

## A simulation-based approach to study stochastic inventory-planning games

JAIDEEP J. RAO<sup>†\*</sup>, KIRAN KUMAR RAVULAPATI<sup>‡</sup> and TAPAS K. DAS<sup>§</sup>

*Non-cooperative decision-making problems in a decentralized supply chain can be characterized and studied using a stochastic game model. In an earlier paper, the authors developed a methodology that uses machine learning for finding (near) optimal policies for non-zero sum stochastic games, and applied their methodology on an  $N$ -retailer and  $W$ -warehouse inventory-planning problem. The focus of this paper is on making the methodology more amenable to practical applications by making it completely simulation-based. It is also demonstrated, through numerical example problems, how this methodology can be used to find (near) equilibrium policies, and evaluate short-term rewards of stochastic games. Short-term rewards of stochastic games could be, in many instances, more critical than equilibrium rewards. To our knowledge, no methodology exists in the open literature that can capture the short-term behaviour of non-zero sum stochastic games as examined in this paper.*

### 1. Introduction

Success in the current business environment depends greatly on agility and responsiveness. With online trading in the limelight and shrinking service lead time, the need for an efficient supply chain distribution system has become critical. Distributors can no longer afford the high inventory buffers that had plagued the supply chains in the past. Supply-chain optimization is being sought through careful planning and resource-sharing among the agents. Novel strategies being innovated are opening new profit-making opportunities that an individual distributor could not exploit on their own.

Decision-making in a decentralized supply-chain scenario is often non-cooperative, and can be characterized and studied using a non-zero-sum stochastic game model; a non-zero-sum game is where gain of one player is not necessarily the loss of another. Such stochastic games, though increasingly prevalent in the current business environment, are among the least understood of the

management-science-related problems. Available game-theoretic tools are limited and are difficult to implement computationally, even for small problems.

Reinforcement learning (RL), a simulation-based stochastic optimization approach, has been used by many researchers to solve stochastic game problems (Littman 1994, Erev and Roth 1998, Hu and Wellman 1998). Recently, Ravulapati *et al.* (2003) presented a combined RL and matrix game-based algorithm that was used to obtain near optimal actions/rewards for the agents of a non-zero sum stochastic game problem modelled as a competitive Markov decision process (CMDP) (Filar and Vrieze 1997). RL is a machine-learning method that allows the decision-maker in a dynamic environment to learn from the rewards obtained from the actions taken over time and reach (near) optimal decisions. RL uses simulation as its modelling tool and can thereby deal with problems that have complex reward and stochastic structures. Also RL can integrate within it various function approximation methods (regression, neural networks, etc.), which makes it possible to solve problems with very large state spaces. The theory of RL is based on three important scientific principles, the Bellman's equation (Bellman 1957), the theory of stochastic approximation (Robbins and Monro 1951), and the theory of function approximation (Ripley 1996). Hence, RL algorithms are well founded and have, in general, good optimality

---

Received 3 January 2002. Revised 26 August 2003. Accepted 22 October 2003.

<sup>†</sup> Pilgrim Software, Tampa, FL 33618, USA.

<sup>‡</sup> Delta Technology, Atlanta, GA 30354, USA.

<sup>§</sup> Department of Industrial and Management Systems Engineering, University of South Florida, 4202 East Tower Avenue, ENG 118, Tampa, FL 33620-5350, USA.

\* To whom all correspondence should be addressed.

e-mail: das@eng.usf.edu

properties. The potential of RL-based methods has been tapped in several cooperative type industrial application problems. Some of these are Crites and Barto (1996), Singh and Bertsekas (1996), Das *et al.* (1999), and Paternina-Arboleda and Das (2000), and Gosavi *et al.* (2002) among others.

The focus of this paper is to present a computationally attractive alternative to the methodology that was presented in Ravulapati *et al.* (2003). This approach uses only simulation and RL and does not require the solution of matrix games as part of the overall solution methodology. The modified approach thus avoids a computationally difficult element. The new approach is exposed through a detailed examination of the same non-cooperative inventory planning problem studied in Ravulapati *et al.* (2003), and the results are used to assess the effectiveness of the new approach. In addition, this paper demonstrates that the RL-based methodology can be used: (1) to study short-term behaviours of games and (2) to conduct perturbation analysis of gaming strategies. For many game applications, returns in finite time horizons are often more critical than the long-term equilibrium rewards. A business may not survive the negative cash flows that could result from the path that a game might evolve through before perhaps reaching a profitable equilibrium. Obtaining short-term rewards of a stochastic perhaps reaching a profitable equilibrium. Obtaining short-term rewards of a stochastic game is, in theory, considerably more complex than the already difficult task of finding an equilibrium behaviour. However, the RL-based methodology provides a unique opportunity in this regard because simulation is used as a modelling tool. Also, the players of stochastic game are often faced with the question ‘What if a subset of the players deviates (willfully or not) from the equilibrium policy?’ The RL-based methodology provides a useful tool for analysing such perturbation scenarios, which is demonstrated through a few chosen perturbed conditions of the retailer/warehouse inventory problem.

The rest of the paper is organized as follows: Section 2 describes how RL works as an optimization tool; Section 3 presents the problem description; the algorithms and their implementation framework are presented in Section 4; analysis using numerical results is presented in Section 5; short-term reward and perturbation analysis results are presented in Section 6; and Section 7 presents the concluding remarks.

## 2. Reinforcement learning (RL): A simulation-based stochastic optimization tool

RL is best suited to stochastic optimization problems that can be modelled as Markov or semi-Markov processes, since it is founded on the theory of stochastic

dynamic programming (Bellman 1957). The major drawbacks of classical dynamic programming (DP) are (1) that it is difficult to set up and solve the linear system of Bellman equations for large state spaces—this is referred to as the *curse of dimensionality*, and (2) that it requires the determination of the transition probabilities which may be very difficult to obtain when the system dynamics are complex—this is also called the *curse of modelling*. RL is a simulation-based asynchronous and approximate dynamic programming approach. It is asynchronous because, instead of updating the values of all system state and action combinations in every iteration as in classical DP algorithms, RL updates one state-action combination at a time. RL eliminates the need for computation of the transition probabilities, since, instead of obtaining the expected value of the future state, it stochastically approximates the future value from the actual value of the state visited. The learning agent assigns rewards and punishments for their actions based on temporal feedback obtained during their active interactions with dynamic systems. The method of assigning rewards and punishments uses both the Bellman’s equation of DP and the Robbins–Monro stochastic approximation.

Any learning model basically contains four elements, which are the environment, the learning agents, a set of actions for each agent, and the environmental response (sensory output). Each learning agent selects an action, and these actions collectively lead the system along unique path until the system encounters another decision-making state. During this state transition, the agents gather sensory outputs from the environment and from it derive information about the new state and the immediate reward from the current transition. Using the information obtained during the state transition, the agents update their knowledge base and select their next actions. As this learning process repeats, performance continues to improve. Suppose, for a system with only one agent (decision-maker), when action  $a$  is chosen in state  $l$ , and the system makes a transition to state  $q$ , this results in an immediate reward  $r(l, a, q)$ . Then, for average reward performance criterion, the updating of the reinforcement value for the state–action combination  $(l, a)$  is done using a learning version of Bellman’s equation as follows.

$$Q_{\text{new}}(l, a) = (1 - \alpha)Q_{\text{old}}(l, a) + \alpha \left[ r(l, a, q) - g^* + \max_b Q_{\text{old}}(q, b) \right],$$

where  $Q_{\text{new}}(l, a)$  and  $Q_{\text{old}}(l, a)$  are the new and old reinforcement values for the state–action combination  $(l, a)$ ,  $\alpha$  is the learning parameter, and  $g^*$  is a scalar (for which the most recent value of the average reward can be used; refer to step 2 of the RL algorithms presented later).

