Information Systems and e-Business Management (2019) 17:117–157
https://doi.org/10.1007/s10257-018-0378-y

ORIGINAL ARTICLE

Application of adaptive strategy for supply chain agent

Yoon Sang Lee[1] · **Riyaz Sikora**[2]

Received: 22 January 2018 / Revised: 18 April 2018 / Accepted: 28 September 2018 /
Published online: 4 October 2018

Abstract

With the tremendous increase in the globalization of trade the corresponding supply chains supporting the manufacture, distribution and supply of goods has become extremely complex. Intelligent agents can help with the problem of effective management of these complex supply chains. In this paper we introduce the design, implementation and testing of an intelligent agent for handling procurement, customer sales, and scheduling of production in a stylized supply chain environment. The supply chain environment used in this paper is modeled after the trading agent competition that is held annually to choose the best agent for managing a supply chain. Our supply chain agent, which we call SCMaster, uses dynamic inventory control and various reinforcement learning techniques like Q-learning, Softmax, ε-greedy, and sliding window protocol to make our agent adapt dynamically to the changing environment created by competing agents. A multi-agent simulation environment is developed in Java to test the efficacy of our agent design. Two competing agents are created modeled after the winners of past trading agent competitions and are tested against our agent in various experimental designs. Results of simulations show that our agent has better performance compared to the other agents.

Keywords Multi-agent systems · Artificial intelligence · E-commerce · Simulation · Supply chain management

✉ Riyaz Sikora
rsikora@uta.edu

Yoon Sang Lee
lee_yoon@columbusstate.edu

[1] D. Abbott Turner College of Business, Columbus State University, 4225 University Avenue, Columbus, GA 31907, USA

[2] College of Business, University of Texas at Arlington, 701 West Street, Arlington, TX 76019, USA

1 Introduction

1.1 Introduction of the research

As procurement and sales options diversify from local to global and from offline to online market, the importance of effective supply chain management (SCM) is increasingly becoming critical (Acquity Group 2014). In a supply chain there exist multiple vendors and various types of customers that are affected by other self-interested agents in the same supply chain. Due to its complexity a supply chain is recognized as a complex adaptive system requiring dynamic and flexible reactions against the changes of environment (Choi et al. 2001). The decision making regarding where and when to purchase raw material, how much to produce products, and at which price to sell the product has been considered important, and such decision makings have been traditionally made by a human supply chain manager. However, due to the high degree of interactions among agents, a large set of internal/external variables, and multiple types of items that need to be considered simultaneously, supply chain managers have had difficulties in their decision makings.

In such a situation, more flexible and dynamic practices may offer better matches between suppliers and customers (Collins et al. 2006). However, establishing a supply chain management model considering upstream, downstream, and operations for an enterprise is a complex task because the model should not only orchestrate the decision making but also appropriately react against the change of environment resulting from direct or indirect interactions of suppliers, customers, and competitors. In addition, evaluating the performance of agents is difficult because it is hard to gain the real world business data for suppliers, customers, and competitors. Saar-Tsechansky (2015) suggested a simulation as a good alternative to evaluate the performance of the model by entailing the artificial construction of key properties when real business data is not accessible, and Adami et al. (2016) provided the validity of an agent-based method where the mathematical treatment is not applicable by comparing the result from the agent-based simulation and game theoretical analysis. Hence, artificial agent can be adopted as a decision-making technology for critical problems in electronic marketplaces by calibrating agents with real data (Chaturvedi et al. 2006). The B2B e-commerce simulation environment in a case study of personal computer (PC) industry suggested in Chaturvedi et al. (2006) demonstrated the efficacy of the agent-base modeling (ABM) for the real world problem. As shown in the studies, since an agent-based simulation can be used as an alternative for a real world environment by providing controlled abstraction of the real world in this way, this paper uses trading agent competition for supply chain management (TAC-SCM), a popular supply chain trading agent competition, as a guideline. We established a theoretical model for a supply chain trading agent that includes procurement, production scheduling, and sales functions. We then implemented a simulation environment and an agent based on our model.

TAC-SCM is a supply chain trading agent competition jointly suggested by a team of researchers from the e-Supply Chain Management Lab at Carnegie Mellon University, the University of Minnesota, and the Swedish Institute of

Computer Science (SICS). The purpose of this competition is to capture many of the challenges involved in supporting dynamic supply chain practices, while keeping the rules of the game sufficiently simple (Collins et al. 2006). TAC-SCM simulates a B2B supply chain model of computer vendors that manufacture PCs by purchasing parts from upstream, which is similar to Dell Computer's supply chain model. Computer vendors in the simulation do not manufacture any computer parts. They purchase all of the parts from suppliers, assemble them into PCs, and sell the completed computers to customers. Therefore, this simulation is composed of suppliers, customers, and PC manufacturer agents, and the game is operated by many detailed rules defining the interactions of three main elements.

As already known in supply chain management industry, the decision making involved in inventory management and selection of bidding prices to win customer orders are difficult for humans due to the supply chains' dynamics and complexity. If a decision support tool could provide support for a human supply chain manager's decision making, it is expected that the manager's decision making process will be more efficient and accurate. In addition, a supply chain management information systems could contribute to the improvement of operational efficiency, operational flexibility, and internal/external analysis (McLaren et al. 2004). According to the report from Acquity Group (2014), eighteen percent of buyers spent 90% or more of their procurement budget on online purchases in 2014. Nowadays, enterprises procure their items from the supply chain rather than producing all parts by themselves as the coverage of supply is widened to global and even to online. The adoption of automation is in its infancy, but the trend of automation will grow in the future in various industries, such as in self-driving cars and the food chain industry (Forbes Technology Council 2016). Hence, a fully automated system will be an alternative to enhance the efficiency of supply chain management.

In TAC-SCM literature, most of the studies on agents suggested mathematical or algorithmic solutions focusing on the optimization. Some agents adopted a family of linear programming as a main solution, and others applied a family of greedy algorithms based on their prediction from previous games data. While other solutions using machine learning (ML) techniques such as data mining, particle filter, k-nearest neighbor are applied for the prediction of the future situation of supply chain, the most common approach was to control the decision making of agent based on static rules of action. In the management science field, a group of studies of supply chain agents applied adaptive learning techniques such as reinforcement learning (RL) and meta-learning to enhance the agents' performance by providing more flexibilities on agents' strategy. However, the studies focused on partial problems of supply chain management (i.e., inventory management, retail problem, ordering policy, etc.). Therefore, to improve the entire performance of supply chain management of an enterprise, three areas of supply chain management, which are procurement, production, and sales, should be considered together. For example, a low sales performance in a certain period might cause high inventory level even if procurement was maintained at an average level; or, low sales prices might increase the revenue, but if it is too low, the total revenue of agent would be lower than the cost. However, as far as we know, the supply chain agent model incorporating three areas of supply chain

management (procurement, production, and sales) and applying adaptive learning techniques in those areas has not been implemented.

The purpose of this study is to propose a model of a supply chain agent that enhances the SCM agent's performance. One requirement of automated supply chain systems is that it should be adaptive to the events from the outside of the system, and, therefore, the system should understand the events and provide the appropriate responds to deal with those events (O'Leary 2008). Hence, we focus on establishing an adaptive supply chain management strategy because the strategy of using static supply chain management rules presented in the literature is not adequate to handle a dynamic environment. We do that by using RL and meta-learning for inventory management and customer bidding. For inventory management, we developed a model estimating a new inventory threshold based on the changes in difference between inventory threshold and actual inventory, and applied Q-learning (Sutton and Barto 1998) and ε-greedy (Alpaydin 2014) to make the model more adaptive. For the sales performance, we designed a bidding strategy based on a sliding window protocol (Sikora and Sachdev 2008) that enables the agent to learn the winning probabilities of bidding prices. We developed a simulation environment and evaluated our model of the agent as follows. First, we implemented the TAC-SCM design but modified some rules to implement more realistic business environment. Second, we implemented two agents that have been reported as performing best in TAC-SCM literature. Third, we chose the better of the two agents and modified the agent by adding our strategies. Finally, we executed the simulation in 12 different simulation settings and evaluated the performance of our model. In all simulations, the agent implementing our model provided better performance than other agents.

1.2 Organization of the study

The rest of this study is as follows. In Sect. 2, agents from previous studies are discussed. The studies regarding TAC-SCM agents and other agents using adaptive machine learning techniques are discussed. In Sect. 3, the model of the agent in this study is described. In Sect. 4, the simulation design and competitors in the simulation are provided. In Sect. 5, three different types of simulations and their result are described, and the results are analyzed to find the weakness and strength of agents' design in Sect. 6. In the last section, Sect. 7, a conclusion of this study and future research topics are presented.

2 Related work

Due to the difficulties in supply chain management, a group of researchers suggested a competition named TAC-SCM (Collins et al. 2006). The specification regulates the structure of supply chain and the trading rules between suppliers and customers. Each game in the competition is composed of PC manufacturer (supply chain trading) agents, suppliers, and customers. PC manufacturing agents purchase parts from suppliers, produce PCs, and sell them to customers. Hence, the manufacturer agents

have to decide when and how many of the components to purchase from suppliers, how much to produce, and at which price they will sell the products to customers. This simulation framework provides core elements of the supply chain while keeping the structure simple enough (Collins et al. 2006).

The key factor is how to measure the performance of an agent. As the measure of trading agents performance, profit is used in TAC-SCM. However, since the amount of customer demand on each day is created using random function, and since it causes different profit levels in different simulations, there have been discussions on providing the same level of customer demand in order to compare the profits of agents in different simulations. The Swedish Institution of Computer Science (SICS) has provided a simulation environment to the annual TAC-SCM tournament. While it has been used by many agents as the test environment, comparison of the results from different runs of simulation was limited due to different customer demands in each execution of simulation. To overcome these differences, a demand-adjusted profit (DAP) is proposed by Wellman et al. (2006). They provide a profit adjustment method using different levels of customer demands. As a different approach, a controlling server providing the same market conditions during multiple games is provided by the University of Minnesota team in Borghetti and Sodomka (2006). They identify two issues: how to manipulate the market conditions for observing agent behavior and how to manage the randomness and repeatability of the games. All of the approaches aim at creating the same levels of customer demands while maintaining the randomness in the level of customer demands.

In SCM practice and TAC-SCM, major decision making must be made in three fields: (1) how many parts to purchase from suppliers; (2) what is the best price to win the customers' offer, while gaining profit; and, (3) how many and which type of product to produce within a limited production capacity. Accordingly agents in TAC-SCM focus on procurement and inventory management, sales, and production.

