Expert Systems with Applications 36 (2009) 6520–6526

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system

Chengzhi Jiang *, Zhaohan Sheng

Department of Management and Engineering, Nanjing University, 22 Hankou Road, Nanjing 210093, PR China

ARTICLE INFO

Keywords:
Inventory control
Reinforcement learning
Supply-chain management
Multi-agent simulation

ABSTRACT

Reinforcement learning (RL) appeals to many researchers in recent years because of its generality. It is an approach to machine intelligence that learns to achieve the given goal by trial-and-error iterations with its environment. This paper proposes a case-based reinforcement learning algorithm (CRL) for dynamic inventory control in a multi-agent supply-chain system. Traditional time-triggered and event-triggered ordering policies remain popular because they are easy to implement. But in the dynamic environment, the results of them may become inaccurate causing excessive inventory (cost) or shortage. Under the condition of nonstationary customer demand, the S value of (T, S) and (Q, S) inventory review method is learnt using the proposed algorithm for satisfying target service level, respectively. Multi-agent simulation of a simplified two-echelon supply chain, where proposed algorithm is implemented, is run for a few times. The results show the effectiveness of CRL in both review methods. We also consider a framework for general learning method based on proposed one, which may be helpful in all aspects of supply-chain management (SCM). Hence, it is suggested that well-designed "connections" are necessary to be built between CRL, multi-agent system (MAS) and SCM.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

Supply-chain management (SCM) has been providing competitive advantages for enterprises in the market. In that, inventory control plays an important role and has been attracting attentions from many researchers in recent years. Some known inventory control policies are studied and improved for all aspects, such as reduced cost, more flexibility. Chen, Li, Marc Kilgour, and Hipel (2006) introduce a case-based multi-criteria ABC analysis, that improves on this approach by accounting for additional criteria, such as lead time and criticality of SKUs. This procedure provides more flexibility to account for more factors in classifying SKUs. Lee and Wu (2006) propose the statistical process control (SPC) based replenishment method, in which inventory rules and demand rules are developed to determine the amount of order replenishment for solving order batching problem. This control system performs well at reducing backorders, and bullwhip effect. Yazgı Tütüncü, Aköz, Apaydın, and Petrovic (2007) present new models for continuous review inventory control in the presence of uncertainty. The optimal order quantity and the optimal reorder point are found to minimize the fuzzy cost.

On the other hand, different inventory management systems could be designed according to a specific industry or environment. Aronis, Magou, Dekker, and Tagaras (2004) apply Bayesian approach to forecasting the demand for spare parts of electronic equipment, providing a more accurate determination on stock level for satisfying negotiated customer service level. Ashayeri, Heuts, Lansdaal, and Strijbosch (2006) also develop cyclic production-inventory optimization models for the process manufacturing industry. ElHafsi (2007) shows that optimal inventory allocation policy in an assemble-to-order system is a multi-level state-dependent rationing policy. Díez, Erik Ydstie, Fjeld, and Lie (2008) design model-based controllers based on discretized population balance (PB) models for particular processes, which are encountered in almost any branch of process industries. Kopach, Balcıoğlu, and Carter (2008) revisit a queuing model and determine an optimal inventory control policy using level crossing techniques in blood industry.

Meanwhile, identifying factors affecting inventory management performance such as cost and demand also assists in designing the controllers. Andersson and Marklund (2000) introduce a modified cost structure at the warehouse, and then multi-level inventory control problem can be decomposed to single-level problems. By applying a simple coordination procedure to them, the near optimal solution is obtained. Zhang (2007) studies an inventory control problem under temporal demand heteroscedasticity, which is found to have a significant influence on firm's inventory costs. Chiang (2007) uses dynamic programming to determine the optimal control policy for a standing order system. Yazgı Tütüncü et al. (2007) make use of fuzzy set concepts to treat imprecision regarding the costs and probability theory to treat customer

* Corresponding author. Tel.: +86 13851833708.
E-mail address: tony_tiaotiao@hotmail.com (C. Jiang).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.07.036

demand uncertainty. Additionally, Maity and Maiti (2007) devise the optimal production and advertising policies for an inventory control system considering inflation and discounting in fuzzy environment.

It is observed that in recent researches mentioned, mathematical or analytical models are preferred, such as Bayesian approach (Aronis et al., 2004), Utility Function Method (Maity & Maiti, 2007), fuzzy set concepts (Yazgı Tütüncü et al., 2007) and Autoregressive and Integrated Moving Average and Generalized Autoregressive Conditional Heteroscedasticity (Zhang, 2007). This kind of method provides strict deduction, which usually involves complicated notations and equations under assumptions. However, on one hand, the problem may be time-varying under dynamic environment, especially in the evolving system like supply chain where the solution in one time may be not suitable for another time. On the other hand, those models are too difficult for managers to implement in the real enterprises because of the complicated calculations involved. This requires the learning ability to enrich one's experience continuously in order to make reasonable decisions. Reinforcement learning (RL) is an approach to machine intelligence that combines the fields of dynamic programming and supervised learning to yield powerful machine-learning systems (Harmon & Harmon, 1996). Chi, Ersoy, Moskowitz, and Ward (2007) demonstrate and validate the applicability and desirability of using machine learning techniques to model, understand, and optimize complex supply chains. To make the best use of learning methods, intelligent entities are the necessary carriers. Multi-agent Systems (MAS) seem to be a good choice where the agents are characterized of intelligence, autonomy, interactive and reactivity. Liang and Huang (2006) develop a multi-agent system to simulate a supply chain, where agents are coordinated to control inventory and minimize the total cost of a supply chain. Govindu and Chinnam (2007) propose a generic process-centered methodological framework for analysis and design of multi-agent supply chain systems.