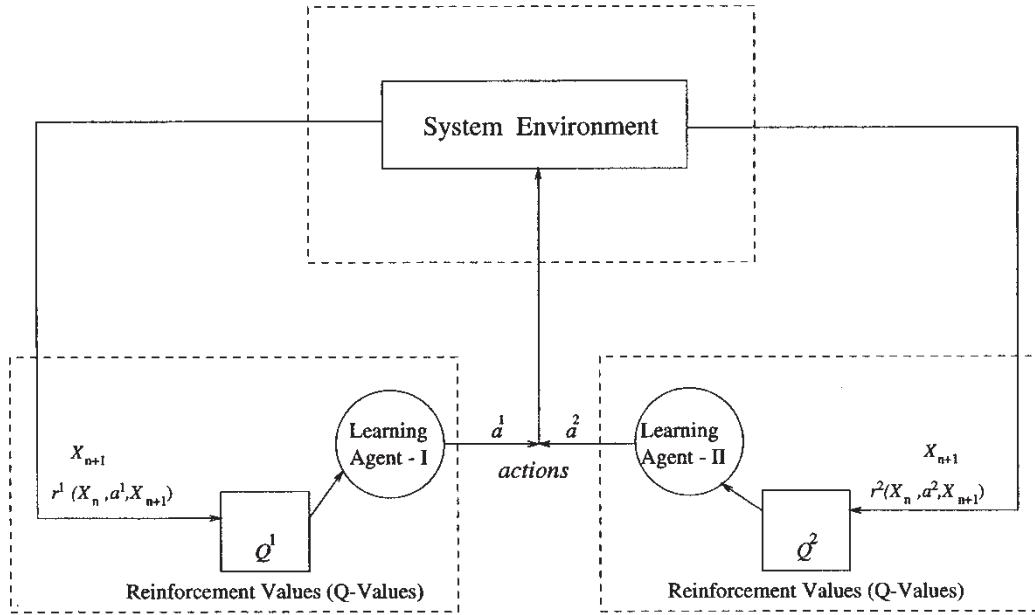


Figure 1. A multi-agent reinforcement-learning model.

A schematic diagram for a two-agent reinforcement-learning model is shown in figure 1. The system is provided with an action, which is a collection of actions chosen by each agent from the action spaces. In turn, each agent receives sensory inputs that determine the next state and the reward or punishment resulting from its most recent action. For example, on the  $n$ th step of the interaction of an  $N$  agent system, based on the system state ( $X_n$ ) and the reinforcement values for agent  $i$  ( $Q^i(X_n=l, a^i)$ ), the  $i$ th agent takes an action  $a^i$ . Note that each agent makes a decision about the next action independent of the other agents. The system evolves stochastically based on the action vector  $a = \{a^i : \forall i \in \{1, 2, \dots, N\}\}$ , as chosen by all agents, and results in output concerning the next system state ( $X_{n+1} = q$ ) and the reward (or punishment) for each agent ( $r^i(l, a, q)$ ) obtained during the transition. These system outputs serve as sensory inputs for the agents. Using the information about the new state and the reward (punishment), the reinforcement values  $Q^i(l, a)$  are updated for the previous state ( $X_n = l$ ) and all players  $i$ .

### 3. Problem description and formulation

The inventory system consists of  $N$  retail outlets belonging to independent agents. The agents deal with a single product, and are interested in maximizing their own profit, rather than that of the entire system. The retail agents may pool inventories at  $W$  central warehouses. Pooling of inventory in central warehouses is a common practice among businesses to deal with the demand uncertainty and to take advantage of the

transportation and storage costs. Demands are independent in each of the  $N$  retail locations. The warehouses do not experience any demand, and the agents make inventory decisions for the warehouses. For simplicity, it is assumed that each agent controls one outlet. The agents share the residual inventory (inventory left after realizing the daily demand) at retail outlets as well as the stocks at the central warehouses to meet the residual demands at the outlets at the end of a business day. Note that the warehouse inventories are used only for meeting residual demands at the end of a day. Unsatisfied demands are not back-ordered. A typical retail/warehouse inventory system is depicted in figure 2. A general framework for study of a deterministic abstraction of this inventory planning problem can be found in Anupindi *et al.* (2001).

In systems, as described above, there are two stages of decision-making that are separated in time. The first stage involves the inventory decision that is made before realizing the demand. This stage includes the decision of stock levels at local as well as warehouse locations. The agents choose local and warehouse stock levels unilaterally (pure competition). The second stage involves the trans-shipment decision that is made after the demand realization. All the agents must come to an agreement (complete cooperation) regarding the shipping of the local residual inventory and the inventory at the warehouses to meet the residual demands, and also regarding the allocation of the excess profit made from inventory sharing. (This hybrid of a non-cooperative as well as cooperative decision-making is commonly referred to as *cooptition*.) However, the decision about the mechanism for allocation of the excess

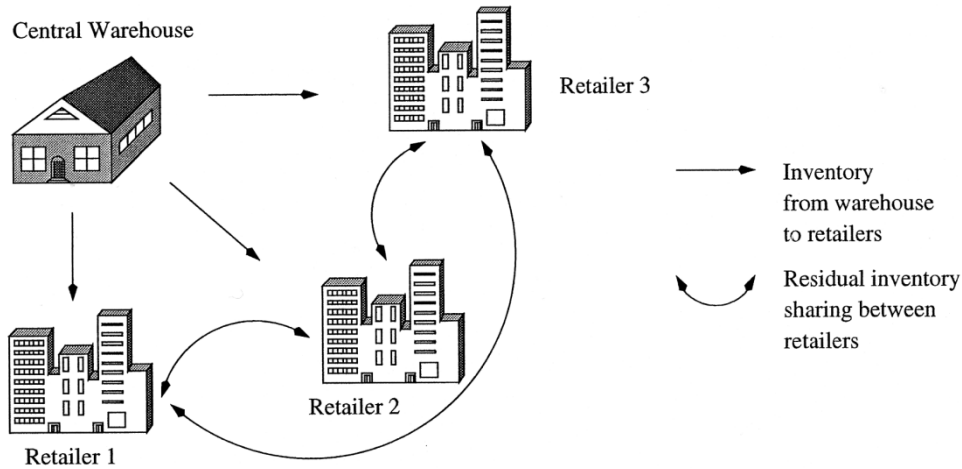


Figure 2. A three-retailer (agent) and one-warehouse inventory system.

profit among the agents should be made before realizing the demands. When an agent makes an inventory decision, the possibilities of the actions of the other agents and corresponding profit allocations are taken into account. Thus, an agent's decision is very much tied to the actions of the other agents. The profit allocation rule should be designed such that it does not give incentive for individual agents or groups of agents to break away from the grand coalition; such allocation policies are called *core allocations*. We have modelled this inventory-planning problem as a combination of a competitive Markov decision problem (CMDP) for the non-cooperative part of the decision-making, and an integer linear programming (ILP) problem for the cooperative part of the decision-making. The ILP is adopted from Anupindi *et al.* (2001).