The first category of decision making is about the amount of procurement, and it is directly related to inventory management. The most popular strategy is to use the concept of inventory threshold. The advantage of this concept is that it prevents the shortage of components caused by a sudden increase of demand while it also has the drawback of sporadic higher procurement and higher holding costs. Agent A (Sibdari et al. 2012) uses a high level of inventory threshold to avoid shortage of materials for production and sends request for quote (RFQ) to suppliers to fill the gap between actual inventory and the threshold. MinneTAC (Collins et al. 2009) combines procurement and sales through future demand estimates. The future demand estimator informs the procurement decision module of what sales modules like to sell and the procurement decision module purchases materials based on this estimation. However, this process may cause low sales-low procurement-low sales loop, so they apply safety-stock-monitor to override the resulting low quotas. Metacor (Chatzidimitriou et al. 2008) basically used the assemble-to-order strategy, but they combined the concept of threshold to overcome the uncertainty of on-time delivery, which is based on demand and procurement lead time, suggested by Simchi-Levi et al. (2008) and Cheng et al. (2002). DeepMaize (Kiekintveld et al. 2006) also uses the notion of threshold, named baseline buffer level, based on inventory projection. To decide the procurement amount DeepMaize calculates a new inventory

projection from the current inventory projection and the quantity required for outstanding customer orders. Once a new inventory projection is created, DeepMaize creates a procurement amount to fill the gap between the new inventory projection and the inventory threshold. TacTex-06 (Pardoe and Stone 2009) uses different thresholds for different component categories. To decide procurement amount, TacTex-06 subtracts the amount projected to be used for production from current inventory and add the amount to be delivered in the future to the current inventory; the result is called intended deliveries. Then, the gap between the intended inventory and threshold is decided as the procurement amount.

All agents consider production scheduling problem as an optimization problem, so they adopted a mathematical method or a greedy algorithm for the solution. One group of agents focuses on the minimization of the penalty created when customer orders have not met the due date, so they apply the optimization technique based on the due date of the order. Agent A (Sibdari et al. 2012) and CrocodileAgent (Podobnik et al. 2006) sort the orders by due dates and schedule the production for the orders whose due dates are closer. FreeAgent also focuses on the delivery dates of customer orders (Eriksson et al. 2006). CiMeux's production scheduling is conducted in the Scheduling module by using Vepsalainen's apparent tardiness cost (ATC) dispatch rule (Vepsalainen and Morton 1987). The priority for ATC is on the orders with large penalties and little time to complete because they require the most immediate action due to high volume of penalty (Benisch et al. 2009). Another group of agents focuses on the maximization of profit. DeepMaize (Kiekintveld et al. 2006) uses a greedy algorithm to maximize the marginal profit. To calculate a marginal profit, DeepMaize uses an expected revenue and an expected cost since the strategy of DeepMaize is to produce products based on the demand prediction from the previous games' data. Botticelli (Benisch et al. 2004a, b) formulates this problem as a probabilistic scheduling problem, so they apply stochastic programming to establish production schedules that optimize profit. TacTex-06 (Pardoe and Stone 2009) uses a greedy algorithm to maximize the profit of production schedule. They sort orders by profit and scheduler tries to produce as late as possible, considering production capacity and materials.

Sales function is considered as a maximization problem by all agents. Since the goal of a trading agent is to gain higher profit, agents have to increase the revenue by winning more customer orders. Excessive orders, however, might cause a higher penalty if they are not fulfilled because of the lack of production capacity or components, so the formulation of the sale problem not only includes profit related factors, such as cost of components, expected winning probability, and bidding price, but also includes the status of production resources. Therefore, while all of the agents in TAC-SCM focus on profit maximization, they vary in the pricing and production resource management strategies. Foreseer (Burke et al. 2006) adopted a mixed integer linear programming and their objective function maximized the expected profit. Mertacor (Chatzidimitriou et al. 2008) combined several techniques and historical data for customer bidding. For the bidding price, they combined a multiple linear regression (MLR) model and a linear interpolation. CiMeux (Benisch et al. 2009) defined the bidding problem as a continuous knapsack (CKP) problem. They defined each knapsack as RFQs for a type of product and considered weight as their

quantities. They used winning probability distribution to calculate the expected revenue for a given product type and price.

DeepMaize (Kiekintveld et al. 2006) used a gradient descent search to find a set of bids that approximately optimized the expected profit of resulting set of orders. The expected profit is defined by using the product of winning probability, profit, and quantity of offer. To decide a set offer to bid, DeepMaize maximizes the sum of expected profits of offers by applying the gradient descent search.

TacTex (Pardoe and Stone 2009) also chooses a set of offers by maximizing the expected profit with a greedy algorithm. To secure the production capacity TacTex creates a set of predicted RFQs and considers actual and predicted RFQs as a set of RFQs to consider for offer. With the set of RFQs, TacTex decided a set of RFQs that maximize the expected profit by applying a greedy algorithm. Agent A (Sibdari et al. 2012) analyzed customer RFQs by product type, reserve price, due date, and penalty and identified 81 RFQ types. With this categorization, they created an Expected Profit Matrix (EPM) providing expected profit by the given time and production capacity. To fill EPM, they conducted experiments to collect the data for dynamic programming and filled the EPM by applying dynamic programming. The characteristic of EPM is that the expected profit is calculated from the historical data. Based on their simulation, they found that the EPM enhanced the performance of Agent A.

As seen in TAC-SCM literature, there are some trends of strategies used in three major functional modules (procurement, production scheduling, sales) of agents. For the procurement, the static inventory thresholds are used; this threshold works as a guideline for deciding the procurement amount. Scheduling problem is considered as an optimization problem in all agents. Sales problem is formulated as an optimization problem and two major approaches are found. One group of research tried adopting a family of linear programming while others used a greedy search algorithm. The techniques are different but the goal of all these techniques is to pursue the maximum profit since profit is a proxy for the combination of cost, revenue, and production scheduling. Another notable trend is the use of historical game data. Most of the agents used not only the current game's data but also the historical data from previous games to establish a prediction model for three areas in SCM.

3 Model of proposed agent-SCMaster

Since the purpose of this study is to develop a model of supply chain agent that enhances the SCM agent's entire performance, we focused on the performance improvement of procurement, sales, production scheduling functions, and the coordination between them. We choose the approach made in Sibdari et al. (2012). To provide proxies for our model, we first developed two agents, named Agent-D and Agent-T described in Sect. 4.3, from existing best practices in TAC-SCM literature. We then modified Agent-T by applying the strategies presented in this section and named it SCMaster.

The approach we made for procurement is to apply the dynamic inventory threshold. The notion of inventory threshold is used in most of TAC-SCM agents, but

most of them use static threshold. We focused on a dynamic inventory threshold to reduce the gap between inventory threshold and actual inventory while managing a certain level of buffer inventory for sudden increase of customer demand. We developed a formula estimating a new inventory threshold based on the change of the gap between inventory threshold and actual inventory. Based on that formula, we control the inventory threshold dynamically by applying Q-learning to the learning rate. Since competitors in the simulation keep changing their bidding strategy dynamically, a dynamic bidding strategy is adopted to provide a better chance of winning customer offers. Hence, we adopted a sliding window protocol to predict the winning probability of different bidding prices. The goal of the production strategy is to minimize the penalty amount caused by the late delivery. We sorted customer orders by the profit of each order to maximize the sum of profits like most of agents in TAC-SCM, but when allocating the production resource, we scheduled the production for outstanding orders on most recent available days and delayed the production for the orders that still have some days to delivery date by as much as possible. Details of the strategy used in SCMaster are describe in the following sections.

3.1 Procurement

The role of the procurement module is to carry out all tasks regarding the procurement of components from suppliers. Its sub tasks are creating procurement RFQs and making a decision whether to accept the procurement offers and sending orders to suppliers.

According to the purchasing process, SCMaster creates and sends procurement RFQs to suppliers to purchase components that each supplier provides. To make decisions for RFQs, the procurement module uses the concept of inventory threshold used by many agents in TAC-SCM. While all of the other agents used fixed inventory threshold, we used the concept of dynamic inventory threshold for each type of component to reduce the component holding cost by reducing the extra inventory. To control the inventory dynamically, we created a formula to track the changes of difference between inventory threshold and actual inventory.

The assumption used in the formulation of this formula is a parallel trend assumption used in Difference in Differences (DID) (Abadie 2005). In DID, the additional change of difference between outcomes of control groups and treatment groups at the end of a time period, compared to one at the start of the period, is considered a treatment effect. The DID is measured as in Eq. (1)

$$\begin{aligned} d = &\{E[Y(i,1)|D(i,1)=1] - E(Y(i,1)|D(i,1)=0)\} \\ &- \{E[Y(i,0)|D(i,1)=1] - E[Y(i,0)|D(i,1)=0]\} \end{aligned} \tag{1}$$

where d is the same sample counter. $Y(i, 1)$ is the outcome of interest for individual i at time t and $D(i, 1)=1$ denotes if individual i has been exposed to the treatment previous to period t, $D(i, 1) = 0$ otherwise. Hence, the difference is expected zero in a parallel trend assumption.

As assumed in DID, change of difference may explain the result of certain effects. In our case the changes in the differences between inventory threshold and actual inventory level may occur as the result of an agent's operational situation. According to this reasoning, we developed the following relation

$$B^c_{t+1} = B^c_t + \alpha\left[\left(B^c_t - I^c_t\right) - \left(B^c_{t-1} - I^c_{t-1}\right)\right] \tag{2}$$

where B^c_t is the inventory threshold of component c on day t, I^c_t is the inventory level of component c on day t, and α is the learning rate.

As seen in Eq. (2), a new inventory threshold is determined by adding the change in amount of gap between inventory threshold and actual inventory to old inventory threshold. The learning rate controls the rate at which the inventory threshold is updated. Instead of using a fixed value for the learning rate we applied a reinforcement learning technique called Q-learning to dynamically update the value of the learning rate from among the following three values: 1.0 for low, 2.0 for medium, and 3.0 for high. We used the Softmax function (Sutton and Barto 1998) for selecting the value of the learning rate. Softmax, using a Gibbs or Boltzmann distribution, is applied to action selection as shown in Eq. (3):

$$\Pr(a) = \frac{e^{Q(a)/\tau}}{\sum_{i=1}^{n} e^{Q(i)/\tau}} \quad \text{where} \quad \tau > 0 \tag{3}$$

where a is the action selected (which in our case is the value of the learning rate α), $Q(a)$ is the action-value estimation function (which gives an estimate of the value of choosing that learning rate), and τ is the temperature parameter used for Softmax function, which controls the trade-off between exploiting the current best value of the learning rate and exploring the other values.