Therefore, this paper proposes a reinforcement learning algorithm combined with case-base reasoning (CRL) in a multi-agent supply-chain system. Similar research is carried out by Kwon, Kim, Jun, and Lee (2007). They suggest a case-based myopic reinforcement learning algorithm for satisfying target service level using vendor managed inventory model. And in this paper, we are trying to provide a simpler learning method with similar or better performance, which could be used more widely and easier to implement by managers. Furthermore, the "connections" are strongly recommended to be built between CRL, MAS and SCM, thus a generic reinforcement learning method is also suggested.

The remainder of this paper is organized as follows. Section 2 explains the multi-agent supply-chain model including the inventory control problem. Section 3 presents the CRL algorithm in more detail. Simulation environment for measuring the performance of CRL is explained and the results are presented in Section 4. Section 5 considers a generic RL method based on the proposed one. Finally, the conclusion and future research are provided in Section 6.

2. Multi-agent supply-chain model

A simplified two-echelon supply chain consisting of multiple customers and retailers is designed for demonstration (see Fig. 1). It is assumed that each retailer receives all ordered stocks from suppliers in a fixed lead time regardless of the amount of order. And it faces nonstationary customer demand under two conditions:

(i) Each retailer has a fixed group of customers whose demand is nonstationary.
(ii) Each customer is free to choose one retailer in each period, i.e., in a competitive market.

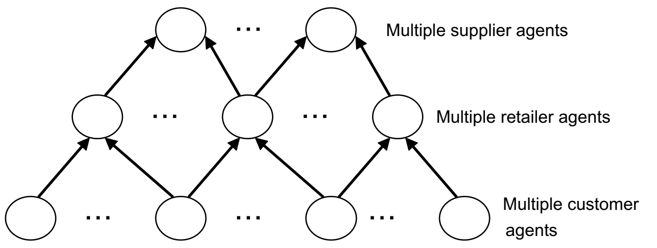

Fig. 1. The supply-chain model under consideration.

The second condition requests additional customer selecting standards and strategies for retailers trying to attract customers or maximize their profits. The former one adopts simplified motivation function proposed by Zhang and Zhang (2007)

$$M_i = PS_i \times P_i + QS_i \times Q_i + ft_i \times in_i. \tag{1}$$

M_i motivation of retailer i ($i = 0$ to $N_{\text{retailer}} - 1$) of a customer agent. PS_i presents customer agent's price sensitivity parameter to retailer i, while QS_i is the quality sensitivity parameter and ft_i is the follower tendency. P_i and Q_i are the price and quality of retailer i respectively. in_i is the influence received from some other customer agents as friends with the respect to retailer i. Eq. (1) is further specified as follows:

$$M_i = (-\alpha^{P_i - P_{\text{ave}}} + k) \times P_i + \left(\beta^{|Q_i - Q_{\text{ave}}|} + L\right) \times Q_i + ft \times in_i, \tag{2}$$

where, $\alpha > 1$, $0 < \beta < 1$, P_{ave} and Q_{ave} are the average price and quality provided by retailers. k is a constant presenting the price sensitivity which is varied according to the social-status of customers. And L is the corresponding quality sensitivity constant of customers. The calculation of in_i is treated as: each customer has a list of influence out from positive to negative corresponding to its own rank of retailers. in_i equals the added value of influence out of retailer i from friend agents.

And a simple adjustment strategy for retailer i is used here as follows:

if its market share is below average, then $P_i = P_i - p$;
else if its market share is above average, then $P_i = P_i + p$;
else no change is made.

Under both conditions, the demand that is not met immediately is treated as lost sales. However, under condition 2, for each customer has its own rank of retailers according to motivation function values, it may select the next one when the retailer of higher rank has insufficient inventory.

Besides, all retailers firstly use periodical review order-up-to (T, S) system and then order-quantity reorder point (Q, S) system for inventory management. And the order-up-to level and reorder point in both systems are learned, respectively using CRL for satisfying target service level (see Figs. 2 and 3). The goal of each retailer is to satisfy its target service level predefined while trying to cut excessive inventory. In this paper, the fill-rate type service level is adopted.

It can be seen that the CRL in (T, S) system takes place before lead time, because it will learn the rewards before and suggest

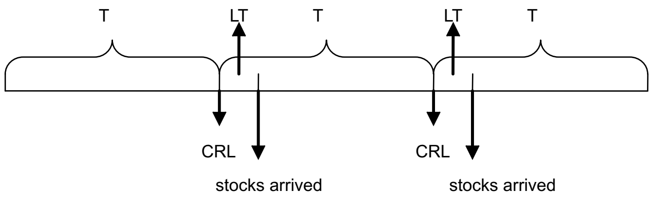

Fig. 2. (T, S) inventory replenishment mechanism.

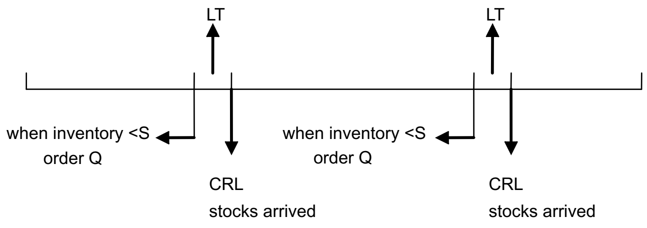

Fig. 3. (*Q*, *S*) inventory replenishment mechanism.

an order-up-to level that is implemented immediately. While in (*Q*, *S*) system, the CRL is implemented after lead time for the reason that the suggested reorder point is used when next condition of inventory replenishment is met.

Finally, all agents and associated attributes are introduced and explained in the following part, the model is programmed under JDK (Java2 Development Kit) 1.5.