When a single super agent is substituted for all the multiple non-cooperative agents, and the super agent is allowed to make decisions for all the agents to maximize the system profit, the CMDP becomes a MDP. This single-agent problem (SAP), when solved optimally, gives a solution that is an upper bound for the non-cooperative multi-agent problem (MAP). Note that the ways in which the SAP and MAP solutions can differ are in the total system reward and in the distribution of individual rewards to the players. In the SAP, the decision-maker has complete information and makes decisions to maximize the total system reward, whereas, in the MAP, the players act with incomplete information while trying to maximize their individual rewards, without regard for the total system reward. As a result, the MAP solution is likely to produce a smaller total system reward compared with the SAP solution. However, the amounts of the rewards earned by the players in a MAP should depend very much on the efficiency of the learning algorithm, and the total reward could at best be equal to the SAP

solution. Hence, the SAP solution should provide a very good upper bound for the results of the RL algorithm. An MDP formulation of the single-agent version of the problem, and also the CMDP/ILP model for the MAP can be found in Ravulapati *et al.* (2003).

#### 4. RL algorithms and implementation framework

In this section, we present average reward RL algorithms for both the single-agent and multi-agent versions of the inventory-planning problem. The single-agent RL algorithm for SAP is identical to an existing algorithm, Relaxed-SMART (Gosavi 1999) which is presented here for completeness. The multi-agent algorithm is a modified version of the algorithm that can be found in Ravulapati *et al.* (2003). The modification is that after learning is completed, instead of using a matrix game approach, the equilibrium value is obtained through simulation using the learnt  $Q$ -values. After the algorithms are presented, we present their implementation framework.

##### 4.1. SAP-RL algorithm

- (1) Initialize iteration or decision epoch count  $n=0$ , reinforcement values  $Q(l,a)=0$  for all states  $l \in E$  and all action combination vector  $a \in A(l)$ .  $A(l)$  is a set of all action combinations available to the centralized decision-maker at state  $l$ , and  $E$  denotes the state space. Set the average system reward at iteration  $n$ ,  $\rho_n=0$ . Initialize the input parameters  $(\alpha_0, \beta_0, \gamma_0, \alpha_\tau, \beta_\tau, \gamma_\tau)$  for choosing the learning rates ( $\alpha$  and  $\beta$ ) and exploration rate ( $\gamma$ ) according to the Darken–Chen–Moody (DCM) search-then-converge scheme (Darken *et al.* 1992) in step 2(d) below.

- (2) While  $n < \text{MaxSteps}$ , do If the system state at iteration  $n$  is  $i \in E$ ,
- (a) With probability  $(1 - \gamma_n)$ , choose the action  $a \in A(l)$  that maximizes  $Q(l, a)$ . With a probability of  $\gamma_n$ , choose a random (exploratory) action from the set  $A(l) \setminus a$ . Simulate the system with the chosen action until the next decision epoch is reached.
- (b) Let the system state at the  $(n+1)$ th decision epoch be  $q$ . Let  $r_{\text{imm}}(l, a, q)$  be the immediate reward earned as a result of reaching state  $q$  by choosing action  $a$  in state  $l$  at the  $n$ th epoch. Update reinforcement values  $Q(l, a)$  as
- $$Q(l, a) \leftarrow (1 - \alpha_n)Q(l, a) + \alpha_n(r_{\text{imm}}(l, a, q) - \rho_n + \max_a Q(l, a)), \quad (1)$$
- where  $\rho_n$  is the average reward for the system at iteration  $n$ .
- (c) Calculate the average reward for iteration  $(n+1)$  as
- $$\rho_{n+1} = (1 - \beta_n)\rho_n + \beta_n \left[ \frac{n\rho_n + r_{\text{imm}}(l, a, q)}{(n+1)} \right]. \quad (2)$$
- (d) Update the learning parameters  $\alpha_{n+1}$ ,  $\beta_{n+1}$  and the exploration parameter  $\gamma_{n+1}$  following the DCM scheme as given below.
- $$\alpha_{n+1} = \left( \frac{\alpha_0}{1 + u} \right), \quad \text{where } u = \left( \frac{n^2}{\alpha_\tau + n} \right), \quad (3)$$
- where  $\alpha_0$  is the starting value, and  $\alpha_\tau$  is a large constant chosen suitably to control the decay rate of  $\alpha$ . The other parameters ( $\beta_{n+1}$  and  $\gamma_{n+1}$ ) are also updated in the same manner.
- (e) Set  $q \leftarrow l$  and  $n \leftarrow n + 1$ .
- (f) If  $n < \text{MaxSteps}$ , go to Step 2(a), else go to Step 3.
- (3) Restart the simulation of the system with the final form of the Q-matrix  $\{Q(l) : l \in E\}$  to obtain the final estimate of the average system reward.

#### 4.2. MAP-RL algorithm

- (1) Let  $N$  be the number of players and the iteration count  $n=0$ . Initialize reinforcement values  $Q^i(l, a^1, a^2, \dots, a^N) = 0$  for all players  $i \in \mathcal{N} = \{1, 2, \dots, V\}$ , state  $l \in E$ , and actions  $a^i \in A^i(l)$ . Set the average reward for the players  $\rho^i = 0$  for all  $i \in \mathcal{N}$ . Define  $R^i(l, k) = \sum_{\{a^1, \dots, a^N\} \setminus a^i} Q^i(l, a^1, \dots, a^i = k, \dots, a^N)$ , which is the sum of all entries of reinforcement value matrix  $Q^i(l)$  corresponding to the action  $k$  of the player  $i$  in state  $l$ . Initialize the values of  $R^i(l, k)$

to zero. Initialize the input parameters  $(\alpha_0, \beta_0, \gamma_0, \alpha_\tau, \beta_\tau, \gamma_\tau)$  for choosing the learning rates ( $\alpha$  and  $\beta$ ) and exploration rate ( $\gamma$ ) according to the DCM search-then-converge scheme in Step 2(d) below.

- (2) While  $n < \text{MaxSteps}$ , do  
If the system state at iteration  $n$  is  $l \in E$ ,
- (a) For all  $i \in \mathcal{N}$ , compute the probability of choosing an action  $k \in A^i(l)$  in state  $l$  by player  $i$  as

$$p^i(l, k) = \begin{cases} (1 - \gamma_n) & \text{if } R^i(l, k) = \max_{k \in A^i(l)} R^i(l, k), \\ \frac{\gamma_n}{(|A^i(l)| - 1)}, & \text{if o/w,} \end{cases}$$

where  $|A^i(l)|$  denotes the cardinality of  $A^i(l)$ . The above says that the action  $a^i = k$  of agent  $i$  in state  $l$  for which the total reinforcement value (summed over all choice combinations of the other players) is the maximum over all  $k \in A^i(l)$  is chosen with probability  $(1 - \gamma_n)$ , and the rest of the actions are chosen with equal fraction of the exploration probability  $\gamma_n$ .

