PSUTAC: A Trading Agent Designed from Heuristics to Knowledge

Shuang Sun, Viswanath Avasarala, Tracy Mullen, John Yen
{ssun, vavasarala, tmullen, jyen}@ist.psu.edu
School of Information Sciences and Technology, the Pennsylvania State University
Information Sciences and Technology Building, University Park, PA 16802 – 6823

Abstract

The Trading Agent Competition has provided a challenging game environment, where competing agents engage in complex decision-making activities in a simulated supply chain domain. We analyze our agent, PSUTAC, together with five other top performing teams on their general strategies for making interrelated procurement, sales, production, and delivery decisions. Heuristic methods are found to form the core of the agent strategies. However, organizing, maintaining, and applying these heuristics in such a complex and uncertain domain is a nontrivial task. We propose a knowledge-based approach to organize these heuristics. Compared with the architecture that PSUTAC used before, the new design has shown many improvements including ease of coding, testing, and transparency.

1. Introduction

Network-enabled supply chains allow companies to outsource business processes more easily, affording quicker time-to-market, cost savings, and lowering risks. Multi-agent system design meshes well with modeling supply chain networks, as it inherently assumes that agents have their own goals, which may be anywhere from pure self-interest to cooperative, thus permitting more freedom of analysis compared to traditional simulation or analytic tools.

The 2003 Trading Agent Competition added a Supply Chain Management game (TAC/SCM) [1] to simulate research in this area. We compare the strategies used by our agent, PSUTAC, to five other top performing agents in this game. Drawing from this analysis, and from lessons learned during our experiences in this game, we propose a new knowledge-based architecture for PSUTAC.

Given that analytic tools are limited in their ability to model complex, multi-firm, multi-dimensional environments, the ability to interact with agents at the knowledge level[2] allows human managers to create and test strategies in a meaningful way. It also provides the groundwork for more semantically meaningful interactions among supply chain agents in the future.

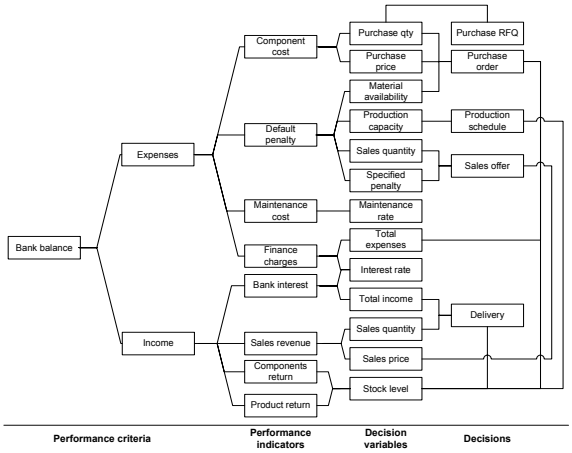

Figure 1. A break down analysis of the TAC/SCM problem.

Figure 1 shows the result of a causal relation analysis for the TAC/SCM game. On the left side, a performance criteria such as bank balance, is decomposed into income and expenses. Income and expenses encompass a composite of performance indicators such as finance charges and sales revenue. Performance indicators can be computed directly from various decision variables (i.e, sales revenue = sales quantity * sales price). On the right side, decisions about purchasing, production, sales, and delivery, draw on current values of decision variables, and also affect their future values. Making decisions in such a dynamic, interrelated, environment is what makes organizing and managing this knowledge non-trivial.

2. Current PSUTAC

The PSUTAC agent uses a "Make-to-Plan" approach to maximize its responsiveness to customers' demand. The approach involves (1) deciding on a required level of production, (2) purchasing all components at the beginning of the game, (3) producing with full capacity, and (4) selling on-hand stock at optimum prices. The aggressive buying strategy and relative conservative selling strategy helped the agent avoid problems created by supply uncertainties. However, the strategy could not handle changes in customer's demand because the component purchase decisions had to be made without any information about consumer demand.