The function $Q(a)$ is defined as a time-weighted function as used in Sikora (2008) and it is shown in Eq. (4). In a time-weight method, action value estimates are the sample average of past observed rewards, with recent rewards getting a higher weight. If an action a has been taken k times in the past, then the $(k+1)$ *th* estimate of the action-value will be given by the Eq. (4):

$$\mathrm{Q}_{k+1}(a) = \sum_{i=1}^{k+1} w_i r_i(a) = Q_k(a) + \alpha\left[r_{k+1}(a) - Q_k(a)\right] \tag{4}$$

where i is the time, $r_i(a)$ is the actual reward received when taking action a, w_i is the weight, and α is the learning rate. For the action-value estimation function $Q(a)$ we used profit, and for the value $r_i(a)$, the sum of the profit of past 4 days is used. The window of past 4 days was heuristically selected for the profit calculation. If the window is too narrow, it will not correctly include the performance information of recent actions. If it is too wide, the equation will have too much noise. For the learning rate α in Eq. (4), 2.0 was also chosen heuristically. For the Q(a) value, an optimistic starting value of Q_0 was used because an optimistic action value encourages the exploration in the beginning, although the time-weighted average action-value

method will not be affected by the bias caused by initial estimates of Q_0 (Sikora 2008). For the temperature of Softmax method, τ, the ε-greedy method is applied where it chooses the current best value for most of the time but with a small probability of ε it chooses one of the other values. In this model, 0.1 was used as the value of ε. The equation used for ε-greedy is given in Eq. (5)

$$\begin{aligned} \Pr(a) &= \frac{1-\varepsilon}{m} \qquad if\ a = \arg\max_i (Q(i)) \\ &= \frac{\varepsilon}{k-1} \qquad else \end{aligned} \tag{5}$$

where ε is a small probability, m is the number of max values of Q(a), and k is the number of remaining actions. Since we have two variables for the Softmax function, two modified Q(a) action-value function were used. For action-value estimation, Eq. (6) was used and for action-value estimation for τ, Eq. (7) was used:

$$\mathrm{Q}_{k+1}(\tau, a) = \sum_{i=1}^{k+1} w_i r_i(\tau, a) = Q_k(\tau, a) + \alpha\left[r_{k+1}(\tau, a) - Q_k(\tau, a)\right] \tag{6}$$

$$\mathrm{Q}_{k+1}(\tau) = \sum_{i=1}^{k+1} w_i r_i(\tau) = Q_k(\tau) + \alpha\left[r_{k+1}(\tau) - Q_k(\tau)\right] \tag{7}$$

After formulation of the Softmax method, the meta-learning algorithm was applied to learn and update the parameters of Softmax method dynamically. The purpose of Meta-learning is to improve the learning system (Vilalta and Drissi 2002), so the meta-learning algorithm suggested in Sikora (2008) was used to update α and τ values. The update cycle for α was set at 1 and for τ was set at 5. The learning process for dynamic inventory threshold was set to begin after 10 days from the start of the simulation to ensure that the inventory threshold is updated based on enough information collected during previous 10 days. The updated meta-learning algorithm for this study is provided in Appendix 1.

Procurement amount is determined by the gap between inventory threshold and the available inventory of the first day within 15 days when the inventory goes below the threshold. The first day found is set to the due date of the RFQ, and the difference of the inventory threshold and the available component inventory of the day is set to RFQ amount.

For the price estimation, the same estimation method utilized in Pardoe and Stone (2009) was used and is shown in Eq. (8) since suppliers also determine the price using this formula

$$Price(d+i) = P_{base} - 0.5 \times P_{base} \times \frac{C_{available}(d+i)}{500 \times i} \tag{8}$$

where $C_{available}(d+i)$ is the available capacity of the supplier from the current day through day i and P_{base} is the base price, and 500 is nominal capacity. Nominal capacity is a type of conceptual average capacity of supplier used to determine the nominal price.

After finding the estimated price, the reserve price is determined as 115% of the estimated offer price. A PC's reserve price range is 0.75–1.25 times of the sum of the nominal price of all components used for the PC. Since our bidding strategy is to focus on the products having higher profit, our agent focuses on the products for which the reserve price range is between 1.0 and 1.25 times of the nominal price. Based on this assumption, the sum of the components' price, which is procurement cost, should be lower than the PC's price in order to make a profit. Hence, 115% of price of nominal price was used as the reserve price. The process of the creation of procurement RFQ is provided in Appendix 2.

Once the offers are received, the agent needs to select which ones to send as procurement orders. The offer acceptance strategy is to accept all offers, because the purpose of procurement is not to purchase components for a specific customer's order, but to fill the inventory threshold to satisfy the component amount requested for production in future 12 days. In addition, since the agent cannot guarantee that it may purchase components whenever it needs them, it is better to accept all offers by applying the concept of expected amount when procurement is possible.

3.2 Sales

The sales module is responsible for selling products to customers. Since the simulation uses auction for sales, sales strategy should determine whether to bid on a customer offer and at which price to bid. The basic bidding strategy of our model is based on the one in Pardoe and Stone (2009), but we modified it by adding new mechanisms, such as profit threshold for sending a sales offer and the sliding window protocol for determining the winning probability.

To secure the production resources for future offers, SCMaster creates a set of offers by merging actual sales RFQs and the expected sales RFQs into a set as suggested in Pardoe and Stone (2009). After collecting RFQs to consider, SCMaster selects an individual RFQ's bidding price creating the maximum expected profit by applying a greedy algorithm. The expected profit is calculated by the formula shown in Eq. (9):

$$\text{Expected profit} = \Pr\left(O_i | P_{bid_i}\right) \times \left(P_{bid_i} - C\right) \tag{9}$$

where O_i is the order that will be resulted from bid_i, Pr $(O_i|P_{bidi})$ is the winning probability of an offer on a given bidding price P_{bid_i}, P_{bid_i} is the bidding price for bid_i, and C is the cost of the product, which is the sum of purchasing prices of components for the product in the RFQ. To find the winning probability for the Eq (9), SCMaster uses the dynamic price bands approach suggested in Sikora and Sachdev (2008). Since dynamic price bands can provide finer granularity for popular parts in the agent's search space by consolidating the weaker price with coarser intervals,

we adopted the sliding window protocol. To control the price band, three constraints are used: min price interval size that can be split ($\delta_{\min}$), min policy vector value that can be split ($\nu_{\min}$), and max combined value of adjacent policy vector values that can be merged ($\nu_{\max}$). In this study, we used three values: $\delta_{\min} = 0.2$, $\nu_{\min} = 10$, and $\nu_{\max} = 0.1$.

Consider following sample price bands and their winning probabilities gained from the result of sale offers:

[760, 800]	[800, 840]	[840, 880]	[880, 920]	[920, 960]	[960, 1000]	[1000, 1040]	[1040, 1080]	[1080, 1120]	[1120, 1140]	[1140,1180]
0.0	0.0	0.0	0.0	0.25	0.25	0.5	0.0	0.0	0.0	0.0

The first two price bands, [760, 840] and [840, 880], could be merged since their combined probability, which is 0, is less than $\nu_{\max}$. At the same time, the price bands [920, 960] could be split because its winning probability 0.25 is greater than $\delta_{\min}$ and the interval of the price band ($|960–920| = 40$) is greater than $\nu_{\min}$. After first iteration, we will have the price bands as below:

[760, 840]	[840, 880]	[880, 920]	[920, 940]	[940, 960]	[960, 1000]	[1000, 1040]	[1040, 1080]	[1080, 1120]	[1120, 1140]	[1140,1180]
0.0	0.0	0.0	0.125	0.125	0.25	0.5	0.0	0.0	0.0	0.0

However, we still have some price bands to be merged or split. Hence, we will continue the iteration until we have no more price bands to be merged or split. In this example, four more iterations are required to completely merge and split the price bands. After each iteration price bands will be adjusted as below:

[760, 880]	[880, 920]	[920, 940]	[940, 960]	[960, 980]	[980, 1000]	[1000, 1040]	[1040, 1080]	[1080, 1120]	[1120, 1140]	[1140,1180]
0.0	0.0	0.125	0.125	0.125	0.125	0.5	0.0	0.0	0.0	0.0

[760, 920]	[920, 940]	[940, 960]	[960, 980]	[980, 1000]	[1000, 1020]	[1020, 1040]	[1040, 1080]	[1080, 1120]	[1120, 1140]	[1140,1180]
0.0	0.125	0.125	0.125	0.125	0.25	0.25	0.0	0.0	0.0	0.0

[760, 920]	[920, 940]	[940, 960]	[960, 980]	[980, 1000]	[1000, 1010]	[1010, 1020]	[1020, 1040]	[1040, 1120]	[1120, 1140]	[1140,1180]
0.0	0.125	0.125	0.125	0.125	0.125	0.125	0.25	0.0	0.0	0.0

[760, 920]	[920, 940]	[940, 960]	[960, 980]	[980, 1000]	[1000, 1010]	[1010, 1020]	[1020, 1030]	[1030, 1040]	[1040, 1140]	[1140,1180]
0.0	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.0	0.0

Note that the last two price bands cannot be combined further since our method uses a fixed number of price bands (11 price bands in the above example). At the beginning of the simulation, the initial probability for each price band is set to ($1/n$, n is the number of price band) and bands are updated every 6 days. The representation of Sliding Window Protocol is given in Fig. 1. Based on the winning probability, SCMaster identifies a RFQ having maximum increase of profit among all RFQs by applying the greedy algorithm as suggested in Pardoe and Stone (2009). One drawback of this strategy is that it might cause a high volume of unfilled orders. To prevent this drawback, we apply a mechanism called the profit threshold for sending sales offers. The profit threshold is a bidding guideline to focus on higher profit products. If the expected profit of the RFQ is equal to or greater than the profit threshold, SCMaster will send the sales offer for the RFQ, otherwise the bidding will not be sent. The profit threshold is determined heuristically and a fixed number 5.0 is selected. Note that since the goal of the

Let
x be a price band list of size n * 1,
[l_i , u_i] ∀ I = 1... n, be the n price band or intervals,
δ_{min} be the minimum price interval size that can split
v_{min} be the minimum policy vector value that can split, and
v_{max}, maximum combined value of two adjacent policy vector values that can be merged

Every day{
Accumulate win and lose information of each order;
}

Every 6 days after day 10{
Repeat{

Find two adjacent winning probabilities, x_i & x_{i+1} in the price band list
s. t. $x_i + x_{i+1} <= v_{max}$;

Find a value, x_j in the price band list to split
s. t. $x_j >= v_{min}$ and $| l_j - u_j | >= \delta_{min}$;

if and only if, ∃ x_i, x_i+1 and x_j, do the following:
combine the price band & winning probabilities *i* and *i+1*,
split the price band and winning probability *j* ;

} Until no more price bands can be combined or split;
}

***** Bold and underlined parts are added parts to the algorithm in*** (Sikora and Sachdev 2008) .

Fig. 1 Sliding window protocol

agents is to maximize their profits their actions (like the use of profit threshold) will drive customers of low profit products away. This is also observed quite frequently in the real world as companies climb the value chain by gravitating away from low profit products towards higher profit products and services. By combining and developing techniques, the updated bidding algorithm for this study is provided in Fig. 2.

3.3 Production scheduling

In our agent, shipment scheduling and production scheduling are conducted as a set of tasks since the two functions are directly related to each other. SCMaster first retrieves outstanding sales orders and allocates available product inventory to fulfill the outstanding sales orders for shipment. After scheduling the shipment, the agent schedules production for the remaining outstanding orders. SCMaster schedules the production by applying the simplified greedy search algorithm in the descending order of the profit of the sales orders. In the calculation of the profit, sales penalty amount is included as part of the cost. Hence, profit of an order is calculated as given in Eq. (10):

$$Profit_{order} = P_{unit,product} \times Qty_{order} - P_{reserve,order} \times r_{penalty,order} \times \min\left(5, d_{pastdue}\right) - C_{order} \times Qty_{order} \quad (10)$$

For each RFQ, compute both the winning probability and the expected profit as a function of price;

Set the bid for each RFQ to be just above the reserve price;

Repeat{
For each RFQ, find the bid lower than the current bid that produces the largest increase in profit per additional computer ordered (or per additional cycle required during periods of high factory utilization);

Choose the RFQ and bid that produce the largest increase;

If the profit of RFQ is greater than the profit threshold for sending sales{
Simulate schedule production for the partial order resulting from lowering the bid **considering the expected component inventory and factory cycle**;

If it cannot be scheduled, remove the RFQ from the list **and do not add to bid list**;

If the production is schedulable, but no further decrease in the bid will lead to an increase in profit, remove the RFQ from the list;

If the production is scheduledable and it is expected RFQ, remove the RFQ from the list;
}

} Until no RFQs are left in the list of RFQs to be considered;

Return the final bid for each RFQ;

*** Bold and underlined parts are added parts to the algorithm in* (Pardoe and Stone 2009).

Fig. 2 Bidding algorithm

where $Profit_{order}$ is the profit of an order, $P_{unit,product}$ is the unit price of the product in the order, Qty_{order} is the quantity of the order, $P_{reserve,order}$ is the reserve price of the order, $r_{penalty,order}$ is the penalty rate of the order, $d_{pastdue}$ is the day after due date of the order, and C_{order} is the cost of the order.