Now, for each player  $i \in \mathcal{N}$ , generate a random variable  $y^i$  from a uniformly distributed random variable  $U(0, 1)$ . Find, using  $p^i(l, k)$ , the cumulative probability distribution function  $F^i(l, k)$ , over the range of actions  $k \in A^i(l)$ . Choose action  $a^i = k$ , for which the condition  $F^i(l, k - 1) < y^i \leq F^i(l, k)$  is satisfied.

- (b) Simulate the system with the chosen actions for all the players (action vector  $a = (a^1, \dots, a^N)$ ) until the next decision epoch. Let the system state at the next decision epoch be  $q$  and  $r^i(l, a, q)$  be the immediate reward for player  $i$  corresponding to the action combination  $a$ .
- (c) Update the reinforcement value for the most recent state-action combination  $Q_{n+1}^i(l, a)$  and the average reward  $\rho_{n+1}^i$  for all the players  $i \in \mathcal{N}$  as follows:

$$Q_{n+1}^i(l, a) = (1 - \alpha_n)Q_n^i(l, a) + \alpha_n(r^i(l, a, q) - \rho_n^i + Q_{\text{exp}}^i(q)), \quad (4)$$

where

$$Q_{\text{exp}}^i(q) = \sum_{\phi \in \{A^1(q) \times \dots \times A^N(q)\}} (p^1(q, \phi_1) \times \dots \times p^N(q, \phi_N)) \times Q_n^i(q, \phi)$$

and  $\phi = (\phi_1, \phi_2, \dots, \phi_N)$  and  $\phi_i \in A^i(q)$ .

$$\rho_{n+1}^i = (1 - \beta_n)\rho_n^i + \beta_n \left[ \frac{n\rho_n^i + r^i(l, a, q)}{(n+1)} \right]. \quad (5)$$

(4) and (5) together represent the two-scale updating scheme of the reinforcement value function  $Q$ . The first equation is the asynchronous version of the average reward Bellman equation. The quantity  $Q_{\text{exp}}^i(q)$  represents the expected reinforcement value of state  $q \in E$  over all combinations of actions at the  $n$ th iteration.

- (d) Find the updated value of the learning parameters  $\alpha_{n+1}$  and  $\beta_{n+1}$ , and the exploration parameter  $\gamma_{n+1}$  using the DCM scheme as given below.

$$\alpha_{n+1} = \frac{\alpha_0}{1+u}, \quad \text{where } u = \frac{n^2}{\alpha_\tau + n},$$

where  $\alpha_0$  is the starting value, and  $\alpha_\tau$  is a large constant chosen suitably to control decay of  $\alpha$ . The other parameters  $\beta_{n+1}$  and  $\gamma_{n+1}$  are updated using the same approach as  $\alpha$ .

- (e) Set  $l \leftarrow q$  and  $n \leftarrow n + 1$ .  
 (f) If  $n < \text{MaxSteps}$ , go to Step 2(a), else go to Step 3.  
 (3) Restart the simulation of the system with the final form of the Q-matrices  $\{Q^i(l) : \forall i \in \mathcal{N}, l \in E\}$  to obtain an estimator of the average system reward.

#### 4.3. Solution framework using RL algorithms

The framework for the RL-based methodology for the non-cooperative inventory planning problem is presented in this section. This framework applies to both the single-agent and multi-agent versions of the problem. The implementation framework is depicted in figure 3. The framework has a policy-learning phase and a learnt-policy implementation phase. The latter is similar to the policy-learning phase without the reinforcement learning component. The policy-learning phase learns the policy, and the learnt-policy implementation phase simulates the learnt policy to estimate the long-run reward. The arrows in the figure indicate the flow of information among the elements within the phases. Shown within parentheses in each element is the software support that was used to implement that element. Brief descriptions of the elements are given below.

**4.3.1. Simulation.** This block is activated after the inventory decisions are made by the retailers. The customer-arrival processes during a business cycle in the inventory distribution system are simulated. *ARENA*<sup>®</sup> simulation software package was used to develop the simulation model. At the end of a business cycle, based on the action taken (inventories ordered) at the beginning of the cycle and the realized demand during the cycle, the simulation block sends the information about the

residual inventory and the residual demands to the transportation block.

**4.3.2. Transportation.** The transportation block decides the optimal sharing pattern for the residual inventories to meet the residual demands by solving the integer linear program (ILP). LP\_SOLVE (ftp://ftp.es.ele.tue.nl/pub/lp\_solve/), was used to solve the ILP. Visual Basic for applications (VBA), a back end for *ARENA*<sup>®</sup> was used as an interface between *ARENA*<sup>®</sup> and LP\_SOLVE. The optimal shipping pattern, the excess profit thus made, and the dual values of the constraints are received by the VBA from the LP\_SOLVE. The VBA module allocates the excess profit to the agents based on a dual-price-allocation rule and then sends the information regarding the starting and end states of the system, and the profit made by each agent to the Reinforcement Learning block.

**4.3.3. Reinforcement learning (RL).** The RL block updates the reinforcement value matrix of each agent using the reinforcement-learning algorithm and the information received from the Transportation block. This block then updates the values of the exploration and learning rates using the DCM scheme. Then, if the stopping criterion is not reached, it decides on the next set of actions for the retailers based on the ending state of the current cycle and the reinforcement values of the available actions. This set of actions is then sent to the simulation block. This cycle of Simulation  $\rightarrow$  Transportation  $\rightarrow$  RL  $\rightarrow$  Simulation continues until stable reinforcement values for all the state and action combinations are obtained.

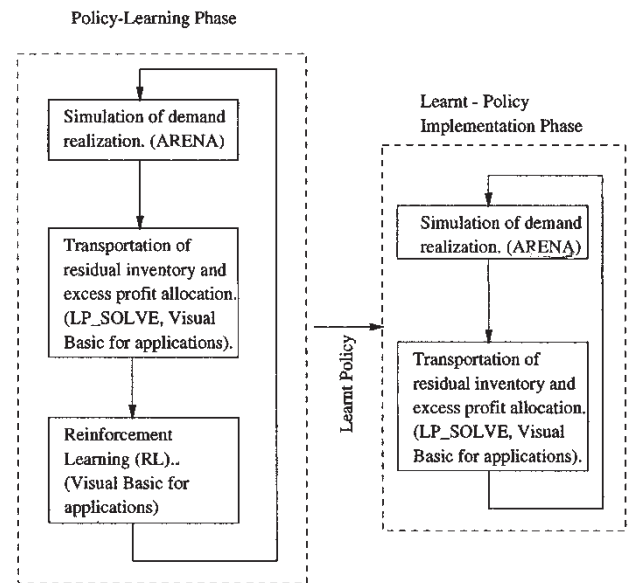


Figure 3. RL-based implementation framework.

5. Numerical results

In this section, the results of implementing the RL-based methodology on a set of numerical versions of the problem are presented. The numerical problems that are examined are identical to those studied in Ravulapati *et al.* (2003) with three-retailers and no warehouses. The reasons for not considering warehouses in the numerical study are: (1) a warehouse does not constitute a decision-maker, since inventory decisions are made only by the retailers; and (2) warehouses add significantly to the system state space increasing the computational burden. The upper bounds on the average system profits were first obtained by implementing the methodology using the SAP-RL algorithm. The methodology using the MAP-RL algorithm was then implemented for the numerical problems. For every problem, the design parameters for both SAP-RL and MAP-RL algorithms were selected using analysis of variance and regression analysis. Table 1 presents four different numerical problem scenarios and the corresponding optimal rewards obtained using the relative value iteration (RVI) technique which are adopted from Ravulapati *et al.* (2003). The salvage and transportation costs that are considered are also the same as in Ravulapati *et al.* (2003). In Section 5.1, we first discuss the implementation of the SAP-RL algorithm to develop useful upper bounds for system rewards. The results show that the SAP-RL algorithm provides a very useful means of obtaining an upper bound for the optimal system reward.

