

# A Market-Pressure-Based Performance Evaluator for TAC-SCM \*

Brett Borghetti, Eric Sodomka, Maria Gini, and John Collins  
{borg,sodomka,gini,jcollins}@cs.umn.edu  
<http://www.cs.umn.edu/~borg>

Dept of Computer Science and Engineering, University of Minnesota,  
200 Union St SE, Minneapolis, MN 55455

**Abstract.** This paper proposes a novel method to characterize the performance of autonomous agents in the Trading Agent Competition for Supply Chain Management (TAC-SCM). We create benchmarking tools that manipulate market environments to control the conditions under which we test trading agents. Using these tools, we show how developers can inspect their agents and unveil behaviors that might otherwise have gone undiscovered.

## 1 Introduction

One of the most prominent proving grounds for current research in autonomous trading agents is the Trading Agent Competition for Supply Chain Management (TAC-SCM). In this yearly international event, autonomous agents battle for supremacy in a simulation where the highest profit-earning agent wins. In TAC-SCM, agents make all the decisions to run a virtual computer-manufacturing operation. They negotiate to purchase parts from suppliers, optimize their assembly lines, and sell by auction their products to customers. They manage parts and product inventories, minimize costs, optimize revenue, and try out-earn their competitors.

Agents compete against 5 other adversaries in each game. Different combinations of competitors, in addition to the randomness in the game, cause different market conditions to arise. An agent must perform well under many different market conditions to succeed.

To evaluate an agent's performance under various market conditions, we explored two key facets of testing: control over the in-game randomness we wanted to stabilize, and manipulation of the market characteristics.

We modified the TAC-SCM server to give a developer the ability to control the extent of the randomness in the suppliers and the customers. We also generated a limited replay capability so different combinations of agents could

---

\* Partial funding provided by NSF under grant IIS-0414466. We would like to thank the entire University of Minnesota Trading Agent Research Group for their help and support with this effort.

experience the identical sequence of random processes in the suppliers and customers in successive trials.

To manipulate the environment, we developed a pair of benchmark agents that control market conditions to simulate specific levels of supply and demand in the market. Any team wishing to evaluate their agent can use these stand-alone market manipulator agents to create a configurable level of pressure in the marketplace. The benchmark agents do not require alteration of the game server or the agent being examined. They control the supply and demand characteristics of a game by buying parts and selling computers at prices that generate the desired demand and supply in the marketplace.

The remainder of this paper is organized as follows. In section 2 we describe other research efforts in this area. Then we discuss some of the challenges we face in section 3. We follow with an explanation of our approach in section 4 and our experiments in section 5. Section 6 and 7 provide our plans for future work and conclusions.

## 2 Related Work

TAC-SCM allows research teams to develop trading agents and compare their relative performance in a complex, standardized environment [1]. Several teams developed methods of analyzing agent performance. The University of Southampton team examined running variations of their competitive agent with different risk strategies for customer pricing [2] to show that their competitive agent made the highest profit among the variations. They also developed a set of controlled experiments for measuring the relative performance of several variations of procurement strategies [3], including Short Term Planning, Long Term Planning, and Mixed strategy. The University of Michigan team analyzed post-competition performance of the TAC-SCM 2004 finalists and explored relationships between total profit and other measurements of performance [4]. The team at the University of Texas at Austin focused on comparing relative performance of variations of their TacTex [5–7] agent. Their baseline for each suite of controlled tests was a consistent set of other competitor agents downloaded from the SICS agent repository<sup>1</sup>.

While each of these teams compared relative agent performance no one developed a stable, universal benchmarking environment in which to characterize agent performance against a standard reference. We felt in order for the field to advance, we must explore this region of performance analysis.

## 3 Challenges

There are two key challenges in designing a useful benchmarking environment for TAC. The first is that outside of an actual competition, it is hard to recreate the effects of multiple agent interactions on the market. The second challenge is

---

<sup>1</sup> <http://www.sics.se/tac/showagents.php>

that after a team makes a slight modification or change of parameters, due to the inherent randomness in the game, the team must run a significant number of simulations to determine statistically whether or not the changes have improved the agent.

## 4 Approach

We wanted to develop a system to control the environmental conditions of the simulation. To do this, we decomposed the environmental control issue into two aspects:

- Managing the randomness and repeatability of the games.
- Manipulating the market conditions for focused observation of agent behavior.