1. *Customer* Class: *ID* (*int*), customer identity, from 0 to $N_{\text{customer}} - 1$. *Demand* (*int*), nonstationary. *k* (*int*), price sensitivity parameter. *L* (*int*), quality sensitivity parameter. *Index*[ ] (*array of double*), rank of motivation values. *Rank*[ ] (*array of all retailer objects*), rank of retailers. *Influence_out*[ ] (*array of double*), influence out to friend agent (s) with respect to retailers. *Influence_in*[ ] (*array of double*), received influence from friend agent (s). *Follow* (*double*), follower tendency. *FriendID*[ ] (*array of int*), IDs of friend agent (s). α, β (*static double*), motivation function parameters. *Mean* (*static int*), *CV* (*static double*), *Deviation* (*static int*), *T* (*static int*), *extent* (*static double*), parameters of nonstationary demand. Range (*static int*), number of friend agent(s). *SuperID* (*Vector*), records the ID of seller each time. *BuyHistory* (*Vector*), records IDs of retailers in each defined period.
2. *Retailer* Class: *ID* (*int*), retailer identity. *price* (*double*), price of products provided. *quality* (*double*), quality of products provided. *cost* (*double*), *profit* (*double*), *bank* (*double*), reserved. *Stock_in* (*Vector*), *Stock_out* (*Vector*), stock flow records. *Money_in* (*Vector*), *Money_out* (*Vector*), money flow records, reserved. *LostSales* (*Vector*) records of lost sales. *Order* (*int*), inventory replenishment quantity each time. *TimeNow* (*int*), running time in marker. *T* (*static int*), review period of (*T*, *S*) system. *Q* (*static int*) order quantity of (*Q*, *S*) system. *Max_Stock* (*int*), order-up-to level for (*T*, *S*) system or re-order point for (*Q*, *S*) system. TargetSL (*static double*), predefined target service level. *SLave* (*static double*), average service level. *Current_Stock* (*int*), current inventory level. *CurrentSL* (*double*), current service level achieved. *Case* [ ] (*array of Vectors*), *CaseSize* (*int*), *Counter* [ ] (*array of int*), *CaseMarker* (*int*), *ActionMarker* (*int*), *UpdateMarker* (*int*), *InvenIncrease* (*int*), *InvenDecrease* (*int*), *SLba* [ ] (*array of double*), *SLaa* [ ] (*array of Vectors*), *Actions* [ ] (*array of Vectors*), elements of CRL. *SuperID* (*Vector*), *SuperHistory* (*Vector*), records of suppliers, reserved. *NextID* (*Vector*), *NextHistory* (*Vector*), records of customers.
3. *Transfer Class*: It is in charge of transferring money and stock between upstream and downstream agents, calculating average price, quality and service level and so on.
4. *Business Class*: It provides the market environment for supply chain. The work flow of other agents is controlled in it.

3. Case-based reinforcement learning

In this section, the elements of case-base reinforcement learning mentioned in Section 2 are explained in more details and the whole procedure is presented. The order-up-to level (*S*) and reorder point (*S*) are learned based on the past experience. The resulting service levels of previously suggested *S* values are treated as rewards for the actions taken before. The (State, Action, reward) records provide reference when similar state is met again.

The details of elements are shown as follows:

$$Case[CaseSize] = \{Case[0], Case[1], \ldots, Case[CaseSize-1]\},$$

where *Case* [*i*] = {*inventory level* (*int*), D_{mean} (*int*)}.

Case [*CaseSize*] (*array of vectors*) is used to record the different states met. Each state includes two elements: current *S* value and estimated current mean of customer demand.

$$Counter[CaseSize] = \{Counter[0], Counter[1], \ldots, Counter[CaseSize-1]\},$$

where *Counter* [*i*] = {*counter* (*int*)}.

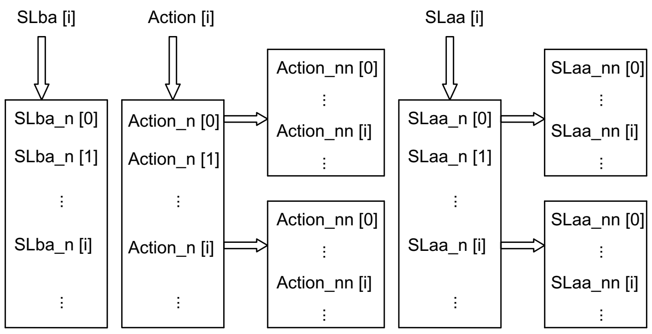

Fig. 4. Overview of data structure.

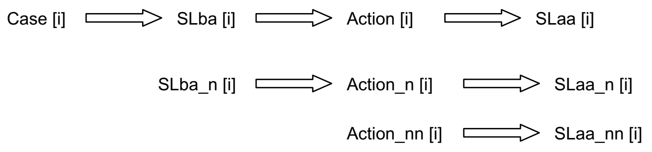

Fig. 5. Corresponding relationships of elements.

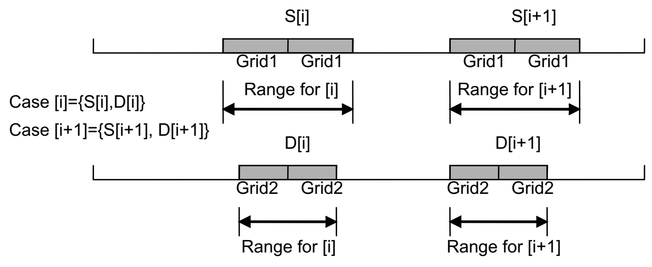

Fig. 6. Criteria for similarity of level 1 searching.