5.1. Obtaining reward upper bounds using the SAP-RL algorithm

The optimal profit values in table 1 represent the upper bounds for the average system profits of the three-retailer problem. However, as discussed earlier, RVI implementation suffers from both modelling and dimensional curses for large state spaces. In this section, the results of using a simulation-based SAP-RL algorithm for finding the upper bounds are presented.

Table 1. Optimal solution for the four test problems using the RVI algorithm

Test case	Unit cost	Profit (\$) per unit (% of unit cost)	Holding cost (\$) per unit (% of unit cost)	Average optimal profit (\$)	3σ limits
1	10	50	10	35.1798	0.56
2	10	50	15	33.5699	0.58
3	10	75	10	55.2584	0.79
4	10	75	15	53.1240	0.84

A designed experiment was conducted to obtain effective values of the design parameters for the algorithm.

The design parameters for an RL algorithm are the learning parameters  $\alpha$  and  $\beta$ , exploration parameter  $\gamma$ , and the parameters to control the decay rates  $\alpha_\tau$ ,  $\beta_\tau$ , and  $\gamma_\tau$ . We used identical values for the three decay parameters and denoted it by  $\theta_\tau$  in the analysis. Unit profit and unit holding cost, expressed as percentages of the unit cost, were also considered as factors in the experiment to assess any interaction of these factors with the design parameters. Table 2 summarizes the values of the factors at two levels that were considered for the experiment.

A one-quarter fraction of the  $2^6$  factorial design was used with  $I=ABCD=BCDF$  as the design generator. The experimental data are shown in table 3. The response variable used in the analysis is the difference in the average system profit obtained from the RVI algorithm and the SAP-RL methodology. The normal probability plot of the effects is shown in figure 4 (where only the outlying effects are labelled). The outlying interaction effects ( $bd$ ,  $bf$ ,  $af$ ,  $ab$ ,  $ae$ ) and all of the main effects were used in the analysis of variance (ANOVA). The remaining interactions were pooled to obtain an estimate of error variance. The ANOVA performed on the experimental data is given in table 4.

Two main factors (learning parameter  $C$  and the decay rate  $F$ ) were found to be significant. Two factor interactions,  $AE$ ,  $BD$ , and  $BF$ , were also found to be significant. For the purpose of optimizing the design parameters, a first-order regression equation was obtained using all the main effects and the significant interactions. For each of the four combinations of profit ( $A$ ) holding cost ( $B$ ) given in table 2, good values of the design parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\theta_\tau$  were obtained by optimizing the regression equation using EXCEL<sup>®</sup> Solver. We noted that, in general, lower-end values of the learning and exploration parameters were preferred over the values in their ranges.

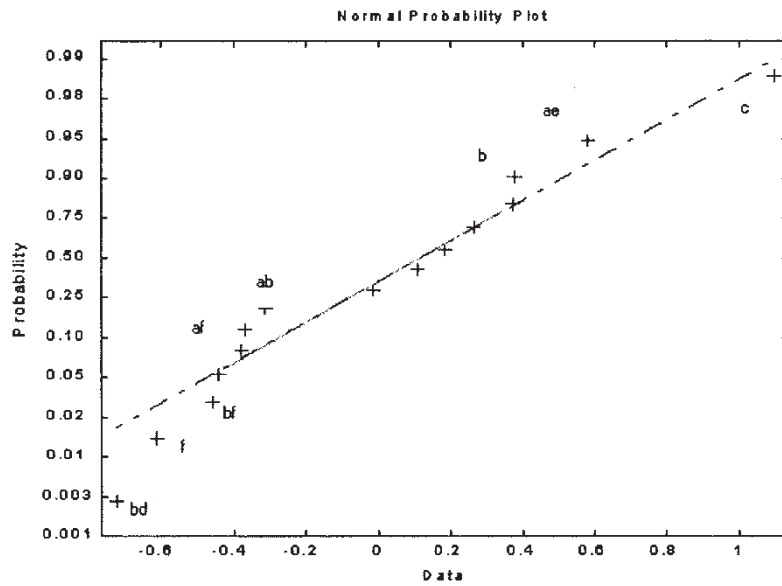
Table 5 shows the upper bounds of the average system profits obtained from the SAP-RL methodology using the designed parameters. A comparison of these results

Table 2. Summary of design factors

Index	Design factor	High	Low
A	Profit (% of cost)	75	50
B	Holding cost (% of cost)	15	10
C	$\alpha$ (learning parameter for $Q$ -values)	0.3	0.05
D	$\beta$ (learning parameter for average reward)	0.3	0.05
E	$\gamma$ (exploration parameter)	0.3	0.05
F	$\theta_\tau$ (decay rate for $\alpha, \beta, \gamma$ )	$9 \times 10^5$	$4.5 \times 10^5$

**Table 3. Average system profit achieved using the RVI algorithm and SAP-RL algorithm**

Treatment combination	RVI		SAP-RL		Difference (RVI-SAP)	Difference as % of RVI
	Average system profit	3 $\sigma$ limits	Average system profit	3 $\sigma$ limits		
(1)	35.18	0.56	35.08	0.61	0.10	0.28
a e	52.26	0.79	54.50	0.84	0.75	1.37
b e f	33.57	0.58	32.95	0.58	0.62	1.84
a b f	53.12	0.84	53.08	0.86	0.04	0.08
c e f	35.18	0.56	33.86	0.51	1.32	3.76
a c f	55.26	0.79	55.01	0.81	0.25	0.44
b c	33.57	0.58	30.82	0.58	2.75	8.18
a b c e	53.12	0.84	49.71	0.58	3.41	6.42
d f	35.18	0.56	35.12	0.23	0.06	0.16
a d e f	55.26	0.79	54.08	0.74	1.17	2.13
b d e	33.57	0.58	33.06	0.63	0.51	1.51
a b d	53.12	0.84	53.03	0.79	0.10	0.18
c d e	35.18	0.56	34.54	0.60	0.64	1.81
a c d	55.26	0.79	53.33	0.69	1.93	3.49
b c d f	33.57	0.58	32.56	0.53	1.01	3.01
a b c d e f	53.12	0.84	52.31	0.69	0.82	1.54



**Figure 4. Normal probability plot for yield difference of RVI and SAP-RL.**

with the optimal upper bounds from RVI (also shown in table 5) indicates that the SAP-RL methodology is an efficient alternative to the classical RVI algorithm for the inventory-planning problem.

*5.2. Results from the MAP-RL methodology*

In this section, the results obtained from the multi-agent versions of the numerical problems using the MAP-RL algorithm are presented. The design parameters for the MAP-RL algorithm were carefully selected

using a regression equation for the four test problems shown in table 1. Using these selected parameters, the average system profit for the test problems were obtained and compared with the corresponding optimal results.