Each day, the PSUTAC makes the following decisions: a) price setting b) what consumer's Request-For-Quotes (RFQ) to bid on and c) production scheduling. First, the agent uses a Gaussian function to set bidding prices randomly. The mean is based on the market price. The variance of the distribution is determined by two weighted decision factors: the current stock level and the overall demand. After setting the prices, the agent selects the bids that have a reserved price higher than the bidding price and the agent offers no more than what is on hand. In addition, the agent delivers an order immediately after it receives one from the customer. Therefore, by bidding and delivering conservatively, the agent achieves a high fill rate and a low penalty rate. Last, the agent schedules its productions, prioritizing products according to the inverse of the various products' stock level.

2.1. MRP strategies

Our competitive strategy has changed from focusing on high efficiency to focusing on high responsiveness. A high efficiency strategy prioritizes the agent's goal towards cost saving, while a high responsiveness strategy emphasizes a high fill rate for the customer' orders. Originally, our material requirement planning (MRP) strategy was Make-to-Order (i.e. plan components and productions requirements according to sales orders). However, we found the strategy led to low order fill rates due to long purchase lead-times and uncertain supplies. The strategy was used in the TAC'03 qualifying round. The average score was -120.0M, which was the lowest among the twenty competing teams.

Then we changed to a more conservative planning model: Make-to-Plan, which stocks a high volume of components and products to ensure a high level of availability. The new strategy was used in seeding round. The average score was 15.25M, which placed PSUTAC in sixth place.

Figure 2. MRP strategies.

Figure 2 illustrates the different MRP strategies and their impact on efficiency and responsiveness. High responsiveness and high efficiency cannot be achieved at the same time [3], thus agent designers need to consider such tradeoffs in their strategy choices.

3. A comparison of TAC agents

This section provides a brief description of agent strategies of TAC'03 finalists[1] and PSUTAC. The strategies (summarized in Table 1) include purchase RFQ, purchase order, production schedule, sales offer, pricing, and delivery. RedAgent, the winner of TAC'03/SCM, uses a market based mechanism based on internally simulated market for allocation of various agent resources and also for determining bid prices [4]. DeepMaize uses an equilibrium analysis to define an expected profitable zone of operation and then employs a feedback control mechanism to suppress deviations from this zone of operation [5]. TacTex uses heuristics that are based on predictions of future circumstances to determine offer prices, production schedules and other decision variables. These future predictions are made assuming the current relationship between decision variables in consideration will remain the same in the future game [6]. PackaTac uses a heuristic-based conservative strategy aimed at avoiding losses in low demand games [7]. Boticelli uses a stochastic programming approach to solve the production scheduling and bidding problems optimally [8].

It is important to note that most of the successful agents relied on simple heuristics in their decision making process. For example, our review of the

[1] WhiteBear was not included because published reports on it are unavailable.

agents' strategies reveals that most of the agents relied on large-scale procurement of components on the first day of the game. However, the preemptive strategy [5] employed by DeepMaize in the later stages of the tournament negatively affected the performance of the agents (including ours) who had no contingency strategies. This failure forces us to review the method that we used for the agent design.

Table 1. A comparison of the agents' decision strategies.