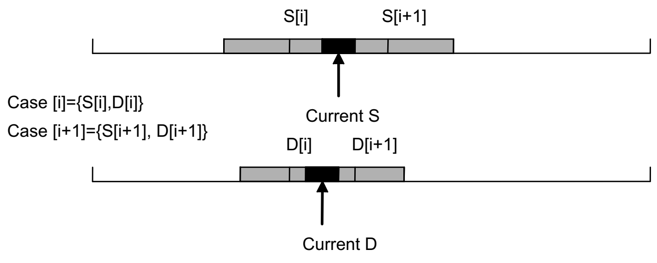

Fig. 7. Situation of two similar states.

Table 1
Simulation modes under condition 1

Mode [*i*]	CV	*T*	Extent
M1	0.2	Uniform (50, 80)	Uniform (−1, 1)
M2	0.2	Uniform (15, 30)	Uniform (−2, 2)
M3	0.4	Uniform (50, 80)	Uniform (−1, 1)
M4	0.4	Uniform (15, 30)	Uniform (−2, 2)

Counter [*CaseSize*] (*array of int*) is an array of counters which memorize the frequency of the corresponding state met.

$$SLba[CaseSize] = \{SLba[0], SLba[1], \ldots, SLba[CaseSize - 1]\},$$

where *SLba* [*i*] = {*SLba_n* [*0*], *SLba_n* [*1*],...,*SLba_n* [*i*],...}, where:
SLba_n [*i*] = {*Current Service Level before action* (*double*)}.

SLba [*CaseSize*] (*array of double*) records the current service levels before action under corresponding case

$$Action[CaseSize] = \{Action[0], Action[1], \ldots, Action[CaseSize - 1]\},$$

where *Action* [*i*] = {*Action_n* [*0*], *Action_n* [*1*],..., *Action_n* [*i*],...}, where:
Action_n [*i*] = {*Action_nn*[*0*], *Action_nn*[*1*],..., *Action_nn* [*i*],...}, where:
Action_nn [*i*] = {*Change amount* (*int*)}.

For multiple choices could be made under a single state, multi-hierarchy structure is adopted for records of actions taken. The action taken is interpreted as adding the "change amount" (positive or negative) to current *S* value of (*T*, *S*) or (*Q*, *S*). The exact amount is presented by *InvenIncrease* (*int*) or *InvenDecrease* (*int*).

$$SLaa[CaseSize] = \{SLaa[0], SLaa[1], \ldots, SL[CaseSize - 1]\},$$

where *SLaa* [*i*] = {*SLaa_n* [*0*], *SLaa_n* [*1*],..., *SLaa_n* [*i*],...}, where:
SLaa_n [*i*] = {*SLaa_nn* [*0*], *SLaa_nn* [*1*],..., *SLaa_nn* [*i*],...}, where:
SLaa_nn [*i*] = {*resulting service level after action* (*reward*) (*double*)}.

According to action structure, records of resulting service levels use the same configuration. The overview of data structure of elements is displayed in Fig. 4 and their corresponding relationships are summarized in Fig. 5.

All initial values of data structures above are set to empty. The case-based reinforcement learning algorithm is summarized as three-level searching. The steps of it are described as follows:

Step 0: Before searching, the *SLaa_n* [*i*] is updated as reward for the last action. The exact location of update is recorded by three markers: *CaseMarker*, *ActionMarker*, *UpdateMarker*.

Step 1: Level 1 searching. Search the case records for similar states. If the case is empty or no similar state is found in the history, go to step 1.1. Else, go to step 1.2. The criteria for similarity are shown in Fig. 6. If current *S* resides in two situation ranges (see Fig. 7), the closer situation (closer to *S*[*i*] or *S*[*i + 1*]) is selected. If the distance is equal, select one randomly.

Step 1.1: If the case record is not full, add the new state to the end of the record. Else, replace the earliest state of lowest frequency (smallest counter) with the new