A designed experiment identical to that used for the SAP-RL methodology was used to select the parameter values for the MAP-RL methodology. The experimental data are given in table 6, and the ANOVA result is shown in table 7.

Other than the interactions *AB*, *AD*, and *AF*, all others were pooled to obtain an estimate of error.

**Table 4. ANOVA for SAP-RL**

Source of variation	Degrees of freedom (df)	Sum of squares (SS)	Mean square SS/df	$F_o = MS_{\text{factor}}/MS_{\text{error}}$	$F_c$
A	1	0.1361	0.1361	1.624	7.708
B	1	0.576	0.576	6.881	7.708
C	1	4.807	4.807	57.389	7.708
D	1	0.567	0.567	6.764	7.708
E	1	0.57	0.57	6.80	7.708
F	1	1.493	1.493	17.832	7.708
AB	1	0.391	0.391	4.676	7.708
AE	1	1.363	1.363	16.275	7.708
AF	1	0.538	0.538	6.429	7.708
BD	1	2.069	2.069	24.709	7.708
BF	1	0.834	0.834	9.961	7.708
Error	4	0.335	0.0837		
Total	15	13.683			

**Table 5. Comparison of average system rewards obtained using the SAP-RL methodology and RVI algorithm**

Unit cost	(% of cost) Unit profit	(% of cost) Unit holding	RVI		SAP-RL		Difference (RVI-SAP)	Difference as % of RVI
			Average profit	3σ limits	Average profit	3σ limits		
10	50	10	35.1798	0.56	35.1741	0.61	0.0057	0.016
10	50	15	33.5699	0.58	33.4084	0.58	0.1615	0.481
10	75	10	55.2584	0.79	55.1151	0.80	0.1433	0.259
10	75	15	53.1240	0.84	53.1176	0.84	0.0064	0.012

**Table 6. Average system profit achieved using the RVI algorithm and MAP-RL algorithm**

Treatment combination	RVI		MAP-RL		Difference RVI-MAP	Difference as % of RVI
	Average system profit	3σ limits	Average system profit	3σ limits		
(1)	35.1798	0.56	35.1715	0.55	0.0083	0.02
a e	55.2584	0.79	55.1696	0.80	0.0888	0.16
b e f	33.5699	0.58	33.0876	0.69	0.482	1.44
a b f	53.124	0.84	53.1155	0.83	0.0085	0.02
c e f	35.1798	0.56	35.178	0.55	0.0018	0.01
a c f	55.2584	0.79	55.1323	0.79	0.1261	0.23
b c	33.5699	0.58	33.1814	0.61	0.3885	1.16
a b c e	53.124	0.84	53.1191	0.85	0.049	0.05
d f	35.1798	0.56	35.04	0.23	0.139	0.40
a d e f	55.2584	0.79	54.7548	0.92	0.504	0.91
b d e	33.5699	0.58	33.1875	0.63	0.382	1.14
a b d	53.124	0.84	52.718	0.91	0.406	0.76
c d e	35.1798	0.56	35.013	0.54	0.1668	0.47
a c d	55.2584	0.79	54.647	0.77	0.6114	1.11
b c d f	33.5699	0.58	32.6051	0.67	0.965	2.87
a b c d e f	53.124	0.84	53.0318	0.86	0.092	0.17

Significant main factors are the holding cost, *B*, and the learning parameter for average profit, *D*. Significant interactions are *AB* and *AF*. For the purpose of obtaining optimal design parameters using regression, a first-

order regression equation (6) (with  $R^2=0.87$ ) was developed as shown below. For the four combinations of the profit (*A*) and holding cost (*B*) given in table 1, the optimal values for the design parameters  $\alpha$ ,  $\beta$ ,

**Table 7.** ANOVA for MAP-RL

Source of variation	Degrees of freedom (df)	Sum of squares (SS)	Variance SS/df	$F_o$	$F_c$
A	1	0.030	0.030	3.893	5.987
B	1	0.073	0.073	9.502	5.987
C	1	0.007	0.007	0.919	5.987
D	1	0.291	0.291	37.719	5.987
E	1	0.054	0.054	7.016	5.987
F	1	0.004	0.004	0.556	5.987
AB	1	0.462	0.462	59.917	5.987
AD	1	0.023	0.023	3.034	5.987
AF	1	0.065	0.065	8.485	5.987
Error	6	0.046	0.008		
Total	15	1.057			

**Table 8.** Comparison of average system rewards achieved using the MAP-RL algorithm with the relative value-iteration algorithm

Unit cost	(% of cost) Unit profit	(% of cost) Unit holding	RVI		MAP-RL		Difference (RVI-MAP)	% difference
			Average profit	$3\sigma$ limits	Average profit	$3\sigma$ limits		
10	50	10	35.1798	0.56	35.1715	0.56	0.0083	0.024
10	50	15	33.5699	0.58	33.2035	0.61	0.3664	1.091
10	75	10	55.2584	0.79	55.1018	0.79	0.1566	0.283
10	75	15	53.1240	0.84	53.1162	0.84	0.0078	0.015

$\gamma$ , and  $\theta_\tau$  were obtained by optimizing the regression equation.

$$\begin{aligned}
Y = & 0.274 - 0.043X_A + 0.0677X_B + 0.021X_C \\
& + 0.135X_D - 0.058X_E + 0.0164X_F \\
& - 0.1699X_A X_B - 0.064X_A X_F.
\end{aligned} \tag{6}$$

Table 8 shows the average system profit obtained using the MAP-RL methodology for the test problems using the selected design parameters. It also provides a comparison of the average system profit obtained using the MAP-RL methodology with the corresponding optimal solutions obtained using relative value iteration technique.

From the results, it can be concluded that the RL-based methodology works quite well in dealing with non-zero sum stochastic game involved in the multi-retailer inventory-planning problem. However, the closeness of the MAP-RL solutions to the RVI solutions should not be considered as a general rule, since we feel that the nature of the inventory problem and the values of its parameter may have contributed to it. Since the performance of the RL algorithm is sensitive to its design parameters, the parameters should be selected with care. The next section shows how the RL-based

methodology can be used to study two very important aspects of stochastic game problems: (1) computation of short-term rewards of a game, and (2) evaluating the effects of one or more agents deviating from the equilibrium policy (perturbation analysis).

## 6. Other utilities of the RL methodology

In order to expound on the topics of short-term rewards and perturbation analysis, a different example problem with a somewhat larger state space was adopted. Average system reward obtained using the SAP-RL methodology was used as an upper bound for the corresponding multi-agent problems. In what follows, the example problem is briefly described, and its equilibrium reward is presented before examining the short-term rewards and carrying out perturbation analysis.

### 6.1. Problem description and analysis

A different set of numerical problems consisting of five agents were considered in this section. The parameters for the problems were chosen somewhat arbitrarily. It was assumed that the demand arrivals for the five agents are Poisson with parameters 1.89, 1.89, 1.72,

1.89, and 1.75, where the time unit is one business day. Each agent can order an inventory between 0 and 3 (four possible actions). All customers at a retail location are willing to be served from other locations in case of a stock-out situation. The cost, selling price, and holding cost of an item are the same across all agents. The salvage price of an unsold item is the same as the cost of the item. Four test problem scenarios with different cost, profit, and holding cost values are given in table 9. The transfer costs for moving an item between retail locations are fixed and are given in table 10. Table 11 provides a summary of the system average profits and corresponding upper bounds for the four test problem scenarios. On average, the multi-agent RL solution is within 1.8% of its single-agent RL upper bound. Figure 5 provides a plot of the upper bound of the average system profit (from SAP-RL) and the average profit for the multi-agent problem (MAP-RL) for the first 3000 business cycles of problem scenario 4.