Teams	Purchase RFQ	Purchase order	Production	Sales offer	Pricing	Delivery
RedAgent [4]	Send large quantities on day one; buy to maintain safety stock.	Order offers in order of increasing price and accept offers with purchase lead time small enough so that future safety stock holds.	Assembler agents buy and sell production cycles in sealed double bid auctions.	Offer no more than current product inventory.	Based on closing prices in the internal PC market used by the agent.	An internal order agent arranges deliveries which are based on base price, profit, and penalty.
Deep Maize [5]	Use preemption strategy to block vendors' available capacities on the first day. Send RFQs to reduce the difference between current stock and reference inventory trajectory that is based on customer orders, expected component utilization, and required buffer.	Use hill climbing search algorithm optimizing the difference between expected value of projected inventory and cost of components.	Optimize over three-day period using linear programming.	Offer based on current product and component inventory and available free cycles.	Model the customer's RFQs as first price sealed bids with independent private values to find optimum bidding prices.	Not specified in the paper.
TacTex [6]	Send large quantities on day one; send RFQs to obtain safety stock.	Accept second day offers on the basis of the projected customer's demand and components' delivery schedule.	Greedy algorithm that sorts orders in order of decreasing value and schedules factory to fulfill them.	Offers based on probability calculations of obtaining an order based on historic game data.	Optimize the function (price-cost)*probability of winning the order.	Deliver orders in order of decreasing value.
Botticelli [8]	Send large quantities on day one.	Accept all offers.	A greedy approach that sorts orders based on due date and penalties and offers on expected profitability.	Model as a stochastic process with prices associated with winning probabilities and maximize over production schedule.	Model as a stochastic process with prices associated with winning probabilities and maximize over production schedule.	Deliver in order of ascending due date, then by descending penalty.
PackaTac [7]	Send small quantity of first day RFQs and later day RFQs to maintain safety stock.	Accept first day offers based on the customer's demand and due dates of component delivery.	Greedy algorithm based production on earliest due-date first heuristic.	Offer no more than current stock + amount that can be produced using current inventory.	Use heuristics based on component cost and reserve price.	Deliver orders in order of due date and in decrease order of profit.
PSUTAC	Send large quantities on day one.	Accept all offers.	Greedy algorithm that sorts products based on availability in inventory.	Offer no more than current product inventory.	Gaussian function based on market price, current inventory and number of RFQs.	Deliver immediately after an order is received.

3.1. Lessons learned: drawbacks of heuristics

In the current PSUTAC agent, the domain knowledge that is required by the agent is embedded as heuristics in various functional modules. This disorganized knowledge environment presented the designers with several obstacles to agent coding, knowledge review, and performance evaluation.

First, knowledge encoding becomes increasingly difficult when the number of heuristics increases. Although it is easy to embed a few heuristics inside functions, when the number of heuristics increases in magnitude, as is the case in a complex environment like TAC, knowledge management becomes a cumbersome task, since it is spread throughout an agent's decision points. Modifying a heuristic often involves updating code at many different places.

Second, knowledge becomes invisible for the designers when the number of heuristics increases. Designing a complete knowledge base that can respond efficiently to all situations or problems is not feasible in a complex problem domain like TAC. The knowledge base design is usually an iterative process with continuous inputs from human designers who determine the knowledge updates through the evaluation of the agent's performance. However, procedural coding of knowledge obscures the incomplete agent design, making it difficult to identify its shortcomings or incompleteness.

Finally, performance evaluation becomes increasingly difficult. When knowledge is distributed, particularly in an ad-hoc manner, it can be difficult to track the effects of individual pieces of knowledge on agent performance. Thus, isolating the effects of an individual heuristic and assessing its impact on agent performance is not straightforward.

These drawbacks proved to be critical to our agent's performance as the game became competitive. In particular, our failure to respond to the preemption strategy employed by Deep Maize in the semi final rounds proved fatal. While the agent designers were aware of deleterious situations, the complicated nature of heuristic information prevented adding knowledge necessary to respond effectively. These issues have motivated us to adopt a rule-based knowledge base using Jess (the Java Expert System Shell) for the TAC04.

4. The new PSUTAC

The new approach of the PSUTAC for TAC'04 employs an expert system for decision making. The approach is based on the premise that being able to express market strategies and knowledge in human-understandable form allows humans to both have more confidence in the agent's behavior, and also to contribute to the decision making based on experiences learned from market scenarios like the one presented in SCM. The expert system will be implemented in Jess, where the human's knowledge about the game domain will be captured in the form of rules [9].

4.1. Knowledge representation

Although representing knowledge as part of heuristic code or as rules seems to be equivalent, the two methods have fundamental differences in their purposes. A piece of heuristic is a rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood [10]. Heuristics are mainly used to simplify an algorithm. In contrast, knowledge is used for making decisions or for proposing solutions. Knowledge can be classified into two categories: (a) declarative knowledge that describes facts and relationships and (b) procedure knowledge that describes how to take actions.

The agent uses declarative knowledge to assess situations. For example, the rule in Table 2 indicates that current demand is high IF the agent knows the current number of RFQ (from the customer) and the number is larger than 180.