### 4.1 Managing Randomness

In an environment such as the trading agent competition, there are many variables which can influence total performance of a given agent. These include the random variables such as the daily capacity of the suppliers and the demand of the customers, as well as the effects that other agents playing in the game have on the environment. While benchmarking an agent, we would like to reduce the randomness of a game, or alternately, replay the random variable sequences so that we can compare different agents' capabilities running under the identical scenario.

We explored two methods of managing randomness in our benchmarking system: minimizing the randomness, and controlling the seed to enable repeatable games.

To minimize the randomness of a given game, we altered the minimum and maximum intervals in the server configuration files. To create game with exactly average characteristics for example, we set average values for all  $[min, max]$  intervals that are configurable within the server. Thus our new settings for each interval are  $[x, x]$  where  $x = \frac{(min+max)}{2}$ . These changes allow us to benchmark agent performance within a relatively narrow window of supplier and customer behavior.

In order to make games repeatable we modified the TAC-SCM server. First, we identified and separated the usages of the TAC servers' random number generator into three categories: server-side independent random processes; server-side tie breaking; and user agent-side activity. Our goal was to control the server-side random processes without changing the behavior of the agents. We also wanted to eliminate opportunities for race conditions that would upset the sequence of random numbers. We modified the server by disconnecting the server side processes and running them with their own configurable random seed. Doing this ensured game repeatability for the random processes used by the server while still maintaining a separate true random number generation capability for user's agent behavior.

## 4.2 Manipulating the Market

To manipulate the market conditions for benchmarking performance, we developed two new agents, the *Market Relief* or “do nothing” agent, and the *Market Pressure* agent. The Market Relief Agent occupies one or more of the 6 slots in a TAC simulation without making any financial transactions. This agent provides relief to the market from the perspective of the other agents in two ways. First, it reduces demand on the suppliers which leads to a lowering of supplier prices. Second, it reduces available supply for the customers which causes customers to pay for computers from other agents.

Designing the Market Relief Agent was relatively simple. We used the example agent code available for download from the Swedish Institute of Computer Science (SICS) Trading Agent Competition website<sup>2</sup>. We modified several areas in the code where it made decisions regarding which customers’ request for quotes (RFQs) the agent should make offers on. By setting those decisions to never bid on RFQs, we effectively disabled the agent. Since it never made any bids to customers and it was designed as a build-to-order agent, it never ordered any supplies. When used in a testing environment with other agents, this agent reduces demand on the suppliers and reduces available supply for the inventory. Both of these actions are equivalent to reducing competition and pressure in the marketplace.

Conversely, the configurable Market Pressure Agent does the opposite: it increases available supply to customers, allowing them to pay less for computers while simultaneously putting more demand on suppliers, encouraging them to increase their prices. Our Market Pressure Agent operates by continually adjusting its customer offer prices to achieve a desired market share. Since it is a build-to-order agent, it purchases parts to build the computers ordered, and the market share achieved on the customer side is reflected on the supplier side. This agent can capture market share on the interval 0% to 100% because it has an unlimited line of credit and has no concern for its own profit earning capability. When our agent captures the desired market share, no other agent(s) can use that portion of the market: the Market Pressure Agent creates pressure in the marketplace.

By using combinations of Market Relief and Market Pressure Agents, developers can control the market environment. Figure 1 shows how the setting of Market Pressure affects remaining available market share as well as market share actually obtained by an agent under observation. When a developer makes alterations to a competitive agent they wish to observe, they can use these tools in concert with the repeatable-game server to benchmark and compare the change in performance of their alteration.