6.2. Short-term reward analysis

Short-term game return is an important deciding factor for an agent being or not being in a game in the long run. The profits made by the individual agents during the learning phase may eventually be different, as all the agents learn their best policies, and the game attains an equilibrium state. A unique ability of the RL-based methodology, to study the short-term rewards obtained by the agents during the learning phase of a game, is demonstrated here with the test problem (scenario 3 from table 9). Average profit data for the first 100 business cycles of the learning phase were collected for all the five agents separately. A second set of data for the first 100 business cycles was also collected while simulating the learnt equilibrium policy. Plots of the

two data sets are shown in figure 6. Agents 1, 4, and 5 attained 10.66%, 7.32%, and 9.95% higher average profits, respectively, from the learned policy than their profits during their first 100 learning cycles. The average profit of agent 2 did not vary significantly between the

Table 9. Test scenarios

Scenario number	Unit cost	Unit profit	Unit holding cost
1	50	25	4
2	500	50	4
3	1000	100	4
4	5000	500	4

Table 10. Transfer cost matrix

–	A1	A2	A3	A4	A5
A1	–	5	7	6	5
A2	5	–	4	5	6
A3	7	4	–	8	7
A4	6	5	8	–	6
A5	5	6	7	6	–

Table 11. System profit for SAP-RL and MAP-RL

Scenario number	SAP-RL (Upper bound)	MAP-RL	% deviation of MAP-RL from the upper bound
1	192.76	188.98	1.96
2	420.06	413.04	1.67
3	849.51	825.80	2.79
4	4186.03	4145.42	0.97

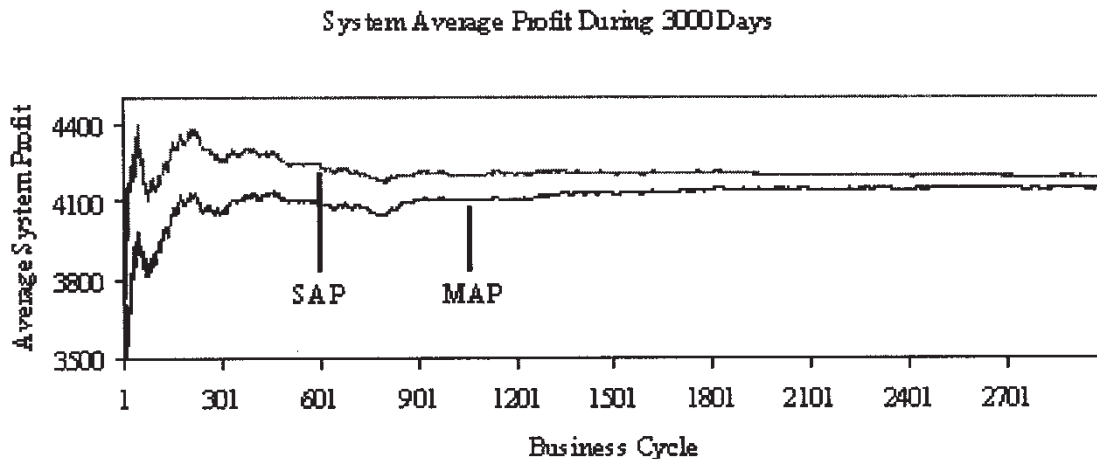


Figure 5. Comparison of average system rewards obtained using the SAP-RL methodology and the MAP-RL methodology for problem scenario.

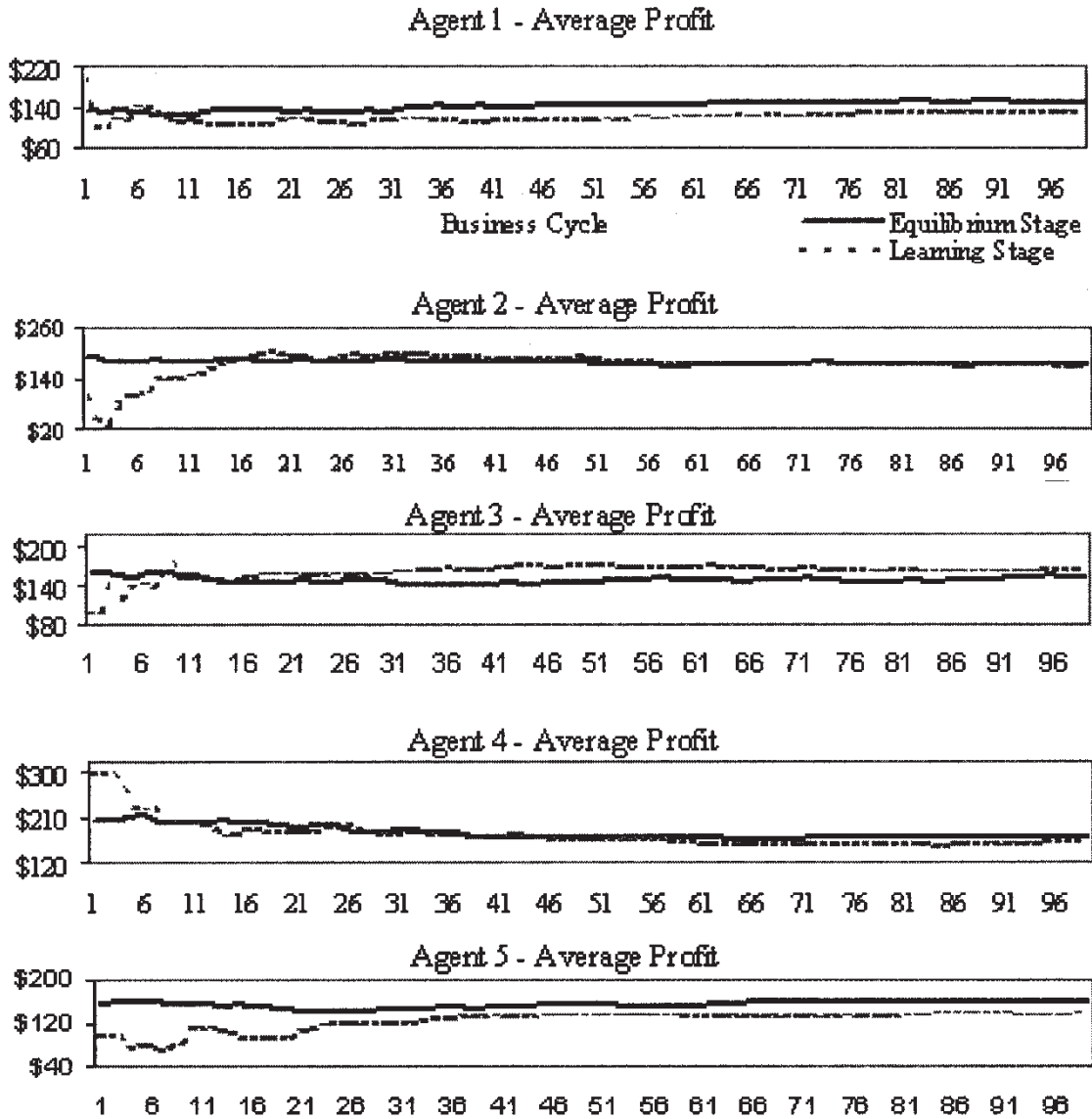


Figure 6. Comparison of short-term rewards with equilibrium rewards for scenario 3.

learning and the equilibrium phases (within 0.14%). The average profit of agent 3 actually decreased on attaining the equilibrium by 6.03%. Short-term non-cooperative game results, as shown in figure 6, could be very helpful for the agents to make long-term business decisions.