Table 2. A rule for assessing the customer's demand

```
((demand high)
        (number_of_RFQ ?rfq)
        (> ?rfq 180));
```

The agent uses procedural knowledge to select actions. In Jess, a procedural rule has two parts separated by the "=>" symbol. The first part consists of the Left Hand Side (LHS) patterns; the second part consists of the Right Hand Side (RHS) actions. The patterns are used to match the current situation (represented as facts) in the knowledge base, while the RHS contains function calls. For example, the agent uses the following rule in Table 3 to select the price setting functionalities (RHS actions) on the basis of the customer's demand and the present inventory level: IF the current demand is high, and the inventory is low, THEN set high prices (for the products).

Table 3. A rule for setting selling prices.

(demand high) (inventory low) => Price (high)

4.2. Knowledge acquisition

Knowledge acquisition, or the process of acquiring domain knowledge, has two different sources: human designers and the agent's own perceptive functions. In a complex environment like TAC where the range of situations that an agent may encounter is very large, it is essential for the agent designers to add or update the agent's knowledge directly. The TAC game stipulation that an agent cannot be changed in the middle of the game forces knowledge acquisition from human input to be completely offline.

Human designers get useful knowledge by analyzing historical data, which is recorded in a database. For example, the average selling price of a product in a low demand game can be set as the base price of that product in the games with similar situations. Human designers may also update the agent's knowledge by evaluating the agent's performance. However, what particular expertise will be useful and how to obtain knowledge through learning need our further investigation. In addition, the agent can update its knowledge online by the perceptive inputs. For example, after receiving a RFQ message, the agent may update its knowledge about the number of RFQs as (number_of_RFQ 230).

4.3. Knowledge-based system design

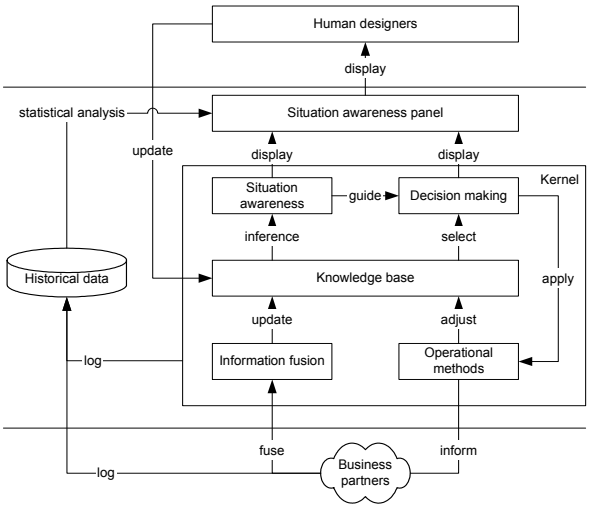

Figure 3. The new architecture for PSUTAC.

The new architecture (Figure 3) is based on the needs for knowledge management including knowledge acquisition, use, and update. The agent has two functional modules: a database that keeps track of all transactional data and a knowledge base centered kernel. The database records daily transactions and decisions. Human designers may analyze data and code knowledge for the agent offline, or monitor the agent's online performance through a situation awareness panel. In addition to the knowledge base (implemented in Jess), the kernel also contains four functional models on information fusion, situation awareness, decision making, and operational methods.

The information-fusion model processes the incoming messages and extracts and combines data into facts that are useful for making decisions. The agent needs to process a large volume of daily information. For example, as many as over 300 RFQs can be sent to the agent in a day, and each RFQ contains information about product, quantity, and delivery date. Therefore, the information content of the messages is large in volume. To keep the information as raw data is time-consuming for later decision-making functions because, each day, the agent only has limited time to process the data and make decisions. The information fusion module can greatly improve the efficiency by extracting and summarizing facts that are needed for making further inferences.

After the knowledge base receives the updates from the information fusion module, the situation awareness module makes inferences and provides the results that the agent uses to make decisions. Situation awareness can be done quickly because the process is based on the facts, not raw data. The result can be used directly to match with patterns specified in the procedure knowledge to make decisions on operations. Although the agent can make inferences directly based on the facts, this step is necessary because (a) it forces designers to be precise when defining the decision variables; (b) since the result will be recorded in the database, it helps designers in location of problems; (c) the results will be displayed at a situation awareness panel, where the designers can evaluate the results by comparing them with observations.