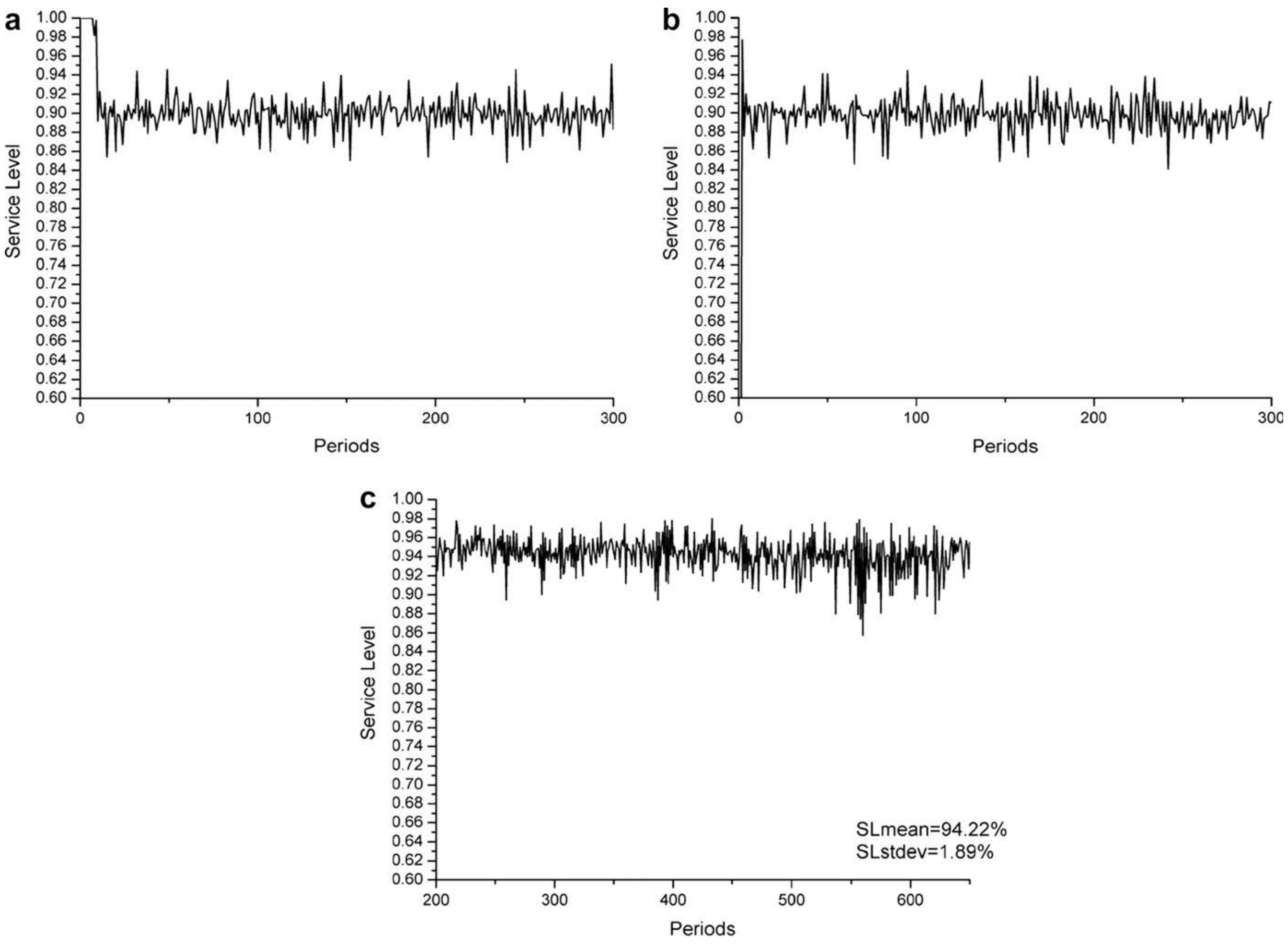

Fig. 8. (a) Initial *S* > *D*, (b) Initial *S* < *D*, (c) Target service level = 95%.

state. Memorize the location in case records using *CaseMarker*.

Step 1.2: Update *CaseMarker*.

Step 2: Level 2 searching. Check *Case* [*CaseMarker*]. If it is a new case, add current service level to *SLba_n* [] and update *ActionMarker* and *UpdateMarker* indicating new record added. Go to step 2.1. Else, search *SLba_n* [] for similar one with the criteria in step 1, setting the grid as 0.2%. Then, if no similar service level is found, go to step 2.1. Otherwise, update *ActionMarker* and go to step 3.

Step 2.1: Add current service level to the end of SLba record. Compare the current service level to target service level. If it is among the range of [TargetSL − 0.2%, TargetSL + 0.2%], no change will be made and 0 is added to action records. Else, calculate *InvenIncrease* or *InvenDecrease* based on the estimated mean of customer demand and difference between current service level and target service level. Add this amount to action records and update *S* value.

Step 3: Level 3 searching. Search *SLaa_n* [*ActionMarker*] for closest service level to target service level. Denote this record as SLC. If SLC is among the range of [TargetSL − 0.2%, TargetSL + 0.2%], go to step 3.1. Else, go to step 3.2.

Step 3.1: Get the corresponding action in *Action_n* [*ActionMarker*] and update *S* value and *UpdateMarker*. In the step 1 of next period, replace SLC with the resulting service level. It guarantees that if this action results in a service level out of target range, it is no longer a qualified action.

Step 3.2: Calculate *InvenIncrease* or *InveDecrease* based on the estimated mean of demand and the difference between SLC and target service level. Add this amount to action records and update *UpdateMarker* indicating that new *SLaa_nn* [*i*] will be added. Thus, the old SLC will not be replaced.

Step 4: repeat previous steps when inventory replenishment condition is met again.

4. Simulation results and analysis

The supply-chain model in consideration simulated consists of 10 retailers and 80 customers. Under condition one in section two,

Table 2
Parameters added for condition 2

Parameters	Scale	Distribution
L	$0 \sim 100$	Random distribution
K	$-100 \sim 0$	Random distribution
Follow_i	$0 \sim 100$	Normal distribution, $\mu = 50$ and $\delta = 15$
Influence_out_i	$-30 \sim 30$	$\{30, 30 - g, 30 - 2g, \ldots, -30\}$ where $g = 60/N_{\text{retailer}}$
Range	2	Constant
P_i	$80 \sim 90$	Random distribution
Q_i	$80 \sim 90$	Random distribution
α	$\alpha > 1$	Depend on testing
β	$0 < \beta < 1$	Depend on testing