After finding the profit of each sales order, the agent starts production scheduling in the descending order of profit. Higher priority of using production resource is given to the orders that are past due. The orders that are past due are scheduled from the most recent possible production day while the orders that are not past due are scheduled from the previous day of the due date toward current day to reduce the production holding cost. The greedy production scheduling for this study is summarized in Fig. 3. As a final step of production scheduling, the agent adjusts the production schedule so that the remaining production resources in recent days can be used first.

4 Simulation environment

The simulation of this study was designed to provide the foundation for designing and evaluating our SCM agent's theoretical model. This simulation is motivated by TAC-SCM but we redefined many parts to provide a more realistic business environment. In our simulation multiple intelligent agents compete against each other to earn higher profits in the e-commerce personal computer (PC) market. The winner is the agent that makes the highest profit after multiple days of execution. Typical sequence of events are as follows: agents receive customers' sales order, purchase

```
Calculate the profit (=price - cost - penalty) of outstanding orders;

Sort outstanding order by the decreasing order of profit;

Repeat{
      If  the order is past due{
          schedule production from (current day + 2);
      }
      else{
        schedule production from due data of the order;
      }

} Until no more orders to consider
   or
   no more production resources( factory cycle or available components);
```

Fig. 3 Production scheduling algorithm

components from multiple suppliers, store the components, assemble components into PCs at their factory, store the completed products (PCs) in storage, and ship the PCs to fulfill customer orders.

A transaction of a sales order begins with sales request for quote (RFQs) from customers to all agents in the simulation. Agents then submit a sales offer with a bidding price to win the order, and each customer chooses the bid with the lowest price. To assemble products, agents need to procure computer components from suppliers. When an agent wants to purchase a component, it issues a procure RFQ to multiple suppliers. Each supplier then decides the price and quantity that will be offered to each agent and sends the procurement offers to the agents. As a final step of procurement, agents decide whether to accept the offer. Delivered components are stored in components storage of trading agents, and the cost of components in storage is charged until they are removed for production.

After receiving the components, agents assemble PCs to fulfill the customers' orders. Agents need to therefore establish plans for production and shipment. The factory assembles PCs as planned in the production schedule using component inventory and factory cycle within the boundaries of available resources. Finished PCs are shifted into product storage on the same day. Once PCs are stored in storage, these products are ready to be shipped to customers according to the shipping schedule to satisfy customer orders. Unlike component delivery, the delivery for PCs should meet the entire amount in sales RFQs. If it is not met, the agent is charged for late delivery entitled sales penalty.

4.1 Suppliers and procurement process

Agents need to acquire components from suppliers to assemble PCs. The list of components and Bill of Material are included in Appendices 3 and 4. Suppliers interact with agents for procurement of raw materials, and computer components. Suppliers' daily work cycle begins with receiving procurement RFQs from agents. The number of

RFQs to a supplier per component is limited to five, so an agent can send a total of ten RFQs to a supplier since each supplier supplies only two types of components. Simulation parameters are defined in Appendix 5. When purchasing components from suppliers, the purchase price is treated as a procurement cost for agents. In the simulation of this study, those price distributions are pre-defined for each day and each component, and each supplier has a pre-defined distribution for the supply amount for each component on each day.

Once procurement RFQs are received, each supplier begins to consider the offer price and offer amount to issue procurement offers. First, each supplier groups the RFQs by the due date and sorts the RFQs in a group by the descending order of the reputations of agents. A priority mechanism, Reputation, is applied to provide an advantage to more loyal agents, and it is adopted from the specification of TAC-SCM (Collins et al. 2006). The reputation is defined as in Eqs. (11) and (12)

$$\varsigma_a = \frac{quantityPurchase_a}{quantityOffered_a} \tag{11}$$

where $quantityPurchase_a$ is the sum of the quantities in all the orders by agent a, and $quantityOffered_a$ is the sum of quantities in all the offers issued by the supplier to the agent a. By considering ς_a, suppliers may measure which agent is more likely to purchase their components for the given procurement offers. With the value of ς_a, suppliers will calculate the final reputation value as follows:

$$rep_a = \frac{\min\left(apr, \varsigma_a\right)}{apr} \tag{12}$$

where apr is the acceptable purchase ratio. This concept is used to prevent excessive punishment for not purchasing the component because agents should not be punished for purchasing from only one supplier (Collins et al. 2006). Then, each supplier sets up an offer price for the agent having the highest reputation by using a pre-defined supply price distribution. As reputation lowers, suppliers increase the offer price to give an advantage to the agents having higher reputation. Note that the reputation metric in our design is used as a proxy for the overall quality. In this step, suppliers only consider the reserve price constraint. In the next step, each supplier considers whether it can supply the entire quantity for each RFQ. If the supplier has enough capacity, it will provide an offer having the same offer quantity to the quantity in the corresponding procurement RFQ. However, if the supplier does not have enough available capacity for all RFQs, this situation is defined as a conflict, and the supplier creates partial offers of which quantities are distributed among the group of RFQs having the same due date. Suppliers use the following Eq. (13) suggested in specification of Collins et al. (2006) to distribute the supply amount to RFQs

$$\forall_{\mathrm{r}} \in R_{conflict}, q_{\mathrm{r}}^{p} = q_r^{'} - C^{conflict} \frac{q_r^{'}\left(1/reputation_r^m\right)}{\sum_{r' \in R_{conflict}} q_{r'}^{'}\left(1/reputation_r^m\right)} \tag{13}$$

where $R_{conflict}$, is the group of RFQ of which sum quantity is not met by the supplier, q_r^p is new price of RFQ r for product (=component) p, q_r' is the RFQ quantity, $C^{conflict}$ is conflict amount, and $reputation_r$ is the reputation of the agent that issued RFQ r. The procurement offers created in this step are called partial offers. As the final step of creation of procurement RFQs, suppliers create new offers that can complement the amount not filled by partial offers by finding the earliest possible day when each supplier can fill the deficit amount. For partial offers and earliest possible offers, suppliers offer the same offer price decided in the second step to continue to provide an advantage to more loyal agents. After processing these steps, suppliers send all the procurement offers to the agents.

4.2 Customer and sales process

To sell the PCs and make the revenue, agents must relate to customers. Thus, the sales process begins with the customer's sales RFQ. According to Borghetti and Sodomka (2006), one of the challenges in designing a useful benchmarking environment for TAC-SCM is to manage the randomness and repeatability of the simulations. Thus, in this simulation, customers use a pre-defined number of daily RFQs and a daily demand amount for each PC type to manage the randomness and repeatability when creating sale RFQs. To create customer demand, we adopted the assumption of the Poisson distribution as in specification of Collins et al. (2006). Hence, customers first calculate the number of RFQs using Poisson distribution as follows:

$$N = Poisson(Q) \tag{14}$$

where N is the number of sales RFQs of the day, and Q is the average of Poisson distribution. However, only the number of RFQs for a day is defined by this distribution in this step. In the second step, customers select the due date of each RFQ randomly between three and 12 days after the current day for each RFQ. After selecting the due date, customers choose the number of products randomly, between 3 and 12.

In TAC-SCM, only a random function is used to create the number of products in a RFQ, and it caused different levels of customer demand in different execution of simulation. We modified the method of creating the customer demand for each RFQ. Since the demand should be same in different executions of simulation, the creation of the number of products follows the pre-defined demand distribution for each day and the amount of demand issued by this random function is subtracted from the demand distribution of the day. To prevent excessive demand, customers use Eq. (15) to decide the demand amount of a day:

$$Qty_d^p = MIN\left(random(3, 12), D_d^p\right) \tag{15}$$

where Qty_d^p is the order quantity on day d, and D_d^p is the demand amount of product p on day d. Through these steps, customers create a random number of sales RFQs on a randomly selected day with a random amount while keeping the entire demand amount as pre-defined.

After receiving the sale offers from agents, customers begin to choose the final orders to give to agents. Considering the reserve price constraint, the customers select a sales offer with the lowest bidding price. If multiple agents bid the same lowest price, customer chooses one agent randomly.

4.3 Competing agents

The key players of this simulation are agents. In our simulation, agents compete against each other to acquire suppliers' components and to win customers' sales. Individual agent has its own factory that can assemble any type of PC, and production is conducted in the factory. Factory has a limited daily production capacity, which is measured by the production cycle, and the production is scheduled 1 day before by the agent. After the production, completed PCs are automatically moved to product storage on the same day. The component holding and product holding costs are calculated and charged at the end of every day. For the shipment of products to customers, agents send a shipping schedule to product storage 1 day before the shipment day. If production inventory does not meet the entire quantity in an order, the shipment of the order does not occur. Based on the basic rules for agents, two competing agents were created by adopting main strategies of two best agents in TAC-SCM literature.

The first agent is entitled Agent-D. Agent-D was created to simulate the DeepMaize agent (Kiekintveld et al. 2006). For creating procurement RFQ, Agent-D uses the concept of reference inventory trajectory and the baseline buffer level. The concept of baseline buffer level is the inventory threshold that Agent-D wants to fill to cope with change in demand. Reference inventory trajectory, the sum of the components required from outstanding customer orders and the baseline buffer level, is the concept used for estimated inventory. Agent-D creates RFQs to fill the gap between trajectory inventory and baseline buffer level. When deciding which RFQs to accept, Agent-D tries to maximize the profit.

For each sales offer, Agent-D finds the bidding price that maximizes the increase of expected unit profit of a sales offer. The unit profit of the offer is determined by subtracting the expected cost from the bidding price of the sales offer. For the expected cost, the concept of effective cost in Kiekintveld et al. (2004) is used. For the production and shipment procedure, Agent-D uses the algorithm suggested in Kiekintveld et al. (2006). As the first step, Agent-D completes shipment scheduling with current available PC inventory. After establishing shipment plans, Agent-D establishes production plans for unfilled orders using current available factory cycle and estimated component inventory. In the second step, Agent-D calculates the expected profits for all possible productions. After this calculation, Agent-D establishes the production plan in the descending order of the profit.

The second agent, Agent-T, was created to simulate the TacTex agent in Pardoe and Stone (2009). For the creation of procurement RFQ, Agent-T uses the concept of inventory threshold. This threshold is fixed at 800 for non-CPU components and 400 for CPU components. To decide the amounts and due dates of procurement RFQs, Agent-T finds the first day when the components inventory goes below the

threshold and calculates the gap between the estimated inventory and the inventory threshold. Based on this information, Agent-T creates procurement RFQs. Once procurement offers arrive, the agent accepts all the offers since Agent-T assumes that the offer price will be lower than the reserve price of procurement RFQ and the reserve price is set based on its prediction. For each sales offer, Agent-T uses a greedy algorithm that searches for the bidding price that maximizes the increase of expected unit profit of a sales offer. Agent-T ships PCs according to shipment schedules using current available PC inventory at first and it establishes production plans using a greedy algorithm for unfilled orders, considering current available factory cycles and estimated component inventory.