Each decision, or a fired procedure rule, triggers an operational method. For example, if the agent decides to set prices high, a high-price-setting function sets the selling prices to high values. Of course, the operation-method module defines a complete set of operators that are needed for the domain. It should be noted that the knowledge base does not need to keep knowledge that should be defined as an operator. For example, the procedure of the high-price-setting function should not be captured as knowledge.

4.4. Benefits of the knowledge-based system.

Compared with the old design, the knowledge based design provides many advantages: (a) different functional modules can access the entire knowledge base equally; (b) knowledge is organized as business rules instead of procedurally encoded "if…then…" statements, thus allowing an easy understanding of an agent's mental model in various situations. Designers can also test the knowledge offline with predefined commands. In addition, the effects of individual pieces of knowledge can be evaluated online by adding or removing rules. (c) the knowledge base is independent of the core agent design, thus enabling incremental knowledge addition and modification. For example, if designers notice that the market report about total demand and supply should also be considered when assessing the customer's demand, they can update the knowledge by retracting the rule defined in Table 2, and asserting new rules, as shown in Table 4.

Table 4. New rules for assessing the customer's demand

```
((rfq high)
        (number_of_RFQ ?rfq)
        (> ?rfq 180));
((market high)
        (demand ?demand)
        (supply ?supply)
        (> ?demand ?supply)) ;
((demand high)
        (market high)
        (rfq high));
```

5. Conclusion

One major drawback of our current design is its ability to handle uncertainty. Rules and facts cannot capture uncertainty or imprecision that inherently presented in the TAC/SCM game. For example, the rule defined in table 2 separates the RFQ numbers between high or low with an arbitrary number—180. Consequently, the agent may change its pricing decisions radically when the number of RFQs is changed from 180 to 181.

Bayesian statistics provide a good means for uncertain reasoning, but are impractical in complex environment. Therefore, we will investigate the applicability of integrating either Bayesian networks or fuzzy rules into our design. Fuzzy rules fit more naturally into our design, but Bayesian networks could also provide a means of evaluating the conditional likelihood of certain key decision variables before applying the rule set.

The TAC/SCM problems are designed to capture the complexities, time-sensitivities, and dynamics of the real SCM domain. Heuristics have been used extensively in the SCM game, indicating the criticality of human expertise in this domain. The complex nature of TAC game makes the heuristic development an iterative and protracted process. The approach we have proposed aims to organize the knowledge required for this game efficiently. For this reason, we believe the proposed design provides not only a viable agent for future competitions, but also an effective template for agent design in real supply chain scenarios.

6. References

[1] N. Sadeh, R. Arunachalam, J. Eriksson, N. Finne, and S. Janson, "TAC-03: A supply-chain trading competition. AI Magazine," *AI Magazine*, vol. 24, pp. 92-94, 2003.

[2] A. Newell, "The Knowledge Level," *Artificial Intelligence*, vol. 18, pp. 87-127, 1982.

[3] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*: Pearson Education International, 2001.

[4] P. W. Keller, F.-O. Duguay, and D. Precup, "RedAgent: Winner of TAC SCM 2003," *SIGecom Exchanges*, vol. 4, pp. 1-8, 2004.

[5] C. Kiekintveld, M. P. Wellman, S. Singh, and V. Soni, "Value-Driven Procurement in the TAC Supply Chain Game," *SIGecom Exchanges*, vol. 4, pp. 9-18, 2004.

[6] D. Pardoe and P. Stone, "TacTex-03: A Supply Chain Management Agent," *SIGecom Exchanges*, vol. 4, pp. 19-28, 2004.

[7] E. Dahlgren and P. R. Wurman, "PackaTAC: A Conservative Trading Agent," *SIGecom Exchanges*, vol. 4, pp. 38–45, 2004.

[8] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz, "Botticelli: A Supply Chain Management Agent Designed to Optimize under Uncertainty," *SIGecom Exchanges*, vol. 4, pp. 29-37, 2004.

[9] E. J. Friedman-Hill, "Jess, The Java Expert System Shell," Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess, 2003.

[10] L. Lexico Publishing Group, "Dictionary.com," 1995.