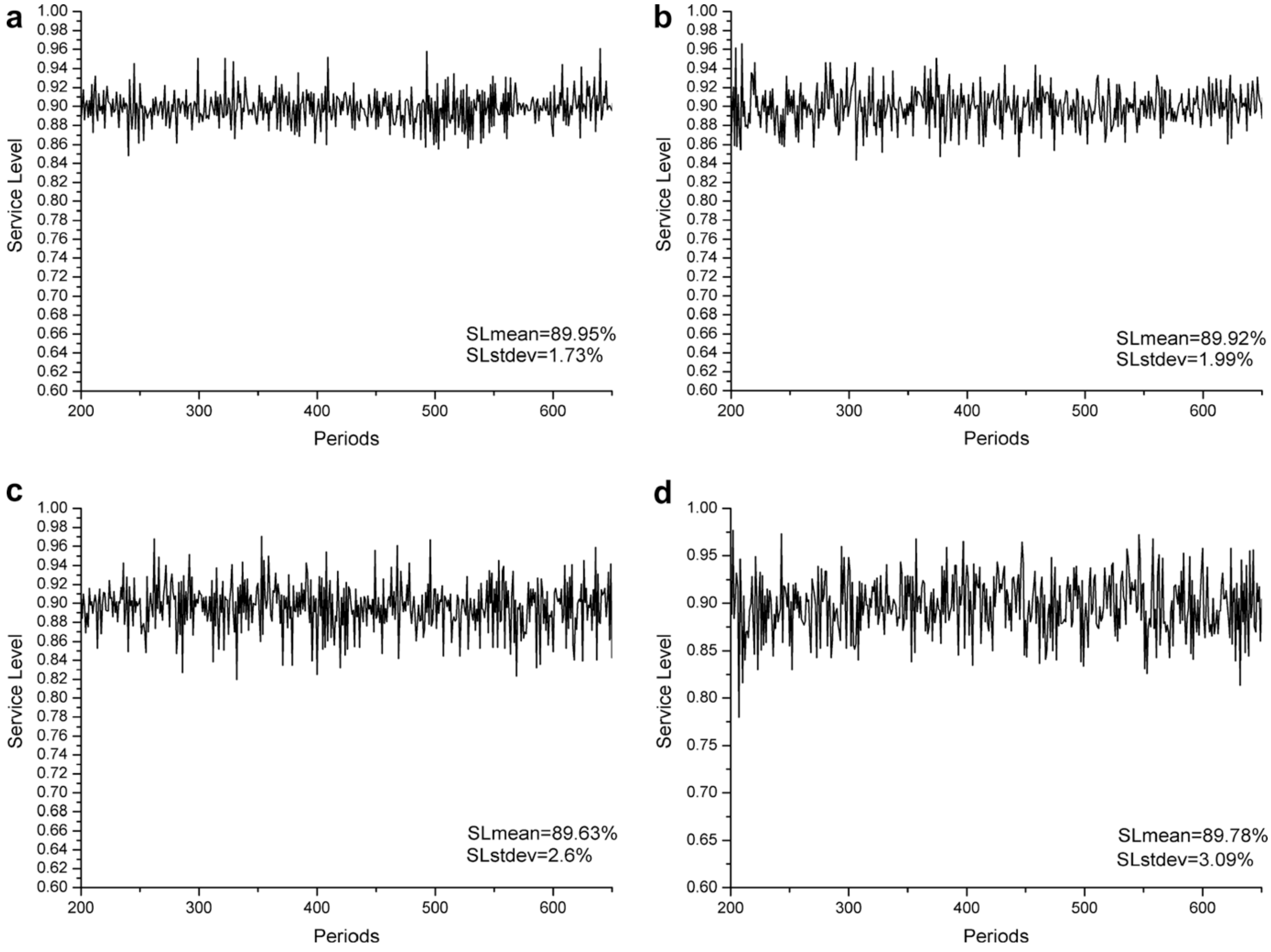

Fig. 9. (a) M1, (b) M2, (c) M3, (d) M4 under condition 1 for (T, S).

each retailer has a fixed group of eight customers. Their demand follows a normal distribution $N(\mu, \delta^2)$, but its mean is changed by two parameters: interval T and ranged *extent*, which follow a uniform distribution. It means that at every interval T, $\mu = \mu + extent$.

Two types of demands are considered which are defined as:

TE 1: T = Uniform (50, 80), *extent* = Uniform (−1, 1).
TE 2: T = Uniform (15, 30), *extent* = Uniform (−2, 2).

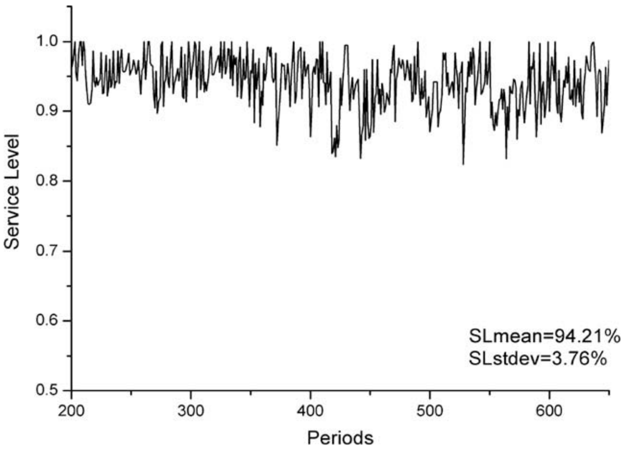

Fig. 10. Results under condition 2 for (T, S).

The standard deviation of customer (δ) is computed by multiplying μ and coefficient of variable (CV). And initial μ is set to 20. The target service level for each retailer is set to 90% equally. Lead time is set to 1 day for (T, S) and 4 days for (Q, S). Each simulation is conducted for 1000 review periods. For the same experiment condition, 20 simulations are repeated with different seeds and their average service level is considered as actual service level. The modes used in simulation under condition 1 are shown in Table 1.

The values of grid1 and grid2 in Fig. 6 are initially set to 20 and 4 and are then changed as the mean value and standard deviation change.

In order to show the independence of initial values, first 300 review period is shown with: Initial supply > demand, Initial supply < demand, target service level is temporarily set to 95% (see Fig. 8).

Fig. 9 shows the simulation results over time for four modes in (T, S) system under condition 1. It can be seen that as the nonstationary of customer is becoming more severe, the deviation of average service level increases. However, the average service level is kept very closely to target service level.

The simulation parameters (see Table 2) are added for simulation under condition 2.