4.4 Modification of simulation

In this simulation, we used the same type of PC and components suggested in Collins et al. (2006) and these are listed in Appendices 3 and 4. However, since the purpose of this study is not to take part in TAC-SCM competition, but to use it as a test framework, several simulation rules were modified to implement the more realistic business environment. In our simulation no special consideration is given for the start or end of the simulation. According to TAC-SCM literature, agents exploit starting conditions to enhance their performance. For example, Mertacor in Chatzidimitriou et al. (2008) purchases twice the component inventory threshold from suppliers on the first day and switches it to normal-mode on the second game day. However, in this study, none of the agents include any algorithms for the initial status of the simulation or the end of the simulation. The reason for not considering the concept of start, end, and data from previous simulations is that there are no such concepts in real world business.

Another modification of the simulation is the definition of demand and supply distribution. Most agents, such as DeepMaize in Kiekintveld et al. (2006) and TacTex in Pardoe and Stone (2009) for TAC-SCM, use prediction data from the previous round to improve their performance. In this study, the distribution of the customer demand and the supply distributions are shared with competing agents to allow them to have high prediction accuracy. The parameters, components, and product are listed in Appendix 5.

5 Experiments and results

For this study, the simulation environment and our agent are implemented using Java 1.7, and a relational data base was used for data repository of the agent. As described in Sect. 3.2, our model is implemented by modifying Agent-T. This study executed three different types of simulations to evaluate the performance and to understand the characteristics of strategies used for SCMaster. The three setups include: (1) Non-Competitive Single Agent; (2) Competitive Homogeneous Agent; and (3) Heterogeneous Agent. We conducted each type of simulation ten times under the uniform demand distribution with a mean of 200 and a standard deviation of 50.

The performance of the agents is measured by their profit. Since this simulation assumes realistic business environment, agents do not utilize strategies taking into account the beginning or the end of simulation. At the end of day 250, we stopped the simulation and measured the profits, together with current revenue and costs. The results of the simulation are analyzed with non-parametric statistical tests. For the comparison of two agents, we used Mann–Whitney U Test and Wilcoxon's Signed Ranks Test; and, for the comparison of revenues of three agents, Kruskal–Wallis Test and Friedman Test were used.

As SCMaster showed the best performance among three agents in (1) non-competitive single agent and (3) heterogeneous agent simulations, we applied three demand distributions, mean = 200 and SD = 10, mean = 200 and SD = 30, and mean = 200 and SD = 100, to evaluate the sensitivity of our model.

5.1 Non-competitive single agent

This simulation is designed to study the baseline performance of each type of agent as they are executed in a simulation as the sole agent. It is expected to find the differences of baseline performances among different types of agents. The experiment setting is depicted in Fig. 4.

To measure the differences in performance, each simulation with a different type of agent was executed ten times under the uniform customer demand distributions with a mean 200 and a standard deviation 50. The results are presented in Table 1. The Kruskal–Wallis test was applied to see the statistical differences of three agents' base performance. The results show that agents have statistically significant performance differences among them (cutoff value = 0.05). The average profit of two agents was compared using the Mann–Whitney U test. SCMaster showed statistically better performance than both Agent-D (Sig. = 0.015) and Agent-T (Sig. = 0.000). Agent-D showed better performance than Agent-T (Sig. = 0.000). Thus, SCMaster has the best, Agent-D has the second best, and Agent-T has the third best performance under a uniform distribution with a mean of 200 and a standard deviation of 50.

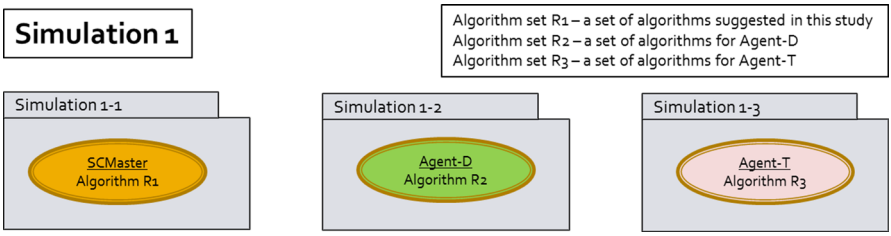

Fig. 4 Non-competitive single agent

Table 1 Result of simulation 1

Distribution	Test method	Agent-D (simulation 1–2)	Agent-T (simulation 1–3)	SCMaster (simulation 1–1)
Uniform (μ = 200; σ = 50)	Avg. profit	5,679,443	2,493,413	**7,854,765**
	SD	1,961,597.63	150,620.96	323,643.41
	Kruskal–Wallis test (2-tailed)	Sig. = .000		
	Mann–Whitney U test (2 tailed)	Agent-D	Sig. = .000	Sig. = .015
		Agent-T		Sig. = .000

The best results are highlighted in bold

5.2 Competitive homogeneous agents

In this simulation, two instances of the same agent type competed against each other. The purpose of this simulation was to study the performance change when the same type of strategy competes in a simulation. Each simulation was conducted ten times and a non-parametric statistic test was applied. The experiment setting is depicted in Fig. 5.

Since two instances of the same type of agents were executed in each simulation, the average performance of each agent type was calculated from 20 samples created by two instances. Results of the simulation are presented in Table 2. Under the demand distribution with a mean of 200 and a standard deviation of 50, the existence of differences of three profits was tested by Kruskal–Wallis Test. The level of significance is 0.000, so the null hypothesis was rejected and it is concluded that there exists statistically significant differences among the three profits. The Mann–Whitney U Test was then used to test the difference between two groups in all tests. When comparing SCMaster and Agent-D, SCMaster shows better performance than Agent-D (Sig. = 0.020), but SCMaster showed worse performance than Agent-T (Sig. = 0.000). Agent-D also showed worse performance than Agent-T in this distribution (Sig. = 0.000). As a result, Agent-T showed the best performance, SCMaster the second best, and Agent-D the lowest performance.

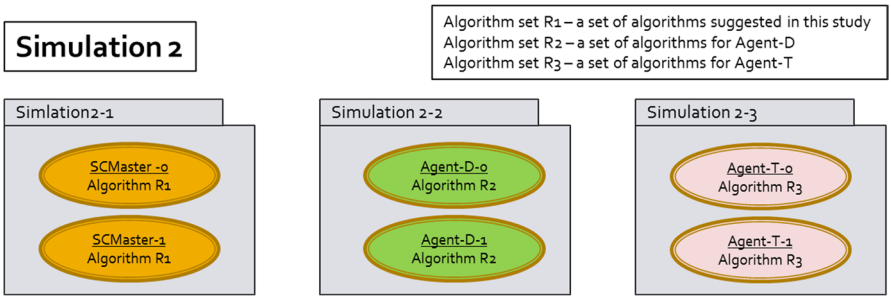

Fig. 5 Competitive homogeneous agents

Table 2 Result of simulation 2

Distribution	Test method	Agent-D (simulation 2–2)	Agent-T (simulation 2–3)	SCMaster (simulation 2–1)
Uniform (μ = 200; σ = 50)	Avg. profit	4,713,611	**8,057,868**	6,169,615
	SD	2,968,098.30	71,507.27	570,996.18
	Kruskal–Wallis test (2-tailed)	Sig. = .000		
	Mann–Whitney U test (2 tailed)	Agent-D	Sig. = .000	Sig. = .020
		Agent-T		Sig. = .000

The best results are highlighted in bold

5.3 Competitive heterogeneous agents

In this simulation, two or three agents of different types compete against each other in the same simulation. The purpose of simulation 3 is to study the performance of SCMaster when it competes against two different types of agents. For the statistical analysis, each simulation was executed ten times. The Friedman Test was used for the comparison of three agents' performance in this simulation, and Wilcoxon's Matched-Pair Signed Ranks Test was used for the comparison of two agents' performance in all simulations. The experiment setting is depicted in Fig. 6 and the results of simulation are presented in Table 3.

As before, the simulation was conducted under the same uniform demand distribution with a mean of 200 and a standard deviation of 50. The result was tested with Wilcoxon's signed test, and it is concluded that SCMaster performed better against Agent-D (Sig. = 0.005) and against Agent-T (Sig. = 0.005).

Under the same demand distribution, the simulation 3–3 tested the existence of the differences among the profits from agents with the Friedman Test, and the level of significance was 0.000. Hence, one can conclude that there are statistically significant differences among the performances of agents when they are all competing

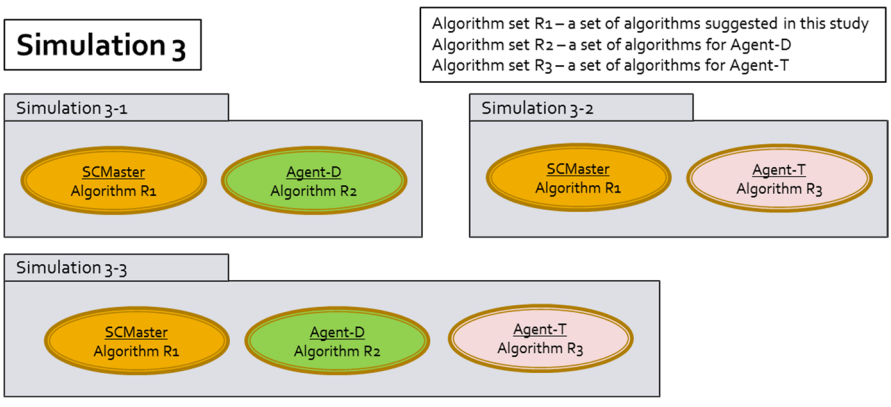

Fig. 6 Competitive heterogeneous agents

Table 3 The result of simulation 3

Distribution	Test method	Simulation 3–1		Simulation 3–2		Simulation 3–3		
		Agent-D	SCMaster	Agent-T	SCMaster	Agent-D	Agent-T	SCMaster
Uniform ($\mu = 200$; $\sigma = 50$)	Avg. profit	2,804,434	**7,508,036**	4,435,694	**7,494,287**	−6,185,019	4,113,609	**7,552,459**
	SD	1,968,954.50	254,793.03	227,682.38	439,093.35	3,830,219.95	661,465.37	304,491.74
	Friedman test (2-tailed)	N/A				Sig. = .000		
	Wilcoxon's signed ranks test (2-tailed)	Sig. = .005		Sig. = .005		Agent-D	Sig. = .005	Sig. = .005
						Agent-T		Sig. = .005

The best results are highlighted in bold

in the same simulation. To test the differences between two agents, we applied Wilcoxon's signed rank test. In the comparison of SCMaster and Agent-D, SCMaster has better performance than Agent-D when three agents are executed in a simulation (Sig. = 0.005). SCMaster and Agent-T was also compared, and the level of significance was 0.005. In the comparison of Agent-D and Agent-T, Agent-T showed a better performance than Agent-T (Sig. = 0.005). As a result, one can conclude that when all three agents compete against each other, SCMaster has the best performance, Agent-T has the second best performance, and Agent-D has the worst performance.