6.3. Sample perturbation analysis

A sample numerical perturbation analysis demonstrating the effect on the system's average profit when subsets of the agents in a game choose to deviate from their equilibrium policies is presented in this section. This analysis was performed by first randomly allowing an agent to deviate from their equilibrium inventory policy by one unit. Then, two agents were allowed to deviate, and finally all five agents were allowed to

deviate from their equilibrium policies. Figure 7 shows the outcome of this analysis. It can be seen that the loss of profit resulting from perturbation is more prominent when the agents order less than the equilibrium policy than when the agents order more than the equilibrium policy. This may partly be due to the presence of a ceiling on the maximum inventory the agents can order. Also, in some of the states, the equilibrium policy of some agents itself could be at this ceiling limit, allowing no further increase.

7. Conclusion

This paper presents a simulation-based modelling approach for assessing short-term and equilibrium

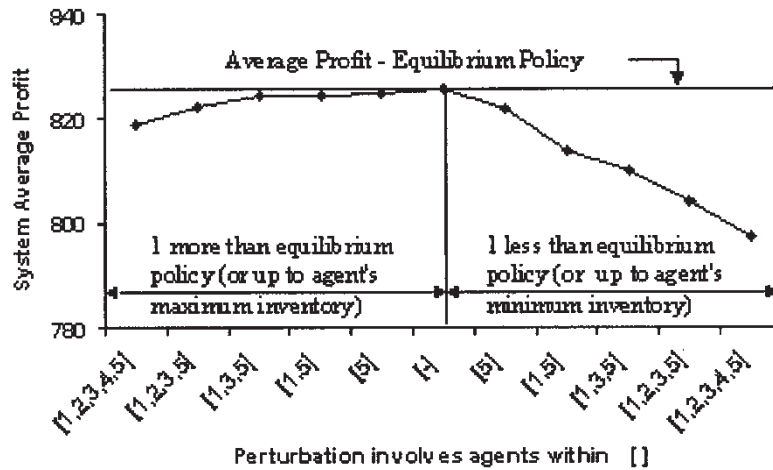


Figure 7. A sample perturbation analysis for scenario 3.

behaviour of non-zero sum stochastic games that are found commonly in competing supply-chain organizations. The approach uses reinforcement-learning (RL) algorithms that were adopted from the related literature and then suitably modified. Since the RL-based approach uses simulation as its modelling tool and evaluates the decision options of a game in an asynchronous manner, it avoids to a significant extent the curses of modelling and dimensionality that are normally associated with real-life non-zero sum stochastic game problems. Also, the RL algorithms are founded on the theory of dynamic programming and the theory of stochastic approximation, and hence in general have good optimality properties.

The problem studied in this paper is an  $N$ -retailer,  $W$ -warehouse non-cooperative inventory-planning problem. The problem was modelled as a competitive Markov decision process, for which a multi-agent reinforcement-learning approach was developed to obtain an equilibrium system reward. To obtain an upper bound on the reward, a single-agent abstraction of the multi-agent problem was conceived for which a single-agent RL approach was developed. The procedure was evaluated on a set of numerical test problems. The results show that the RL-based approach holds promise in being able to provide a useful means of studying real-life non-zero sum stochastic games for which no computationally feasible approach exists in the current literature. The RL-based approach was also shown to be a useful tool in evaluating short-term rewards of a stochastic game which is more difficult to assess than the equilibrium behaviour of a game. For a discussion on different approaches that can be used to deal with very large state space, refer to Gosavi *et al.* (2002) and Ravulapati *et al.* (2003).

## References

- ABOUNADI, J., BERTSEKAS, D., and BORKAR, V. S., 1998, Learning algorithms for Markov decision processes with average cost. *LIDS-P-2434*, Laboratory for Information and Decision Systems (Cambridge, MA: MIT).
- ANUPINDI, R., BASSOK, Y., and ZEMEL, E., 2001, A general framework for the study of decentralized distribution systems. *Journal of Manufacturing and Service Operations Management*, **3**, 349–368.
- BELLMAN, R. E., 1957, *Dynamic Programming* (Princeton, NJ: Princeton University Press).
- CRITES, R., and BARTO, A., 1996, Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.) *Advances in Neural Information Processing Systems 8* (Cambridge, MA: MIT) pp. 1017–1023.
- DARKEN, C., CHANG, J., and MOODY, J., 1992, Learning rate schedules for faster stochastic gradient search. In D. A. White and D. A. Sofge (eds.) *Neural Networks for Signal Processing 2—Proceedings of the 1992 IEEE Workshop* (Piscataway, NJ: IEEE Press).
- DAS, T. K., GOSAVI, A., MAHADEVAN, S., and MARCHALLECK N., 1999, Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science*, **45**, 560–574.
- EREV, I., and ROTH, A. E., 1998, Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *The American Economic Review*, **88**, 848–881.
- FILAR, J., and VRIEZE, K., 1997, *Competitive Markov Decision Processes* (New York: Springer).
- GOSAVI, A., 1999, An algorithm for solving semi-Markov decision problems using reinforcement learning: Convergence analysis and numerical results. *PhD dissertation*, University of South Florida.
- GOSAVI, A., BANDLA, N., and DAS, T. K., 2002, A reinforcement learning approach to airline seat allocation for multiple fare classes with overbooking. *IIE Transactions on Operations Engineering* (Special Issue on Advances on Large Scale Optimization for Logistics, Production, and Manufacturing Systems), **34**, 729–742.
- GOSAVI, A., DAS, T. K., and SARKAR, S., In press. A simulation-based learning automata framework for solving semi-markov decision problems under long-run average cost. *IIE Transactions on Operations Engineering*.
- HU, J., and WELLMAN, M. P., 1998, Multi-agent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 242–250.

- LITTMAN, M. L., 1994, Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163.
- PATERNINA, C. D., and DAS, T. K., 2000, Intelligent dynamic control policies for serial production lines. *IIE Transactions*, **33**, pp. 65–77.
- RAVULAPATI, K. K., RAO, J. J., and DAS, T. K., in press. A reinforcement learning approach to stochastic business games. *IIE Transactions on Scheduling and Logistics*.
- RIPLEY, B. D., 1996, *Pattern Recognition and Neural Networks* (Cambridge: Cambridge University Press).
- ROBBINS, H., and MONRO, S., 1951, A Stochastic Approximation Method. *Annals of Mathematics and Statistics*, **22**, 400–407.
- SINGH, S., and BERTSEKAS, D., 1996, Reinforcement learning for dynamic channel allocation in cellular telephone systems. *Neural Information Processing Systems* (Cambridge, MA: MIT Press).