Fig. 10 show the results under condition 2. The customer demand in this situation is much more nonstationary for the reason that one retailer may loose most of its customers after increasing

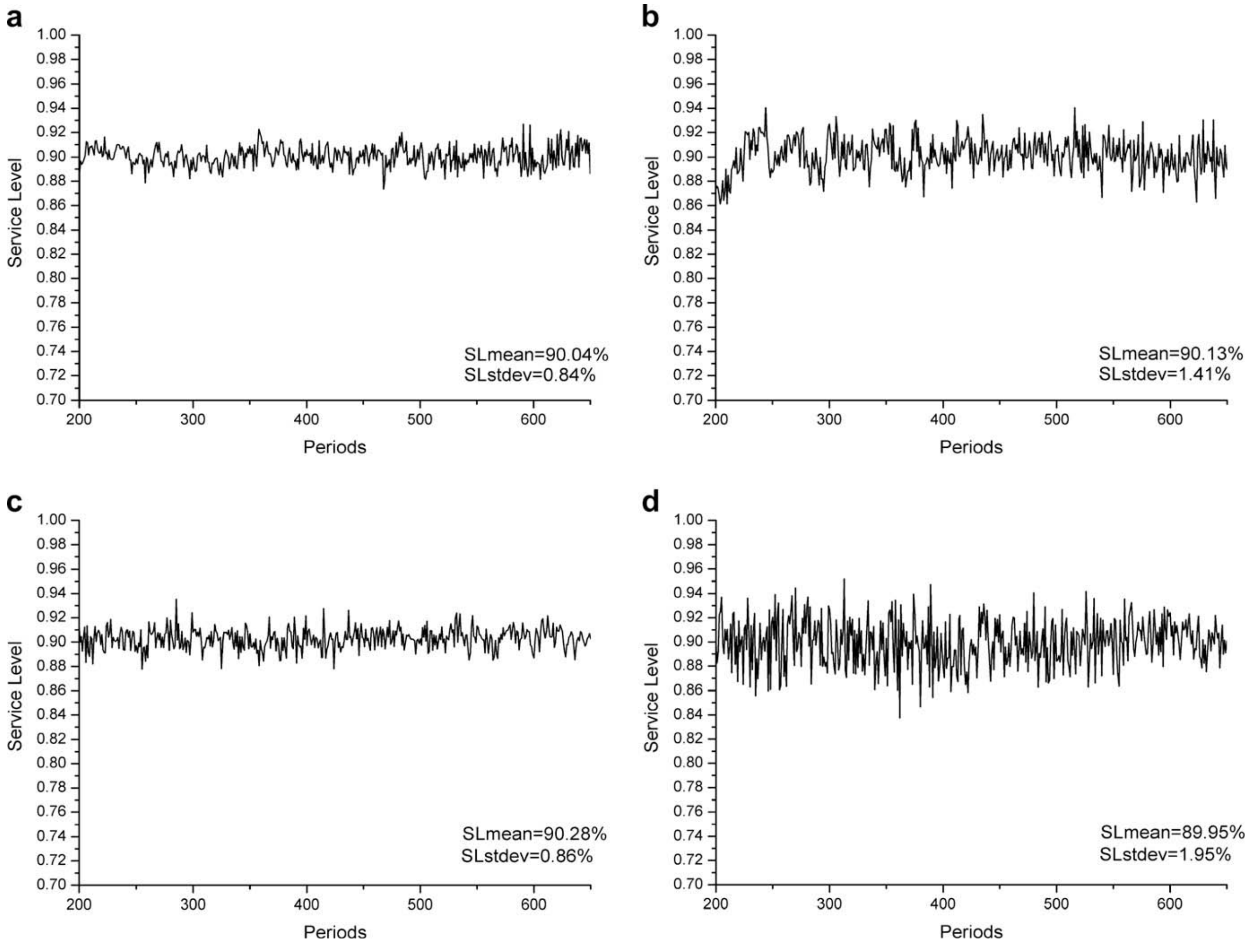

Fig. 11. (a) M1, (b) M2, (c) M3, (d) M4 under condition 1 for (Q, S).

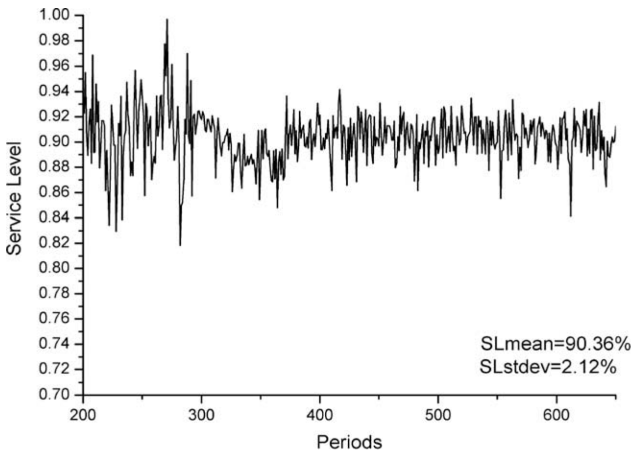

Fig. 12. Results under condition 2 for (*Q*, *S*).

its price while others may win much more customers after decreasing their prices. Therefore, the performance here is not as good as the one under condition 1.

Fig. 11 shows the simulation results in (*Q*, *S*) system under condition 1 and Fig. 12 shows the performance under condition 2. The similar trend is found as in (*T*, *S*) system. Furthermore, in all situations, the case-based reinforcement learning performs better in (*Q*, *S*) system than in (*T*, *S*) system. This is mainly due to the procedure indicated in Figs. 2 and 3. In (*T*, *S*) system, the suggested *S* value is calculated before lead time, i.e. it is based on duration of $[T_1, T_2]$. And the valid duration of *S* is actually from $[T_1 + LT, T_2 + LT]$. Meanwhile, in (*Q*, *S*) system, the *S* value is calculated for the period of lead time (the actual valid duration). This difference makes the better performance in (*Q*, *S*) system.