5.4 Sensitivity analysis

Although we used the set of simulation parameters mentioned in Appendix 5 we did perform a sensitivity analysis by varying these parameters. Even though the results changed in terms of the actual profit earned by the agents, the relative performance between and amongst the agents remained the same in all the simulations. In Sects. 4.1–4.3, we measured the performance of three agents under a uniform demand distribution with a mean of 200 and a standard deviation of 50. However, one required functionality of the agent is the adaptiveness to environment. Hence, we also conducted sensitivity analysis by executing the entire set of experiments (simulation 1, 2, and 3) under three different customer demand distributions: a uniform distribution with a mean of 200 and a standard deviation of 10; a uniform distribution with a mean of 200 and a standard deviation of 30; and a uniform distribution with a mean 200 and a standard deviation 100.

The results of simulation 1 under three distributions are provided in Table 4. SCMaster provided the highest performance among three agents under three customer demand distribution, and the order of the performance of other agents remained the same under all distributions.

The results of simulation 2 and simulation 3 are provided in Tables 5 and 6 respectively. The results for both simulations, simulation 2 and 3, are also similar to the result under the distribution with a mean of 200 and a standard deviation of 50. In simulation 2, Agent-T showed the best performance, SCMaster showed the second best performance, followed by Agent-D under three demand distributions. In simulation 3–1 and 3–2, SCMaster has better performance than Agent-D

Table 4 Result of simulation 1 for sensitivity analysis

Distribution	Test method	Agent-D (simulation 1–2)	Agent-T (simulation 1–3)	SCMaster (simulation 1–1)
Uniform ($\mu = 200$; $\sigma = 10$)	Avg. profit	6,475,470	2,275,473	**7,720,129**
	SD	1,732,693.35	233,450.35	370,376.85
Uniform ($\mu = 200$; $\sigma = 30$)	Avg. profit	6,085,697	2,582,944	**7,937,590**
	SD	1,532,089.50	248,750.58	428,386.26
Uniform ($\mu = 200$; $\sigma = 100$)	Avg. profit	5,967,857	3,790,433	**8,212,886**
	SD	1,905,506.75	330,178.94	354,392.44

The best results are highlighted in bold

Table 5 Result of simulation 2 for sensitivity analysis

Distribution	Test method	Agent-D (simulation 2–2)	Agent-T (simulation 2–3)	SCMaster (simulation 2–1)
Uniform ($\mu = 200$; $\sigma = 10$)	Avg. profit	4,027,230	**8,133,712**	6,115,102
	SD	1,822,969.01	99,824.15	790,472.70
Uniform ($\mu = 200$; $\sigma = 30$)	Avg. profit	5,008,354	**8,095,446**	6,151,833
	SD	2,620,000.87	78,993.15	725,916.03
Uniform ($\mu = 200$; $\sigma = 100$)	Avg. profit	5,509,744	**7,747,697**	6,074,656
	SD	2,033,183.01	95,205.99	632,975.79

The best results are highlighted in bold

and Agent-T respectively, and SCMaster showed the best performance among three agents in the simulation 3–3.

6 Discussion

In the previous sections, various market situations were simulated. The results of those experiments can provide an understanding of the characteristics of SCMaster's mechanism and other agents.

In simulation 1, the baseline performance of three agents was measured, and the result showed that, in any demand distribution, SCMaster has the best performance, Agent-D has the second best, and Agent-T has the lowest performance. For further analysis, we select one agent of each type and present their detailed performance figures. As can be seen from Table 7, the three types of costs showed differences among agents. The difference of profit came from the balance between revenue and sales penalty. When considering only revenue, Agent-D and Agent-T had more revenue than SCMaster. However, when looking at the sales penalty, we find that Agent-T and Agent-D paid higher penalty than SCMaster did.

As described in the simulation rules, sales penalty is caused from outstanding orders. A high sales penalty implies that the agent does not have enough products to deliver. In other words, the production amount is not enough to satisfy all customer orders. However, the average factory utilization of Agent-T was 1905 cycles (=97.5%), for Agent-D it was 1969 cycles (=98.4%), and for SCMaster it was 1946 cycles (=97.3%). According to the factory unitization one may conclude that the component supply is enough for production, and the agents are fully utilizing the factory cycle. The high ratio of the actual production to the production schedule supports this (SCMaster = 0.99, Agent-D = 0.99, Agent-T = 0.99). To find the source of the difference in performance, the number of sales orders for each agent was compared: SCMaster had 3582 orders, Agent-D had 4328 orders, and Agent-T had 10,859 orders. From these figures, Agent-T and Agent-D had more outstanding orders than SCMaster had. All agents simulate the production scheduling based on the partial offer before sending the sales offers. If agents cannot schedule the partial offer amount in the simulation, they do not send the sales offer. Note that partial

Table 6 The result of simulation 3 for sensitivity analysis

Distribution	Test method	Simulation 3–1		Simulation 3–2		Simulation 3–3		
		Agent-D	SCMaster	Agent-T	SCMaster	Agent-D	Agent-T	SCMaster
Uniform ($\mu = 200$; $\sigma = 10$)	Avg. profit	4,442,391	**7,760,469**	4,398,388	**7,645,224**	−5,074,865	3,810,187	**7,525,807**
	SD	2,001,522.68	324,569.41	216,113.87	343,271.33	2,651,584.44	690,593.53	528,312.95
Uniform ($\mu = 200$; $\sigma = 30$)	Avg. profit	4,271,035	**7,903,478**	4,365,773	**7,688,071**	−5,756,916	3,438,389	**7,506,421**
	SD	2,332,246.67	314,988.03	363,332.69	227,147.23	2,975,836.89	1,485,985.01	435,275.77
Uniform ($\mu = 200$; $\sigma = 100$)	Avg. profit	3,750,077	**8,156,249**	4,855,926	**8,037,760**	−6,105,920	4,318,848	**7,822,662**
	SD	2,525,082.47	402,246.26	228,041.59	505,412.31	3,201,842.36	313,649.94	305,292.11

The best results are highlighted in bold

Table 7 Summary of performance of three agents in simulation 1

	Revenue	Cost**					Profit (ratio)
		Product holding cost (efficiency)	Component holding cost (efficiency)	Procurement cost (efficiency)	Sales penalty (efficiency)	Total cost (efficiency)	
Agent-D	107,501,700	114,201 (0.001)	349,805 (0.003)	98,300,135 (0.914)	2,104,727 (0.020)	100,868,869 (0.938)	6,632,831 (0.062)
Agent-T	102,194,450	116,200 (0.001)	460,585 (0.005)	91,221,650 (0.893)	6,042,853 (0.059)	97,841,288 (0.957)	4,353,162 (0.043)
SCMaster	98,850,854	168,052 (0.002)	423,928 (0.004)	88,912,275 (0.899)	1,657,212 (0.017)	91,161,466 (0.922)	**7,689,388 (0.078)**

The best results are highlighted in bold

**Cost efficiency is calculated by the formula cost efficiency = cost/revenue, so the lower ratio implies better performance

offer is a weighted average of amount of all offers using winning probability of each bid, so it is less than the sum of the amount in all RFQs. In the case of Agent-T, it sent all the offers that passed the production scheduling simulation, but won more orders than expected because simulation 1 has no competitors. In the case of the SCMaster, the agent has one more filter for sending a sales offer, which is the profit threshold. Since the strategy of SCMaster is to focus on higher profit products even if the offer price is lower, this filter reduced the number of offers and, as a result, reduced the number of outstanding orders. In the case of Agent-D, since it uses a hybrid approach in sending the sales offer (it uses a lower profit threshold for sending sales offers), its number of winning orders is also in the middle and, accordingly, the number of outstanding orders is in the middle as well. As seen in the analysis above, simulation 1 provided an insight on how each agent worked differently by eliminating the effect caused by the competition from other agents.

The result of simulation 2 can be explained based on the understanding we acquired from the analysis of simulation 1. A notable change in the results is that Agent-T showed the best performance among three agents. The bidding strategy of Agent-T relies on its expected winning probability. The advantage of this strategy is that the agent can maximize its revenue as long as the production resources support enough products to fulfill the orders. However, if the agent wins more orders than it estimated and the production resource cannot support those orders, the outstanding orders will be turned into sale penalties, and, as a result the cost to the agent will increase. In simulation 1, Agent-T won more orders than it could support since the agent monopolized the market, but, in the simulation 2, the conflict between the bidding strategies of two instances of Agent-T reduced the number of winning orders. The reduced winning orders also reduced the outstanding orders, and as a result of the reduced sale penalties improved the overall performance of the agent. The performance improvement of Agent-T due to the reduction of sales penalty is shown in Table 8. In simulation 2, the competition between agents reduced Agent-T's orders from 10,859 in simulation 1–6318 in simulation 2 and, as a result, it reduced outstanding orders while the agent used 97.5% of factory cycles. The profit of Agent-D was similarly reduced in simulation 2. The number of orders was reduced from 4328 to 4042 as a result of the competition and the factory used 98.0% of its cycles. The revenue remained almost the same (107 M), but the sales penalty was reduced. However, the procurement cost increased from 98.3 M in simulation 1–100 M in simulation 2. Excessive procurement cost might reduce profit. However, since the procurement cost is proportional to revenue and agents must purchase components to make revenue, it is not reasonable to use the raw procurement cost to measure procurement efficiency. Hence, the concept of efficiency ratio (procurement cost per revenue) was adopted and the procurement performances were compared. Note that the lower ratio implies better performance. We report the efficiency ratios in Tables 7, 8, 9, 10 and 11. We also report the profit ratios (profit per revenue) and the higher the ratio the better. In simulation 1, the ratio was 0.91 and it was 0.93 in simulation 2. This indicated that the procurement efficiency deteriorated in the presence of competition. As a result, the difference of 2 M in procurement cost was responsible for the increase of total cost, and the increased cost reduced the profit by 2 M. SCMaster also had reduced profit in simulation 2. Since SCMaster's orders

Table 8 Summary of performance of three agents in simulation 2

	Revenue	Cost**					Profit (ratio)
		Product holding cost (efficiency)	Component holding cost (efficiency)	Procurement cost (efficiency)	Sales penalty (efficiency)	Total cost (efficiency)	
Agent-D	107,329,660	116,782 (0.001)	356,120 (0.003)	100,088,395 (0.933)	1,861,181 (0.017)	102,422,477 (0.954)	4,907,183 (0.046)
Agent-T	102,311,480	120,044 (0.001)	406,709 (0.004)	90,206,625 (0.882)	3,501,391 (0.034)	94,234,769 (0.921)	**8,076,711 (0.079)**
SCMaster	71,120,890	186,904 (0.003)	226,392 (0.003)	64,152,900 (0.902)	398,719 (0.006)	64,964,914 (0.913)	6,155,976 (0.087)

The best results are highlighted in bold

**Cost efficiency is calculated by the formula cost efficiency = cost/revenue, so the lower ratio implies better performance

