corresponds to the allocation subproblem, has many similarities to a scheduling problem. In a scheduling problem there is a set of interested agents, a set of time slots, a set of reserve prices for each, and a set of agent valuations. In our particular problem, the "time slots" are PC sales at a particular time from a AA to a OA, and the AA establishes the reserve price based on heuristics designed to maintain inventory at a certain level.

Scheduling involves valuations that are strongly combinatorial. Specifically, any two sets of slots before the due date are substitutable, and time slots are strongly complementary as they only have utility as a set. Likewise, individual PCs only have utility when enough of them have been collected by the OA to form an order. PCs are also substitutable, but PCs acquired at different times are not perfectly substitutable because of storage costs introduced in the 2004 version of the game. Again, this was not a factor in the 2003 game.

Auction-like mechanisms for finding an optimal solution to the resource allocation problems appeal to the notion of competitive equilibrium. Both scheduling and a particular allocation establish a competitive equilibrium for a particular set of prices if the allocation gives each agent maximum utility given current prices (the utility is the valuation of the allocated set, less the price paid for the items in the set)and all units are sold for more than the reserve price. These equilibria do no always exist and are not necessarily unique. If a competitive equilibrium exists for a certain set of prices, then the allocation set for that equilibrium most also be an optimal for the scheduling problem[5].

The stochastic nature of the SCM game means that the allocation problem is more complicated than the simpler scheduling problem, as there is uncertainty about inventory levels in the future. Thus the auction must unfold over time to accommodate new information from the resource market, and as new OAs are created. This means that the PC market is significantly more complicated then the standard scheduling problem. Unfortunately, no known auction-like mechanism can guarantee to terminate on an optimal allocation even if an equilibrium exists for a particular scheduling problem[6]. One of the most popular auction-like protocols for scheduling, the ascending auction protocol, can be shown to terminate on a allocation that may be arbitrarily far away from the optimal allocation for certain scheduling problems[5].

4 Conclusion

Not being able to guarantee optimal results with a market mechanism for the allocation subproblem should not be too surprising. Scheduling is known to be a specific instance of set packing, which is NP-complete, so in polynomial time the best that can be found is an approximation. What is problematic is the potentially unbounded distance between the optimal solution and the solution that the ascending auction protocol finds in the simpler scheduling problem.

Despite this potential drawback, we find that RedAgent's internal market system performed very well in the 2003 SCM tournament, placing first. We can also note that RedAgent beat Botticelli in 2003 by a considerable margin, with an average score difference of 8.28 M. RedAgent was the most aggressive agent in terms of sending out offers to customers, was able to offer the lowest prices for each PC configuration[3], and was the first agent to manufacture a PC[4]. This indicates that while an auction-like optimization scheme will not return good results in every situation, in the more limited scope of SCM it can produce an effective agent.

It is therefore surprising that RedAgent failed to be a finalist in 2004, finishing last in its quarter-final with an embarrassing average score of -42.63 M. Not only was this the lowest score out of any of the quarter-finalists, but RedAgent also failed to play a single positive game in the quarter-finals. By contrast, during the 2004 tournament, Botticelli placed second in its quarter-final with a respectable average score of 12.57 M, first in its semi-final with an average score of 13.996 M, and fourth in the finals with an average score of 441,711. There was no RedAgent submission for the 2005 version of SCM.

While there is insufficient information to suggest a definite reason for RedAgent's failure in 2004, a possible explanation for RedAgent's poor performance is the previously mentioned addition of storage costs to the 2004 game. This would, as noted before, heavily impact a *buy-to-build* agent like RedAgent.

RedAgent also does not explicitly model the behaviour of other agents. RedAgent makes a non-competitive assumption and rolls the effect of other agents into the game's environment. This reduces RedAgent's ability to learn and counter strategies from other agents, and limits its effectiveness.

Another possible explanation is that RedAgent's market-based approach, while exceptionally intuitive and simple in concept, might make analysis, refinement and improvement more difficult. In a stochastic programming approach like Botticelli, the modeling assumptions that are used are explicit and well defined. There is also a large body of work applying stochastic programming solutions to other optimization problems. In RedAgent's heuristic market-based approach, many of the assumptions are tacit and are obscured by the auction-like mechanism.

Additionally, because of the insensitivity of RedAgent to parameter tuning, improving the agent would likely entail redesigning the entire market mechanism rather than incrementally improving specific modules and adjustment of constants. The lack of a general approach for using markets in a distributed scheduling setting[6] is further evidence of the difficulties faced by a market-based approach for a TAC SCM agent.

While RedAgent's simple and intuitive design produced early success in TAC SCM, it seems plausible that more rigorous and sophisticated approaches, such as Botticelli's stochastic programming approach, will mature to produce more profitable agents.

References

[1] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz. A stochastic programming approach to scheduling in tac scm. *Fifth ACM Conference on Electronic Commerce*, page 152, 2004.

[2] John Collins, Raghu Arunachalam, Norman Sadeh, Joakim Eriksson, Niclas Finne, and Sverker Janson. The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, Carnegie Mellon University, 2004.

[3] Philipp W. Keller, Felix-Olivier Duguay, and Doina Precup. Redagent-2003: An autonomous market-based supply-chain management agent. In *AAMAS*, pages 1182–1189, 2004.

[4] Norman Sadeh and Raghu Arunachalam. The 2003 supply chain management trading agent competition. *Proceedings 6th ACM Conference on Electronic Commerce*, page 113, 2004.

[5] Yoav Shoham and Kevin Leyton-Brown. *Multi Agent Systems*, chapter 2. Draft, 2005.

[6] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffery K. KacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economics Behavior*, 35:271, 2001.