5. Consideration for general case-based reinforcement learning in supply chain

The efficiency of the CRL is proved in the simulation results. But the application of this algorithm in supply chain is far from the end. Various issues in supply chain are difficult to resolve under evolving or dynamic environment, such as trading competition. RL could be used in many areas because of its generality. In RL, the computer simply gives a goal and then it learns how to achieve that goal by trial-and-error iteration with its environment (Harmon & Harmon, 1996).

Based on the algorithm for inventory control in this paper, a general framework is considered. The same structure could still be used as follows.

(*States*, *Target level achieved before action*, *Actions*, *Target level achieved after actions*)

States may be made up of related parameters in environment, which describe the current situation, e.g., average price and average quality. And current target may be calculated before any actions taken, e.g., current profit or market share in trading competition. Sometimes, manager will not know which action could lead to a result closer to target. But he can try some kind of action based on some motivation or information and learn the delayed reward for that action. This is so-called trial-and-error iteration. The qualified actions could be directly used from experience while unqualified actions are avoided. And the category of qualified and unqualified is dynamic according to perceived rewards reflecting the changes in environment. With the accumulated experience, the error is becoming less and less.

6. Conclusion and future research

In this paper, the problem of dynamic inventory control for satisfying target service level in supply chain with nonstationary customer demand is studied. The case-based reinforcement learning is applied and proved experimentally to be effective. Furthermore, the general CRL is considered for the purpose of applying CRL widely in supply chain. Hence, the thinking behind this paper is to link CRL to supply-chain management where multi-agent system (MAS) is necessary. The future researches may reside in two directions. One direction is to extend the CRL to a multi-stage multi-agent supply chain, so that bullwhip effect may be observed and reduced. The other direction is to apply CRL to other issues in supply chain such as trading competition.

Acknowledgements

This research is supported by National Natural Science Foundation of PRC under the Grant Nos. 70731002 and 70571034 and Computational Experiment Center for Social Science at Nanjing University.

References

Andersson, Jonas, & Marklund, Johan (2000). Decentralized inventory control in a two-level distribution system. *European Journal of Operational Research, 127*, 483–506.

Aronis, Kostas-Platon, Magou, Ioulia, Dekker, Rommert, & Tagaras, George (2004). Inventory control of spare parts using a Bayesian approach: A case study. *European Journal of Operational Research, 154*, 730–739.

Ashayeri, J., Heuts, R. J. M., Lansdaal, H. G. L., & Strijbosch, L. W. G. (2006). Cyclic production-inventory planning and control in the pre-Deco industry: A case study. *International Journal of Production Economics, 103*, 715–725.

Chen, Ye, Li, Kevin W., Marc Kilgour, D., & Hipel, Keith W. (2006). A case-based distance model for multiple criteria ABC analysis. *Computers & Operations Research, 35*, 776–796.

Chi, Hoi-Ming, Ersoy, Okan K., Moskowitz, Herbert, & Ward, Jim (2007). Modeling and optimizing a vendor managed replenishment system using machine learning and genetic algorithms. *European Journal of Operational Research, 180*, 174–193.

Chiang, Chi (2007). Optimal control policy for a standing order inventory system. *European Journal of Operational Research, 182*, 695–703.

Díez, Marta Dueñas, Erik Ydstie, B., Fjeld, Magne, & Lie, Bernt (2008). Inventory control of particulate processes. *Computers and Chemical Engineering, 32*, 46–67.

ElHafsi, Mohsen (2007). Optimal integrated production and inventory control of an assemble-to-order system with multiple non-unitary demand classes. *European Journal of Operational Research*. doi:10.1016/j.ejor.2007.12.00.

Govindu, Ramakrishna, & Chinnam, Ratna Babu (2007). MASCF: A generic process-centered methodological framework for analysis and design of multi-agent supply chain systems. *Computers & Industrial Engineering, 53*, 584–609.

Harmon, Mance E., & Harmon, Stephanie S. (1996). Reinforcement learning: A tutorial. Available from: <http://www.nada.kth.se/kurser/kth/2D1432/2004/rltutorial.pdf>.

Kopach, Renata, Balcıoğlu, Barış, & Carter, Michael (2008). Tutorial on constructing a red blood cell inventory management system with two demand rates. *European Journal of Operational Research, 185*, 1051–1059.

Kwon, Ick-Hyun, Kim, Chang Ouk, Jun, Jin, & Lee, Jung Hoon (2007). Case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications*. doi:10.1016/j.eswa.2007.07.00.

Lee, H. T., & Wu, J. C. (2006). A study on inventory replenishment policies in a two-echelon supply chain system. *Computers & Industrial Engineering, 51*, 257–263.

Liang, Wen-Yau, & Huang, Chun-Che (2006). Agent-based demand forecast in multi-echelon supply chain. *Decision Support Systems, 42*, 390–407.

Maity, K., & Maiti, M. (2007). A numerical approach to a multi-objective optimal inventory control problem for deteriorating multi-items under fuzzy inflation and discounting. *Computers and Mathematics with Applications*. doi:10.1016/j.camwa.2007.07.011.

Yazgı Tütüncü, G., Aköz, Onur, Apaydın, Ayşen, & Petrovic, Dobrila (2007). Continuous review inventory control in the presence of fuzzy costs. *International Journal of Production Economics*. doi:10.1016/j.ijpe.2007.10.01.

Zhang, Xiaolong (2007). Inventory control under temporal demand heteroscedasticity. *European Journal of Operational Research, 182*, 127–144.

Zhang, Tao, & Zhang, David (2007). Agent-based simulation of consumer purchase decision-making and the decoy effect. *Journal of Business Research, 60*, 912–922.