Table 9 Summary of performance of two agents in simulation 3–1

	Revenue	Cost**					Profit (ratio)
		Product holding cost (efficiency)	Component holding cost (efficiency)	Procurement cost (efficiency)	Sales penalty (efficiency)	Total cost (efficiency)	
Agent-D	104,476,150	116,402 (0.001)	352,096 (0.003)	97,977,200 (0.938)	1,841,372 (0.018)	100,287,071 (0.960)	4,189,079 (0.040)
SCMaster	98,626,474	142,230 (0.001)	396,037 (0.004)	88,130,125 (0.894)	1,973,074 (0.020)	90,641,465 (0.919)	**7,985,009 (0.081)**

The best results are highlighted in bold

**Cost efficiency is calculated by the formula cost efficiency = cost/revenue, so the lower ratio implies better performance

Table 10 Summary of performance of two agents in simulation 3–2

	Revenue	Cost**					Profit (ratio)
		Product holding cost (efficiency)	Component holding cost (efficiency)	Procurement cost (efficiency)	Sales penalty (efficiency)	Total cost (efficiency)	
Agent-T	100,945,820	117,280 (0.001)	409,588 (0.004)	91,246,725 (0.904)	4,256,666 (0.042)	96,030,260 (0.951)	4,915,560 (0.049)
SCMaster	98,302,664	136,337 (0.001)	346,294 (0.004)	87,423,875 (0.889)	1,546,267 (0.016)	89,452,773 (0.910)	**8,849,891 (0.090)**

The best results are highlighted in bold

**Cost efficiency is calculated by the formula cost efficiency = cost/revenue, so the lower ratio implies better performance

Table 11 Summary of performance of three agents in simulation 3–3

	Revenue	Cost**					Profit (ratio)
		Product holding cost (efficiency)	Component holding cost (efficiency)	Procurement cost (efficiency)	Sales penalty (efficiency)	Total cost (efficiency)	
Agent-D	89,479,610	111,665 (0.001)	406,605 (0.005)	94,606,405 (1.057)	1,229,445 (0.014)	96,354,120 (1.077)	−6,874,510 (−0.077)
Agent-T	100,266,130	117,433 (0.001)	416,745 (0.004)	91,091,625 (0.908)	4,076,065 (0.041)	95,701,869 (0.954)	4,564,261 (0.046)
SCMaster	96,546,595	127,580 (0.001)	388,688 (0.004)	86,446,700 (0.895)	1,728,681 (0.018)	88,691,649 (0.919)	**7,854,946 (0.081)**

The best results are highlighted in bold

**Cost efficiency is calculated by the formula cost efficiency = cost/revenue, so the lower ratio implies better performance

were reduced from 3582 to 2546 due to the competition, factory utilization also was reduced to 69.8%. Even if SCMaster had enough factory cycles and component inventory, the profit of the agent was reduced in simulation 2 because of the lower number of sales orders gained. However, since the component holding cost, procurement costs, and the sales penalty were also decreased together, SCMaster maintained a larger profit than Agent-D in spite of lower revenue than that of Agent-D. This result indicates that when only one type of agent competes against each other in a market, Agent-T can be a better choice than SCMaster or Agent-D in terms of absolute profit. However, SCMaster still gives the best return in terms of the profit ratio reported.

In simulation 3, agents compete against other types of agents with different strategies. This simulation is the most similar to the market composition where trading agents work. The result of simulation 3–1 is presented in Table 9. When comparing SCMaster and Agent-D in simulation 3–1, it is found that the procurement cost was critical. The revenue of Agent-D was larger than that of SCMaster. All costs except the procurement cost are almost the same. Factory utilization was also almost the same: SCMaster used 97.0% and Agent-D used 97.3%. However, Agent-D paid about 97 M in procurement costs while SCMaster paid 88 M. SCMaster's procurement efficiency was 0.89 while that of Agent-D was 0.93. This result implies that SCMaster has better procurement performance than Agent-D. This difference is caused by better inventory management mechanisms. In the case of the SCMaster, it uses a dynamic inventory threshold, so the agent can prevent excessive inventory by adjusting the threshold dynamically. However, since Agent-D uses a fixed inventory threshold, it does not control the inventory flexibly. This difference caused the difference of 3.79 M in profit despite the higher sales revenue. The result of Agent-D in simulations 2 and 3–1 implies that Agent-D's procurement policy has a tendency to lead to excessive purchase of components, and it is less efficient than that for SCMaster.

In simulation 3–2, the performance of SCMaster and Agent-T are compared. The summarized results are presented in Table 10. The procurement efficiency ratio of SCMaster was 0.89, and that of Agent-T was 0.90, which caused 1.1 M of profit difference. This difference is caused by the inventory management mechanisms, as discussed above. Another weakness of Agent-T is the high penalty amount. Both agents use more than 95% of the factory cycle on average (Agent-T = 1945 cycles/day = 97.2%, SCMaster 1919.9 cycles/day = 95.9%), but Agent-T won 7940 orders and SCMaster won only 3447 orders. This implies that Agent-T has too many orders to process, and this excessive orders resulted in a high sales penalty.

In simulation 3–3, three agents were executed together in a simulation. The result is summarized in Table 11. Agent-D showed negative profit. This was caused by the low procurement efficiency and pool bidding performance. The bidding success rate of Agent-D was 0.45 and the average order prices were lower than those of Agent-T in most product types and lower than those of SCMaster in all product types (See Appendix 6). In previous simulations where two agents competed, Agent-D had a chance to win the high revenue (= high bidding price) orders, but, in simulation 3 where three agents competed, Agent-D lost that chance due to its lower bidding performance. In addition, Agent-D was not able to

recover from the low bidding performance due to its low procurement efficiency. Hence, Agent-D's overall low performance is caused by the combination of low bidding prices and low procurement efficiency. Agent-T showed a similar pattern in revenue, cost, and sales penalty with those in previous simulations. Agent-T showed the highest sales revenue and sales penalty due to the characteristics of its bidding strategy. The procurement efficiency was also similar, which was 0.91, to that in simulation 3–2. SCMaster also showed a similar pattern in revenue, cost, and sales penalty with those in previous simulations. The procurement efficiency was also similar, which was 0.90, to that in simulation 3–2. However, in this simulation, SCMaster's sales revenue was less than that of Agent-T, but its sales penalty was also less than that of Agent-T. Therefore, SCMaster's better performance is because of higher bidding performance and higher procurement efficiency.

As seen in Table 12, SCMaster showed high bidding performance. Its bidding success rate is 97% even if the number of offers is smaller than those of other agents, and its order prices in all product types are higher than those of Agent-T and Agent-D (see Appendix 6). Because of this high success rate and order prices, SCMaster could make enough revenue with a smaller sales penalty at the same time. In addition to sales efficiency, SCMaster also showed better performance on procurement.

As seen in the results of simulation 3–3, the critical factor of success of the SCM agent is the coordination of sales and procurement. In the case of Agent-D, its bidding performance is lowest among three agents, but it purchased too many components. Therefore, even if Agent-D paid the least sale penalty its profit was in deficit due to the imbalance between procurement and sales. While showing the highest sales efficiency, Agent-T also failed in finding its balance between sales and procurement. Although it procured more components than SCMaster to support its high sales volume, Agent-T lost its balance due to the lack of enough production resources. However, SCMaster found the right balance between sales and procurement. SCMaster was not the best agent in sales among the three, but it also reduced procurement amount and sales penalty. This implies that the sales of SCMaster were more efficient than those of other agents in the perspective of resources management and that procurement supports the production efficiently, preventing the excessive procurement. This improved performance of SCMaster shows that SCMaster's model provided techniques not only for the improvement of individual SCM function, but also for the coordination of its three functions. Hence, we can conclude that the balance between sales and procurement is the most critical factor for the performance of SCM trading agent.

In summary, we find that SCMaster gives the best results among the three agents when it achieves a high bidding success rate for high profit orders, a lower procurement cost, and a lower sales penalty. The bidding strategy of SCMaster is to focus

Table 12 Bidding performance analysis

	Agent-D	Agent-T	SCMaster
Bidding success rate (order/offer)	3885/8632 = 0.45	7660/10,920 = 0.70	**3472/3566 = 0.97**

The best results are highlighted in bold

on high profit orders. Most of its bids are successful because SCMaster's flexible bidding algorithm provides the bidding prices that can win such orders. The lower procurement cost is achieved through the flexible inventory threshold that avoids excessive inventory. Among the three conditions of success for SCMaster, the lower sales penalty is the most important because it represents the synchronization of bidding and inventory management strategies in maximizing revenue while minimizing inventory cost.

7 Conclusion and future research

7.1 Conclusion of the study

In this study, we developed an enhanced model of supply chain trading agent to improve a supply chain agent's performance. Major functions in a supply chain management agent are bidding to suppliers, procurement of components, and production, so we focused on the improvement of those functions. Winning customers' offers is important to increase the revenue of the agent, and finding appropriate bidding prices is the key to success on the bidding. At the same time, a bidding module should consider the availability of production resources when placing bids, otherwise the outstanding orders will cause a sales penalty. Meanwhile, procurement must supply enough components in time to avoid the lack of components for production. However, excessive components may cause a high volume of holding costs, so agents decide when and how much to purchase depending on the status of the supply chain. Therefore, although the different SCM functions seems separate they are actually linked together and can affect each other's performance.

TAC-SCM provided ample literature and the guideline for SCM simulation, but approaches in TAC-SCM mainly focused on the application of optimization techniques. When considering the characteristic of supply chain, agents need to be flexible against the changes of environment. Therefore, we explored adaptive learning techniques for the SCM modules. In the literature, studies using Q-learning and meta-learning techniques focused on the performance improvement of a part of SCM such as retail or inventory management. As the purpose of this study is to enhance the entire performance of the supply chain trading agent, we improved three SCM functions, which are procurement, sales, and production. To improve the performance of the procurement module, we combined Q-learning, Softmax, ε-greedy, and meta-learning to manage the inventory threshold dynamically, and they successfully improved the agent's procurement performance. By using profit as the reward of action-value function, we succeed in the combining the performance of sales and procurement. For the bidding module a sliding window protocol was applied to provide enhanced accuracy of winning probability and profit threshold was used to resolve bidding problems.

In the simulation SCMaster showed the best performance among three types of agents, and the result of analysis demonstrated that SCMaster's major functions are well organized. In that perspective, this study contributed to the enhancement of supply chain trading agents by selecting appropriate techniques and combining

them. Most studies have used optimization techniques or have focused on the enhancement of partial function of SCM. However, we filled the research gap by not only developing adaptive techniques, but also combining the techniques to provide enhanced overall performance.

7.2 Future research

Although this study demonstrated considerable improvement in performance, many areas were found that could be improved in the future research. The profit threshold used in the sales offer module can be made flexible in the future study. We used a fixed number found heuristically, but it is expected that dynamic methods can be developed. Another possible application of this study is to support human decision makers and add more human-oriented features to the simulation. Apart from finding the most efficient method, there exist a lot of human-related factors, such as trust and impression. By adding these features, it is expected that we may design a more realistic supply chain environment where humans and agents co-exist and interact.