---

<sup>2</sup> <http://www.sics.se/tac/page.php?id=16>

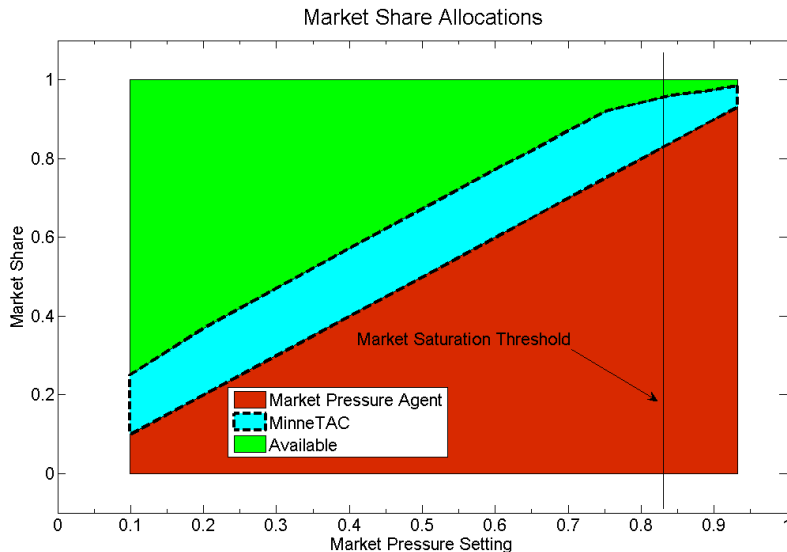


Fig. 1. Market Pressure Effects on MinneTAC and Available Market Share

## 5 Experiments

Using our Market Pressure Agent and a server set for a minimum-randomness average<sup>3</sup> game we examine what happens to an agent when we alter product supply and demand for parts in the marketplace. As shown in Figure 2, our MinneTAC [8] agent earns a relatively constant profit (shallow slope) until the Market Pressure Agent absorbs significant market share<sup>4</sup> near the *Market Saturation Threshold*: 83% (5/6) of the available market. Market Saturation represents the point at which there is little unmet customer demand remaining: if one agent wishes to sell more computers and gain market share from another agent, price wars occur.

In Figure 2 the area under the profit curve is a measure of the average profit-making ability of the agent with respect to a fixed reference. In general, if we wish to improve the average performance of an agent, we must increase the area under the curve.

First, notice the profit slope changes abruptly around 70% market pressure. The point at which the slope of agent's profit changes in this region is relevant:

<sup>3</sup> Recall that an average game is one in which we set average values for all  $[min, max]$  intervals that were configurable within the server. Thus our new settings for each interval are  $[x, x]$  where  $x = \frac{(min+max)}{2}$ .

<sup>4</sup> We measure market share using the *CMieux Analysis and Instrumentation Toolkit for TAC SCM* software [9] developed at the Carnegie Mellon University. This software is available at <http://www.cs.cmu.edu/~mbenisch/ait/> or in the 3rd party software section of the SICS TAC-SCM website.

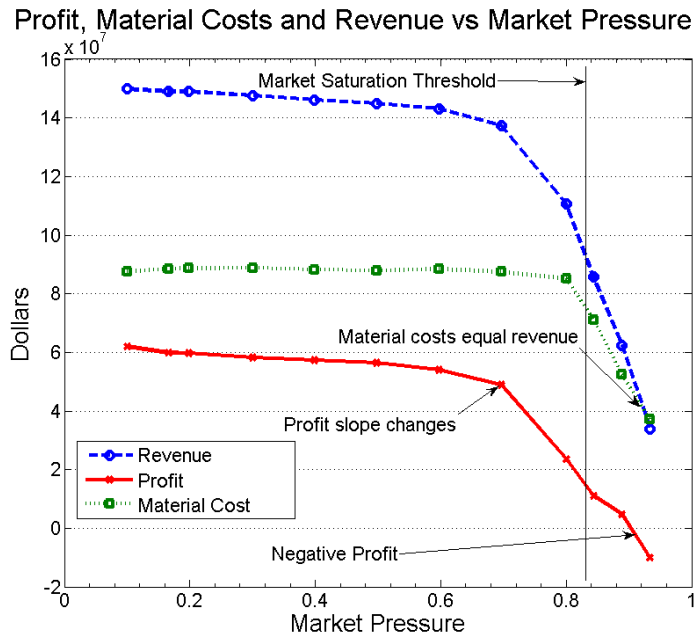


Fig. 2. Market Pressure Effects on MinneTAC Revenue

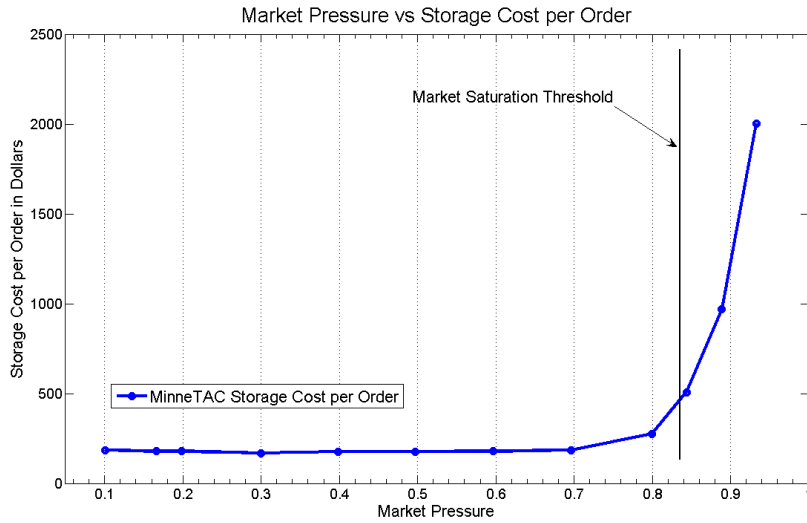
since a better performing agent has greater overall area under the curve, ideally we would like this point to be as far to the right as possible. Thus, if we make adjustments to our agent, we should chose adjustments that slide this point to a higher market pressure and make the slope more shallow.