Appendix 1: Meta-learning algorithm

Let
N_τ and N_a be the number of intervals for the parameter τ and a;
$\boldsymbol{Q}(\boldsymbol{\tau}, \boldsymbol{a})$ and $\boldsymbol{Q}(\boldsymbol{\tau})$ be the estimated profit vectors;
$\Pr(\tau)$ and $\Pr(a)$ be the probability distributions for τ and a resp;

Initialize
$Q(\tau, a)$ and $Q(\tau)$ vectors to optimistic initial values;
$\Pr(\tau)$ and $\Pr(a)$ to $1/N_\tau$ and $1/N_a$ respectively;
update the values of τ and a using the above prob. Distributions;

Repeat{
play the game using action a and obtain a reward r;
update the profit vectors $Q(\tau, a)$ and $Q(\tau)$ using Eq. (6) and Eq. (7) respectively;
every *5* iterations do{
update $\Pr(\tau)$ using 1- ε with ε=0.1;
update τ using its updated prob. Distribution;
}

Update $\Pr(a)$ using the Eq.(3);
Update a using its update prob. Distribution;

} Until (the stopping criteria is met)

***** bold and italicize part is the modified part from the previous algorithm.***

Appendix 2: Procurement RFQ creation algorithm

```
Update Inventory Buffer Level {
    Every 5 days{
        Update temperature of Softmax method;
    }

   Update learning rate in Q-Learning;
}

Loop for each component {

    Set due date of RFQ as the first day when amount of available component < buffer level;

    Set quantity of RFQ as the difference between buffer level and the amount of available
    component;

    Set supplier of RFQ as the supplier who can provide cheaper offer price;

    Set reserve price of RFQ as 115 % of estimated supply price on due date;

    If  cheaper estimated price is not found during days between the current day
        and due date of RFQ {
        Send procurement RFQ to supplier;
    }
    else {
       Remove the RFQ from sending list;
    }

}

Create probes for empty slot and for due dates that agent doesn't have available capacity information;

Send probes;
```

Appendix 3: Component catalog

Components	Base price	Supplier	Description
100	1000	Pintel	Pintel CPU, 2.0 GHz
101	1500	Pintel	Pintel CPU, 5.0 GHz
110	1000	IMD	IMD CPU, 2.0 GHz
111	1500	IMD	IMD CPU, 5.0 GHz
200	250	Basus, Macrostar	Pintel motherboard
210	250	Basus, Macrostar	IMD motherboard
300	100	MEC, Queenmax	Memory, 1 GB
301	200	MEC, Queenmax	Memory, 2 GB
400	300	Watergate, Mintor	Hard disk, 300 GB
401	400	Watergate, Mintor	Hard disk, 500 GB

Appendix 4: Bill of materials

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

Appendix 5: Parameters for the simulation

Parameter	Simulation setting
Length of simulation	250 days
Agent assembly cell capacity	2000 cycles/day
Nominal capacity of supplier assembly lines	550 components/day
Acceptable purchase ratio for single-source suppliers	0.75
Acceptable purchase ratio for two-source suppliers	0.45
Average number of customer RFQs per product on a day	13
Average number of demand per product on a day	200
Range of lead time (due date) for customer RFQs	3–12 days from the day the RFQ is received
Range of penalties for customer RFQs	10% of the customer reserve price annually
Customer reserve price	75–125% of nominal price of the PC components
Annual storage cost rate	37.5% of nominal price of components
The number of RFQs to a supplier per component	5

Appendix 6: Bidding performance analysis

SKU (product type)	SCMaster		Agent-D		Agent-T	
	Avg. order price	Difference with Agent-D	Avg. order price	Difference with Agent-T	Avg. order price	Difference with SCMaster
1	979.56	26.30	953.27	40.17	913.09	−66.47
2	1020.98	190.98	830.00	−129.53	959.53	−61.45
3	1017.12	31.34	985.78	26.37	959.41	−57.71
4	1053.77	47.99	1005.79	2.83	1002.96	−50.81
5	1256.23	374.69	881.54	−292.55	1174.09	−82.14
6	1292.17	462.17	830.00	−390.96	1220.96	−71.20
7	1284.33	91.14	1193.19	−29.53	1222.72	−61.61
8	1323.93	86.22	1237.71	−27.68	1265.40	−58.53
9	979.82	23.67	956.16	44.81	911.34	−68.48
10	1020.46	190.46	830.00	−130.87	960.87	−59.59
11	1017.59	32.53	985.06	30.12	954.93	−62.66
12	1042.88	152.38	890.50	−101.70	992.20	−50.68
13	1230.90	80.12	1150.78	−5.79	1156.57	−74.33
14	1280.64	78.42	1202.22	−10.88	1213.09	−67.54
15	1278.90	77.19	1201.70	−7.70	1209.40	−69.50
16	1326.45	85.67	1240.78	−27.44	1268.23	−58.22
Order success rate (order/offer)	3472/3566 = 0.97		3885/8632 = 0.45		7660/10,920 = 0.70	

References

Abadie A (2005) Semiparametric difference-in-differences estimators. Rev Econ Stud 72(1):1–19

Acquity Group (2014) Uncovering the shifting landscape in B2B commerce. https://www.accenture.com/t20150624T211502__w__/us-en/_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Industries_15/Accenture-B2B-Procurement-Study.pdf

Adami C, Schossau J, Hintze A (2016) Evolutionary game theory using agent-based methods. Phys Life Rev 19:1–26

Alpaydin E (2014) Introduction to machine learning. MIT Press, Cambridge

Benisch M, Greenwald A, Grypari I, Lederman R, Naroditskiy V, Tschantz M (2004a) Botticelli: a supply chain management agent designed to optimize under uncertainty. SIGecom Exch 4(3):29–37. https://doi.org/10.1145/1120701.1120706

Benisch M, Greenwald A, Naroditskiy V, Tschantz MC (2004b) A stochastic programming approach to scheduling in TAC SCM. In: Proceedings of the 5th ACM conference on electronic commerce, EC'04. ACM, New York, pp 152–159. https://doi.org/10.1145/988772.988796

Benisch M, Sardinha A, Andrews J, Ravichandran R, Sadeh N (2009) CMieux: adaptive strategies for competitive supply chain trading. Electron Commer Res Appl 8(2):78–90. https://doi.org/10.1016/j.elerap.2008.09.005 **(Special section: supply chain trading agent research)**

Borghetti B, Sodomka E (2006) Performance evaluation methods for the trading agent competition. In: AAAI, pp 1857–1858. http://www.aaai.org/Papers/AAAI/2006/AAAI06-307.pdf

Burke DA, Brown KN, Tarim SA, Hnich B (2006) Learning market prices for a real-time supply chain management trading agent. In: AAMAS 2006 workshop on trading agent design and analysis/agent mediated electronic commerce, pp 29–42

Chaturvedi AR, Mehta SR, Dolk D, Gupta M (2006) Computational experimentations in market and supply-chain co-design: a mixed agent approach. Inf Syst E-Bus Manag 4(1):25–48. https://doi.org/10.1007/s10257-005-0021-6

Chatzidimitriou KC, Symeonidis AL, Kontogounis I, Mitkas PA (2008) Agent Mertacor: a robust design for dealing with uncertainty and variation in SCM environments. Expert Syst Appl 35(3):591–603. https://doi.org/10.1016/j.eswa.2007.07.050

Cheng F, Ettl M, Lin G, Yao DD (2002) Inventory-service optimization in configure-to-order systems. Manuf Serv Oper Manag 4(2):114–132. https://doi.org/10.1287/msom.4.2.114.282

Choi TY, Dooley KJ, Rungtusanatham M (2001) Supply networks and complex adaptive systems: control versus emergence. J Oper Manag 19(3):351–366

Collins J, Arunachalam R, Sadeh NM, Eriksson J, Finne N, Janson S (2006) The supply chain management game for the 2007 trading agent competition. http://Reports-Archive.Adm.Cs.Cmu.Edu/Anon/Isri2007/CMU-ISRI-07-100.Pdf

Collins J, Ketter W, Gini M (2009) Flexible decision control in an autonomous trading agent. Electron Commer Res Appl 8(2):91–105. https://doi.org/10.1016/j.elerap.2008.09.004 **(Special section: supply chain trading agent research)**

Eriksson J, Finne N, Janson S (2006) Evolution of a supply chain management game for the trading agent competition. AI Commun 19(1):1–12

Forbes Technology Council (2016) Upcoming technology: 10 trends to watch in the next five years. www.forbes.com. https://www.forbes.com/sites/forbestechcouncil/2016/12/27/upcoming-technology-10-trends-to-watch-in-the-next-five-years/#1b8ddfe32fd1

Kiekintveld C, Wellman MP, Singh S, Soni V (2004) Value-driven procurement in the TAC supply chain game. ACM SIGecom Exch 4(3):9–18

Kiekintveld C, Miller J, Jordan PR, Wellman MP (2006) Controlling a supply chain agent using value-based decomposition. In: Proceedings of the 7th ACM conference on electronic commerce, EC'06. ACM, New York, pp 208–217. https://doi.org/10.1145/1134707.1134730

McLaren TS, Head MM, Yuan Y (2004) Supply chain management information systems capabilities. An exploratory study of electronics manufacturers. Inf Syst E-Bus Manag 2(2–3):207–222. https://doi.org/10.1007/s10257-004-0035-5

O'Leary DE (2008) Supporting decisions in real-time enterprises: autonomic supply chain systems. Inf Syst E-Bus Manag 6(3):239–255. https://doi.org/10.1007/s10257-008-0086-0)

Pardoe D, Stone P (2009) An autonomous agent for supply chain management. Handb Inf Syst Ser Bus Comput 3:141–172

Podobnik V, Petric A, Jezic G (2006) The CrocodileAgent: research for efficient agent-based cross-enterprise processes. In: On the move to meaningful internet systems 2006: OTM 2006 workshops, lecture notes in computer science. Springer, Berlin, October 29, pp 752–762. https://doi.org/10.1007/11915034_100

Saar-Tsechansky M (2015) The business of business data science in IS journals. MIS Q Manag Inf Syst 39(4):3–6

Sibdari S, Zhang XS, Singh S (2012) A dynamic programming approach for agent's bidding strategy in TAC-SCM game. Int J Oper Res 14(2):121–134. https://doi.org/10.1504/IJOR.2012.046644

Sikora RT (2008) Meta-learning optimal parameter values in non-stationary environments. Knowl Based Syst 21(8):800–806. https://doi.org/10.1016/j.knosys.2008.03.041

Sikora RT, Sachdev V (2008) Learning bidding strategies with autonomous agents in environments with unstable equilibrium. Decis Support Syst 46(1):101–114. https://doi.org/10.1016/j.dss.2008.05.005

Simchi-Levi D, Kaminsky P, Simchi-Levi E, Shankar R (2008) Designing and managing the supply chain: concepts, strategies and case studies. Tata McGraw Hill Education Private Limited, New York

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, adaptive computation and machine learning. MIT Press, Cambridge

Vepsalainen APJ, Morton TE (1987) Priority rules for job shops with weighted tardiness costs. Manag Sci 33(8):1035–1047. https://doi.org/10.1287/mnsc.33.8.1035

Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artif Intell Rev 18(2):77–95. https://doi.org/10.1023/A:1019956318069

Wellman MP, Jordan PR, Kiekintveld C, Miller J, Reeves DM (2006) Empirical game-theoretic analysis of the TAC market games. In: AAMAS-06 workshop on game-theoretic and decision-theoretic agents. http://wwweb.eecs.umich.edu/deepmaize/papers/WellmanGTDT-final.pdf