Second, notice the where market pressure exceeds 92%. In this region, our agent is forced to lose money because parts cost more than the revenue we could earn by selling products. In this region our agent would be better off scaling back on financial transactions: since material costs exceed revenue, any further transactions would only lead to a deeper negative profit.

Another interesting phenomenon that occurred while evaluating MinneTAC was that as market pressure increased, the amount the agent spent on storage fees<sup>5</sup> per order increased drastically. Our intuitive explanation is that as market pressure escalates, there are fewer opportunities for an agent to buy low and sell high, and MinneTAC chooses to hold onto parts and products longer because the average number of days between a buy low and a sell high opportunity increases. The version of MinneTAC we tested does not factor storage cost into its procurement or sales decisions. As shown in Figure 3, under high pressure environments, MinneTAC experiences a 20-fold increase above nominal storage

<sup>5</sup> In a standard TAC-SCM competition, the storage rates are randomly determined constants on the interval [25%, 50%] The storage rate that will be used for an entire game is selected at random and broadcast to all agents at the start of the game.

costs per order for the agent<sup>6</sup>. The increase in storage costs poses an obvious threat to profit. A worst case analysis reveals that if the storage fee per game was set at maximum (50% per 220 days), and the agent was negotiating well for parts and receiving them at the best possible discount rate (about 50% of base cost), storage costs per day per dollar of inventory would be equal to  $\frac{0.5 \times 1}{0.5 \times 220} = 0.0045$ . In other words, to make a profit, an agent would have to sell the product for 5% more than the original parts cost for every 10 days it held those parts or products. Since profit margins in TAC-SCM finals tend to be very slim, agent developers must find ways of identifying tight markets, especially during games when storage fees are high. In these circumstances, the agent developers must consider the effects of storage costs in calculating the expected profit from sales when trying to decide whether to order parts and build computers on speculation.



**Fig. 3.** Market Pressure Effects on MinneTAC Storage Costs

Next, we compared the performance of several agents: the winning agent in the TAC-SCM 2005 competition, TacTex (University of Texas at Austin); the 3rd place finalist, Mertacor (Aristotle University of Thessaloniki, Greece); and MinneTAC, our agent that took 5th place in the finals. We set the server to use our repeatable mode<sup>7</sup> and examined the performance of these agents individually under various market pressure conditions. As shown in Figure 4 TacTex outperforms the other two agents by a wide margin (\$20M in this game)

<sup>6</sup> Our experiment used an average value for storage rate (37% of base part cost per 220 days).

<sup>7</sup> Recall that repeatable mode uses the modified server set with the identical initial random seeds for every game. This allows independent control of the sequence of random numbers generated for the server processes, tie-breaking, and agent behavior.

until the market pressure exceeds 70%. At this point, TacTex performance drops significantly, and the other two agents perform better. If we consider the area under each curve (the integral of the function depicted by the profit line), it is clear that TacTex has a higher area under the curve, and thus is most likely to perform the best on average<sup>8</sup> when competing against these other agents.

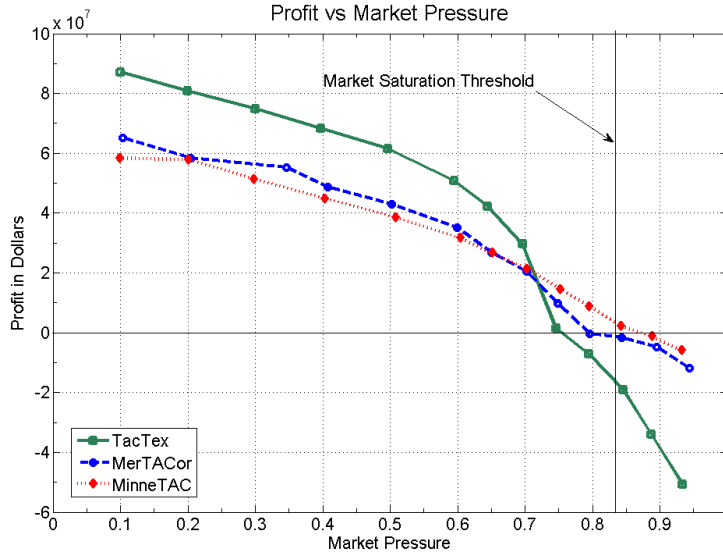


Fig. 4. Comparison of Agent Performance

## 6 Conclusion

We developed a framework of tools to control the simulation environment and measure the performance of trading agents under various market conditions. Since we are likely to see high market pressures exhibited by competitors in actual competitions, we can use these market pressure tools to simulate various market pressure levels and discover undesirable agent behavior before the agent is used in competition. Furthermore, teams can now evaluate their agent in a variety of market environments that were previously unable to be simulated with combinations of existing past competitors. Finally, these tools allow research

<sup>8</sup> Here, average means that in a large number of games, when the 3 remaining unnamed competitors in a 6 competitor game gain a random market share distributed uniformly across all values of possible available market share, TacTex is operating in the region of market pressure where it performs well, and it should, on average, outperform MinneTAC and Mertacor

teams to better understand how individual changes to their agents affect performance in various market conditions. We believe that agents optimized with the market pressure tools will perform better in actual competition.

## 7 Future Work

While we've provided a benchmark framework and a set of tools that will allow developers to characterize the performance of their agents against a standard reference, there are many areas in the framework that need refining.

For example, our current Market Pressure Agent has only a limited control over the pressure it creates in the supply-side chain. When a new customer order arrives, the Market Pressure Agent purchases parts immediately in high quantities with an as-soon-as-possible delivery date. But because other agents compete for parts in 3 dimensions (quantity, price, and delivery date), and the Market Pressure Agent only operates in the quantity and price dimensions, the Market Pressure Agent is unable to create pressure against future purchases of parts the other agents make. As a result, our agent tightly controls market share in the product market, but it only loosely influences the parts market. Resolving this problem is a topic for future research.

A second area of future work is to develop a variable-pressure agent. Instead of trying to achieve a fixed pressure level for the entire game, this new agent could achieve different market pressures for pre-determined intervals throughout the game. We could then examine the ability of an observed agent to react to the change in market pressure to measure its level of adaptability.

## References

1. Collins, J., Arunachalam, R., Sadeh, N., Ericsson, J., Finne, N., Janson, S.: The supply chain management game for the 2006 trading agent competition. Technical Report CMU-ISRI-05-132, Carnegie Mellon University, Pittsburgh, PA (2005)
2. Minghua He, Alex Rogers, D.E., Jennings, N.R.: Designing and evaluating an adaptive trading agent for supply chain management applications. In: IJCAI-05 Workshop on Trading Agent Design and Analysis TADA-05, Edinburgh, Scotland (2005)
3. Minghua He, Alex Rogers, X.L., Jennings, N.R.: Designing a successful trading agent for supply chain management. In: Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, (Hakodate, Japan)
4. Kiekintveld, C., Vorobeychik, Y., Wellman, M.P.: An analysis of the 2004 supply chain management trading agent competition. In: IJCAI-05 Workshop on Trading Agent Design and Analysis TADA-05, Edinburgh, Scotland (2005)
5. Pardoe, D., Stone, P.: TacTex-03: A supply chain management agent. SIGecom Exchanges: Special Issue on Trading Agent Design and Analysis 4(3) (2004) 19–28
6. Pardoe, D., Stone, P.: Bidding for customer orders in TAC SCM. In: AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI. (2004)
7. Pardoe, D., Stone, P.: Predictive planning for supply chain management. In: Proceedings of the International Conference on Automated Planning and Scheduling. (2006)

8. Ketter, W., Kryzhnyaya, E., Damer, S., McMillen, C., Agovic, A., Collins, J., Gini, M.: Analysis and design of supply-driven strategies in TAC-SCM. In: AAMAS-04 Workshop on Trading Agent Design and Analysis, New York (2004) 44–51
9. Benisch, M., Andrews, J., Bangerter, D., Kirchner, T., Tsai, B., Sadeh, N.: Cmieux analysis and instrumentation toolkit for tac scm. Technical Report CMU-ISRI-05-127, School of Computer Science, Carnegie Mellon University (2005)